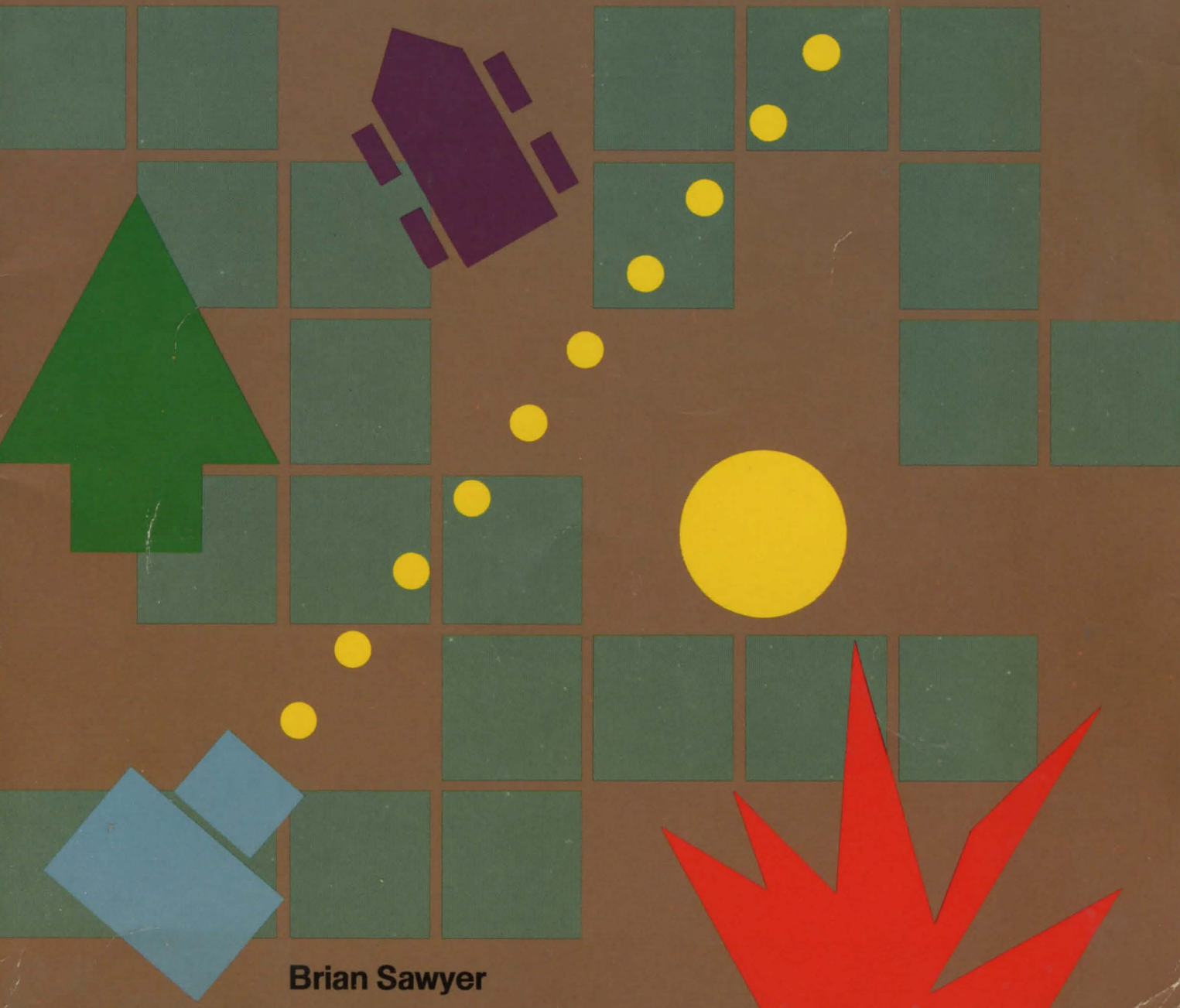
Osborne McGraw-Hill

THE Coleco™ Adam™ Entertainer

• GAMES • GRAPHICS • SOUND



Brian Sawyer

The Coleco™ Adam™ Entertainer

The Coleco™ Adam™ Entertainer

Brian Sawyer

Osborne/McGraw-Hill
Berkeley, California

Published by
Osborne/McGraw-Hill
2600 Tenth Street
Berkeley, California 94710
U.S.A.

For information on translations and book
distributors outside of the U.S.A., please write to
Osborne/McGraw-Hill at the above address.

The Coleco™ Adam™
Entertainer

1234567890 DLDL 8987654

ISBN 0-88134-134-7

Karen Hanson, Acquisitions Editor
Brad Hellman, Technical Editor
Ted Gartner, Copy Editor
Judy Wohlfrom, Text Design
Yashi Okita, Cover Design

Table of Contents

Introduction: The Coleco Adam Entertainer

The Coleco Adam Entertainer contains thirty programs that provide entertainment while educating the user on the special features of the Coleco Adam. Each program uses one of the Adam's special abilities, such as graphics, the joystick controller, and printer, and each is presented with instructions and listings, as well as an explanation of how the program operates. This book will be a valuable aid to Adam owners of all levels of programming experience. Even if you don't know how to program, you can enjoy hours of entertainment playing the many games in this book, such as stalking killer bees, shooting down space junk, guessing secret codes of fruit, and other adventures. Aspiring programmers can also learn programming techniques for the Adam that can be used in their programs.

A portion of this book contains programming tools. These tools are useful *subroutines* (sections of a program that perform a certain task) that you can include in your programs. Each of these programs contains an example of how the tool is being used in an interesting way.

The Coleco Adam Entertainer is divided into thirty chapters, each containing a program. Every chapter begins with *instructions* that describe what the program does and how to operate it. Next comes the program's *technical description.* This section explains how the program works. Important commands, subroutines, and variables are identified in this section. A line-by-line description of the program is also included. Finally, a complete program listing is provided.

These programs have been written to operate on Coleco's SmartBASIC Version 1.0, and most of them utilize the Adam's color, graphics, game controllers, and printer. To use a program from the Coleco Adam Entertainer, you must first load the SmartBASIC into your Adam. This is done by placing your SmartBASIC pack into the data drive and pressing the computer reset button. When the SmartBASIC heading appears at the top of the screen, the language has been successfully loaded. Now you are ready to enter a program from the book. Carefully type in the program listing and check each line after you entered it for any mistakes. Beware of common errors, such as mistaking the number one (1) for the letter (l), or mistaking a zero (0) for the letter (O). A single mistake in the program can result in the program not operating properly, or even not at all.

After you have entered in the program, take some time to check it over for correctness by making sure that you haven't missed any lines. Now save the program by inserting a blank digital data pack in the data drive. Then choose a file name for the program. The name of a file can be as many as ten characters long. These characters can be upper- or lowercase, numbers, or a period. You can't begin the name of a file with a number, nor can there be any blanks in it. We suggest that you use the name of the program in the book and modify the name if it has any blanks or if it exceeds ten characters. For example, you could name the Blackjack program BLACKJACK or the Shape Designer program SHAPE. When you have chosen a file name, type **SAVE** *file-name* and press RETURN. Wait for the Adam to finish saving the program and then remove the data pack. Now if you ever need to retrieve this copy of the program, simply insert the digital data pack in the drive, type **LOAD** *file-name,* and press RETURN.

When the program has been entered, inspected, and saved on tape, it is time to execute it. When the SmartBASIC prompt appears, type **RUN** and press RETURN. The Introductory Screen should appear. The screen contains the program title and a message asking you to press the left or right controller button on controller 1 when you are ready to play.

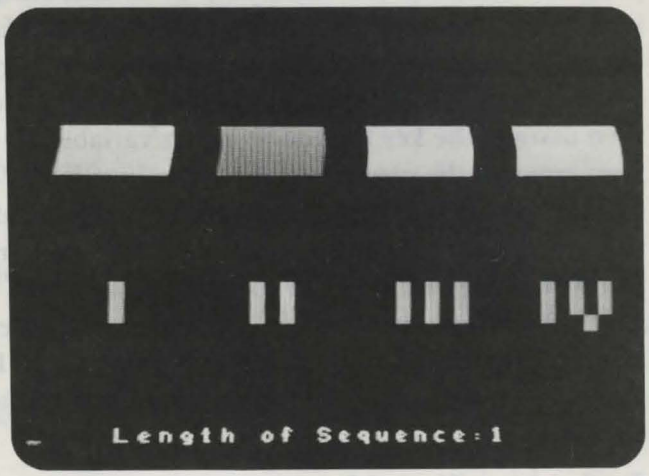
There are some mistakes that you might have made in the program that will cause the Adam to *lock up* when the program is executed. When the Adam locks up, it will not respond to any command. To restore the operation of the computer, press the reset button and then load SmartBASIC. This process will destroy the program that you had in memory, but since you saved it on tape, all you need to do is reload it. Correct the error that caused the computer to lock up, and save the new version on tape with the same name. If the program still behaves improperly, but does not lock up, you can terminate it immediately by holding down the CONTROL key and pressing C. After the program has been terminated, you will need to clear the display on the screen so you can correct the error. To do so, type the command **TEXT** and press RETURN.

Continue to correct any errors until the program works properly and then save a final copy of the program on a data pack, again with the same name.

Every program contains a section of lines at the beginning called the Standard Introduction. The Standard Introduction displays the title of the program and sets some variables that are used in every program. The Standard Introduction is located from line 1 to 95 in every program. One good idea is to type in the Standard Introduction separately and save it on tape. Now each time you type in a program, load the Standard Introduction first; rather than typing it, change the title to the appropriate program, and then enter the program beginning after line 95. For more information, see the chapter on the Standard Introduction.

The game controllers are one of the Adam's most useful features for games and are used in many of the programs. Your Adam comes with two game controllers that are identical. There are two sockets on the side of the Adam, one for Game Controller 1, and one for Game Controller 2. Make sure the Controller is plugged firmly into the socket when you are using it. Each Controller has a keypad on its face that contains the numbers 0 through 9 and the characters * and #. This keypad can be used to enter numbers into the Adam without having to use the keyboard. Each Controller has two buttons on it, one on the left side, and one on the right. Above the keypad is the Controller joystick, which can be moved up, down, left, right, or at a diagonal.

The Adam's printer is used by the programs that require a printout. Before you use the printer, make sure it has paper in it and is loaded in straight so that printing will not occur at an angle. Whenever the printer is printing and you want it to stop, hold down the CONTROL key and press S. To resume printing, press the CONTROL key and S again.



Memory Mix

Program 1

Memory Mix tests your ability to remember and repeat a sequence of colors flashed on the screen of your Adam. The game displays four white boxes on the screen. Below each box there is a Roman numeral. These boxes correspond to the first four function keys. The Adam will flash a color in each box in a random sequence. You must remember the pattern and repeat it by pressing the function keys.

The game starts with the Adam flashing a single color in a box. Each time you replay the pattern correctly, the sequence increases by one flash. If you make a mistake, you must start again with one box. When you want to quit, type Q.

Technical Description

The four colored boxes in this game are drawn quickly on the Adam's low-resolution graphics screen by using the HLIN command. Executing HLIN x,x+7 AT y draws a horizontal line of length 7 at the vertical position of y. The

line starts at screen position x,y and ends at x+7,y. Using a FOR-NEXT loop, the program increases the value of y by 5, executing the HLIN command at each increment. This method draws a 7×5 box much faster than it could be done with PLOT command.

The GET a$ command in line 160 waits for the player to press a key and then assigns the key's value to string variable a$. To see which function key has been entered, the program finds the value of ASC (a$). This command returns an integer value that corresponds to the character a$. When this value equals 129, function key I has been pressed. A value of 130 corresponds to function key II, 131 to function key III, and 132 to function key IV.

The random number function is used to ensure that the memory pattern will be different each time the game is played. In line 120, the RND function picks a number from 1 to 4 that corresponds to the next box to be flashed.

Important Variables

Variable	Function
cl(*number*)	Array containing the color of each of the four boxes, where *number* equals four entries.
max	Maximum length of the pattern.
n	Number of the next box to be flashed.
q	Length of the current memory pattern.
s(*number*)	Array containing numbers from 1 to 4 that make up the memory pattern. The value of *number* is from the variable max.
x,y	Screen position of the box being drawn, where x represents the horizontal coordinate and y the vertical coordinate.

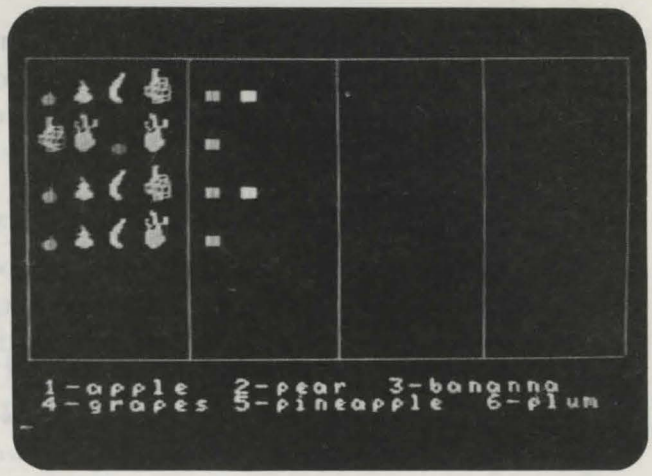
Program Description

Lines	Function
1-95	Standard introduction.
100-117	Calls subroutine that will initialize variables and draw the boxes.
118-140	Play the current pattern on the screen.

150-200	Allow the user to play back the pattern on the function keys while checking for mistakes.
300-307	Print message informing the player of his mistake.
1000-1010	Subroutine to enter low-resolution graphics mode.
1100-1110	Subroutine to read the four colors (magenta, dark blue, dark red, and dark green) into the array cl.
1200-1280	Subroutine to play one flash of the pattern on the screen.
1300-1410	Subroutine to draw Roman numerals at the bottom of the screen.

```
  1 REM  (c) 1984 by Brian Sawyer
  5 TEXT
 10 pm$ = "Memory Mix"
 15 FOR n = 1 TO 4
 20 PRINT CHR$(162)
 25 NEXT n
 30 tb = 16-INT(LEN(pm$)/2)
 35 PRINT TAB(tb); pm$
 40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
 45 PRINT " Hit Controller Button to Play"; CHR$(160)
 50 iv = 0: r = 0
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
    AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 GOSUB 1000
110 GOSUB 1100
112 GOSUB 1300
115 COLOR  = 15
116 PRINT: PRINT: PRINT: PRINT CHR$(160)
117 FOR n = 1 TO 4: GOSUB 1200: NEXT n
118 FOR t = 1 TO 800: NEXT t
120 n = INT(RND(1)*4+1)
122 PRINT "    Length of Sequence:"; q+1; CHR$(160)
125 q = q+1: s(q) = n
130 FOR m = 1 TO q
132 n = s(m): PRINT CHR$(7);
```

```
 135 COLOR  = cl(n): GOSUB 1200
 137 FOR t = 1 TO 300: NEXT t
 140 NEXT m
 150 FOR m = 1 TO q
 160 GET a$
 165 IF a$ = "q" OR a$ = "Q" THEN  TEXT: END
 170  n = ASC(a$)-128
 172 IF n < 1 OR n > 4 THEN 160
 175 COLOR  = cl(n)
 176 PRINT CHR$(7);
 180 GOSUB 1200
 190 IF n <> s(m) THEN 300
 200 NEXT m
 210 GOTO 118
 300 PRINT " MISTAKE !!"
 320 q = 0
 330 FOR d1 = 1 TO 2000: NEXT d1
 340 HOME
 350 PRINT " NEW SEQUENCE:"
 360 FOR d1 = 1 TO 1000: NEXT d1
 370 GOTO 115
1000 GR
1010 HOME: RETURN
1100 max = 100: DIM s(max)
1105 DIM cl(4): FOR n = 1 TO 4: READ cl(n): NEXT n
1106 DATA  1,2,3,4
1110 RETURN
1200 PRINT CHR$(7)
1205  x = (n-1)*10+2
1210 FOR y = 10 TO 15
1220 HLIN x, x+7 AT y
1230 NEXT y
1235 FOR t = 1 TO 500: NEXT t
1240 COLOR  = 15
1250 FOR y = 10 TO 15
1260 HLIN x, x+7 AT y
1270 NEXT y
1280 RETURN
1300 COLOR  = 7
1310 VLIN 28, 33 AT 5
1320 VLIN 28, 33 AT 14
1330 VLIN 28, 33 AT 16
1340 VLIN 28, 33 AT 24
1350 VLIN 28, 33 AT 26
1360 VLIN 28, 33 AT 28
1370 VLIN 28, 33 AT 34
1380 VLIN 28, 32 AT 36
1390 VLIN 28, 32 AT 38
1395 PLOT 37, 32
1400 PLOT 37, 33
1410 RETURN
```



Fruit Detective

Program 2

Rather than solving murders or finding lost cats, you, as the detective, have a different job: to guess a pattern of fruit. The Adam has picked fruits from a collection of six kinds: apples, pears, bananas, grapes, pineapples, and plums. You must identify the four fruits and their arrangement by using deductive logic.

Begin by entering your first guess of the fruit pattern using the keypad on game controller 1. This is done by entering a 4-digit sequence with each digit corresponding to a fruit (1=apple, 2=pear, 3=banana, 4=grapes, 5=pineapple, 6=plum). As you press each number on the keypad, the corresponding fruit will be displayed on the screen. The Adam will then rate your guess by displaying the clues to the right of the fruit. One white box will be printed for each fruit whose identity and position have been correctly guessed, and one green box will be printed for each fruit that has been identified correctly but positioned incorrectly. These clues will also be printed on your printer for you to study.

Technical Description

The Adam's shape tables are used to draw the different fruits. Six shape tables (one for each fruit) are read into the Adam's memory in lines 8000-8130. The data statements for the six fruits appear in lines 9000-9576.

The HCOLOR command in line 1416 sets the color of the shape table that is about to be drawn. Executing DRAW fr AT *x,y* draws the fruit at position *x,y* on the high-resolution screen. The variable fr in DRAW fr AT *x,y* contains the number corresponding to the specific fruit to be drawn. For example, if fr equals 1, an apple will be drawn at position *x,y*. The *x* in DRAW fr AT *x,y* represents the horizontal coordinate and *y* represents the vertical coordinate. In the program, *x* and *y* are replaced by the appropriate variable.

The printer prints the player's guesses and gives clues throughout the game. The user can modify the program, however, so that it will not print anything on the printer. This is done by changing the value assigned to variable ap in line 105 from 1 to 0. Now the command PR #ad in lines 205, 1700, and 1750 will no longer activate the printer.

Important Variables

Variable	Function
a(*number*)	Array corresponding to the secret fruit pattern. The size of this array is 4.
ap	Indicates whether the printer should be used.
cl(*number*)	Array of colors for each of the six fruits. Since there are six different fruits, the size of the array is 6.
fr	Number of the fruit currently being drawn.
g(*number*)	Array of numbers corresponding to a player's current guess of the secret pattern. The size of the array is 4.
gn	Number of guesses shown on the screen.
gs	Current number of guesses.
w1	Number of fruits in the current guess that exist in the pattern but are in the wrong place.
w2	Number of fruits in the current guess that are in the correct place.

x,y	Screen position of the fruit currently being drawn; x represents the horizontal axis and y the vertical axis.

Program Description

Lines	Function
0	Shape table memory pointer.
1-95	Standard introduction.
100-130	Main program.
200-260	Message telling the player he has won.
1000-1050	Subroutine to enter high-resolution graphics mode; initializes arrays.
1100-1150	Subroutine to draw a grid on the screen using the HPLOT TO command.
1200-1230	Subroutine to create a random secret pattern.
1300-1320	Subroutine to find the position for the fruit to be drawn.
1400-1440	Subroutine to read the player's guess.
1500-1590	Subroutine to rate the player's latest guess.
1600-1660	Subroutine to draw the clues on the screen.
1700-1770	Subroutine to print the clues on the printer.
1800-1830	Subroutine to print the menu.
7000	Data that represents the color of the fruit.
8000-8130	Subroutine to load the shape tables into memory.
9000-9576	Shape table data.

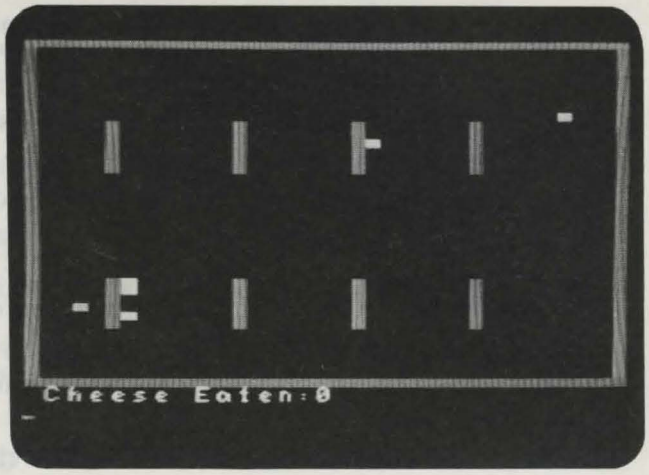
```
 0 HIMEM :41000
 1 REM  (c) by Brian Sawyer
 5 TEXT
10 pm$ = "Fruit Detective"
15 FOR n = 1 TO 4
```

```
 20 PRINT CHR$(162)
 25 NEXT n
 30 tb = 16-INT(LEN(pm$)/2)
 35 PRINT TAB(tb); pm$
 40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
 41 NORMAL
 42 d = d+1: IF d/2 = INT(d/2) THEN  INVERSE
 45 PRINT " Hit Controller Button to Play"; CHR$(160)
 50 iv = 0: r = 0
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
    AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 GOSUB 1000
110 ap = 1
111 GOSUB 8000
115 GOSUB 1100
116 GOSUB 1200
117 GOSUB 1300
118 GOSUB 1400: GOSUB 1500
119 IF w2 = 4 THEN 200
120 GOSUB 1800
125 gs = gs+1
130 gn = gn+1: GOTO 117
200 HOME
205 PR #ap
210 PRINT " YOU GUESSED THE FRUITS!!"
215 PR #0
220 PRINT "Would you like to play again?"
230 GET a$
240 IF a$ = "Y" OR a$ = "y" THEN 95
250 TEXT
260 END
1000 HGR
1010 HCOLOR  = 15
1015 DIM g(4), a(4), cl(6), h(4)
1020 SCALE  = 1: ROT  = 0
1046 FOR n = 1 TO 6
1047 READ cl(n)
1048 NEXT n
1050 RETURN
1100 HGR: HCOLOR  = 1
1110 HPLOT 10, 10 TO 250, 10
1115 FOR x = 10 TO 250 STEP 60
1120 HPLOT x, 10 TO x, 150
1130 NEXT x
1140 HPLOT 10, 150 TO 250, 150
1145 GOSUB 1800
1150 RETURN
```

```
1200 FOR n = 1 TO 4
1210 a(n) = INT(RND(1)*6+1)
1220 NEXT n
1225 gn = 1: gs = 1
1230 RETURN
1300 IF gn = 13 THEN  gn = 1: GOSUB 1100
1305 y = (gn-1)*22+30: x = 17
1310 IF gn > 6 THEN  x = 135: y = (gn-7)*22+30
1320 RETURN
1400 REM  Read guess from player
1405 FOR n = 1 TO 4
1410 pd = PDL(13): IF pd = 15 THEN 1410
1412 IF pd = 0 OR pd > 6 THEN 1410
1413 g(n) = pd
1415 fr = pd: IF fr = 6 THEN  fr = 1
1416 HCOLOR  = c1(pd)
1420 DRAW fr AT x+(n-1)*13, y
1422 PRINT CHR$(7);
1425 FOR t = 1 TO 300
1427 NEXT t
1430 NEXT n
1440 RETURN
1500 w1 = 0: w2 = 0
1505 GOSUB 1700
1507 FOR n = 1 TO 4
1508 h(n) = 1
1509 NEXT n
1510 FOR n = 1 TO 4
1520 IF g(n) = a(n) THEN  w2 = w2+1: h(n) = 0: g(n) = 0
1525 NEXT n
1526 FOR m = 1 TO 4
1530 FOR n = 1 TO 4
1535 IF h(n) = 0 THEN 1560
1545 IF a(n) = g(m) THEN  w1 = w1+1: h(n) = 0: GOTO 1570
1560 NEXT n
1570 NEXT m
1580 GOSUB 1600
1585 GOSUB 1750
1590 RETURN
1600 cx = 60
1605 HCOLOR  = 1
1606 IF w1+w2 = 0 THEN  RETURN
1610 FOR n = 1 TO w1+w2
1615 IF n > w1 THEN  HCOLOR  = 3
1620 FOR h = 1 TO 5
1630 HPLOT x+cx, y-h TO x+cx+6, y-h
1640 NEXT h
1645 cx = cx+14
1650 NEXT n
1660 RETURN
1700 PR #ap
1710 PRINT "Guess #"; gs; ": ";
1720 FOR d = 1 TO 4
```

```
1730 PRINT g(d); "   ";
1740 NEXT d: PRINT
1742 FOR dl = 1 TO 1000: NEXT dl
1745 PR #0: RETURN
1750 PR #ap
1755 PRINT "Fruits in correct position: "; w2;
1757 PRINT " Fruits that exist but in wrong position :"; w1
1758 FOR dl = 1 TO 1000: NEXT dl
1760 PR #0
1770 RETURN
1800 HOME
1810 PRINT " 1-apple  2-pear  3-banana "
1820 PRINT " 4-grapes 5-pineapple  6-plum"
1830 RETURN
7000 DATA 15,13,13,14,11,9
8000 qs = 5: pt = 16766: tl = 41000
8005 of = qs*2+3
8010 by = tl: GOSUB 8100
8015 POKE pt, lo: POKE pt+1, hi
8020 POKE tl, qs
8030 FOR n = 1 TO qs
8040 by = of: GOSUB 8100
8050 POKE tl+n*2, lo: POKE tl+n*2+1, hi
8060 READ a: POKE tl+of, a: of = of+1
8065 IF a <> 0 THEN 8060
8070 NEXT n
8080 RETURN
8100 hi = INT(by/256): lo = by-hi*256
8120 RETURN
8130 RETURN
9000 DATA 6
9003 DATA 45,61,60,63,45,45,61,60,63,47,45
9006 DATA 45,39,63,63,45,45,47,61,60,63,45
9009 DATA 36,4,0
9035 DATA 45,39,63,47,45,45,61,60,63,63,45
9038 DATA  45,45,39,63,63,47,45,45,61,60,63
9041 DATA 47,44,61,60,47,61,45,47,61,60,47
9044 DATA 36,39,4,0
9070 DATA 36,61,62,37,63,53,44,39,61,47,44
9073 DATA 47,63,37,47,47,47,61,60,45,39,47
9076 DATA 45,60,39,45,37,63,37,37,44,62,62
9079 DATA 62,62,54,54,54,54,53,45,60,7,0
9105 DATA 41,45,60,60,55,36,60,55,45,45,45
9108 DATA 45,54,63,39,36,36,45,53,54,63,63
9111 DATA 39,36,60,63,55,54,45,36,36,44,45
9114 DATA 45,54,62,39,36,36,45,53,54,53,62
9117 DATA 63,63,39,36,36,44,39,36,53,54,54
9120 DATA 36,36,44,4,0
9146 DATA 45,39,63,45,45,39,63,63,45,45,45
9149 DATA 39,63,63,47,45,45,45,39,63,63,63
9152 DATA 101,45,45,61,55,61,45,39,60,63,63
9155 DATA 55,44,55,36,45,54,47,36,45,45,60
9158 DATA 63,63,45,37,36,39,36,60,36,46,53
9161 DATA 54,54,53,54,36,44,37,36,53,54,63
```

```
9164 DATA  55,55,54,63,39,36,61,60,36,37,0
9503 DATA 109,9,173,18,62,62,55,63,39,63,60
9506 DATA 220,36,36,44,45,45,45,45,45,45,53
9509 DATA 54,54,54,54,62,63,63,63,63,63,63
9512 DATA 39,36,36,0
9538 DATA 109,9,173,146,34,63,60,63,62,55,223
9541 DATA 36,36,36,36,45,45,45,45,45,45,54
9544 DATA 54,54,54,54,63,63,63,63,63,63,36
9570 DATA 109,9,173,146,63,63,63,255,35,36,36
9573 DATA 44,45,45,45,45,45,53,54,54,54,62
9576 DATA 63,63,63,63,63,39,4,0
```



Rat Race

Program 3

In this game, you are not the average rat, but a genetically superior and sophisticated rat. You have been placed in a box along with four other rats for a scientific experiment. The other rats are not particularly smart, and they have somehow gotten the notion that *you* are the cheese. So they will chase you and try to eat you.

The game begins when the experimenter drops a piece of yellow cheese into the box at a random place. Your goal is to eat this cheese without being consumed by the other rats. Use the joystick on game controller 1 to move around the box.

Hint: The other rats are so ignorant that they will not pursue anything that is out of their sight. You can fool them by hiding behind the barriers in the box while you are stalking the cheese. You win when you have successfully devoured four pieces of cheese.

Technical Description

The cheese and rats are plotted on the low-resolution screen. The RND function is used in lines 1600-1610 to pick random values for the variables tx and ty. These variables contain the screen coordinates where the cheese is to be plotted, with tx as the horizontal and ty as the vertical coordinates. Line 1620 finds the color of the space where the cheese is about to be drawn. If the space is black (and therefore equal to 0), the space is occupied, and another place must be found in which to drop the cheese.

The variable ce is the number of pieces of cheese that you have consumed. You can change the number of pieces of cheese you must eat in order to win by modifying line 160 IF ce=*number* THEN 200, where *number* reflects the number of pieces to be eaten.

The color of the rats, the barriers, and the cheese are contained in the variables rc, bc, and cc respectively. You can change the values of these variables in line 1145, and they will be drawn in different colors.

Important Variables

Variable	Function
bc	Color of the barriers.
cc	Color of the cheese.
ce	Number of pieces of cheese player rat has consumed.
n	One of the four rats that is being plotted.
pc	Color of the player rat.
rc	Color of the vicious rats.
s	Color of the space that the player is about to move.
tx,ty	Screen coordinates of the cheese. The variable tx is the horizontal coordinate and ty is the vertical coordinate.
x,y	Screen coordinates of the player rat. The variable x is the horizontal coordinate and y is the vertical coordinate.
x(*number*), y(*number*)	Arrays of the screen coordinates of the four vicious rats. The array x corresponds to the horizontal coordinate and the array y to the vertical coordinate. *Number*, the size of both arrays, equals the number of rats, which is 4.

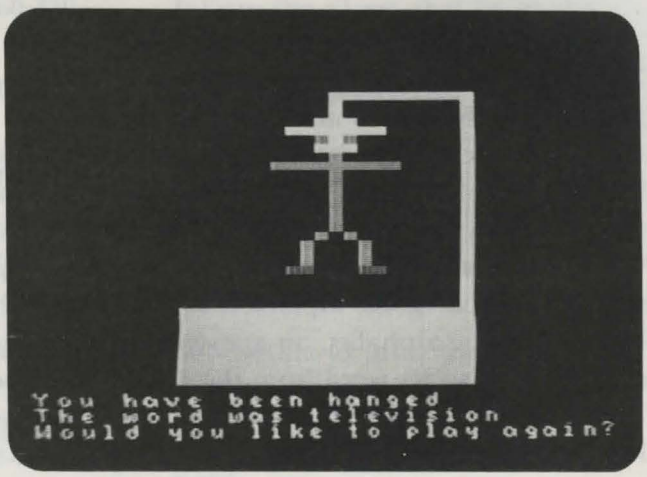
Program Description

Lines	Function
1-95	Standard introduction.
100-280	Main program.
1000-1070	Subroutine to enter low-resolution graphics mode and draw a border around the screen.
1100-1150	Subroutine to set the rats' starting positions.
1200-1260	Subroutine to move the rats toward the player.
1300-1340	Subroutine to draw the four hostile rats.
1400-1440	Subroutine to put player's rat at the beginning point.
1500-1590	Subroutine to check the joystick and allow the player to move.
1600-1640	Subroutine to drop the cheese at a random place in the box.
1700-1720	Subroutine to print the score.
1800-1860	Subroutine to inform the player that he or she has lost.

```
  1 REM  (c) 1983 by Brian Sawyer
  5 TEXT
 10 pm$ = "Rat Race"
 15 FOR n = 1 TO 4
 20 PRINT CHR$(162)
 25 NEXT n
 30 tb = 16-INT(LEN(pm$)/2)
 35 PRINT TAB(tb); pm$
 40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
 45 PRINT " Hit Controller Button to Play"; CHR$(160)
 50 iv = 0: r = 0
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
    AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 GOSUB 1100
110 GOSUB 1000
120 GOSUB 1300
125 GOSUB 1700
130 GOSUB 1400
```

```
140 GOSUB 1200
145 IF s = pc THEN 1800
146 IF tx = 0 THEN  GOSUB 1600
150 GOSUB 1500
155 GOSUB 1500
160 IF ce = 4 THEN 200
170 GOTO 140
200 HOME
210 FOR dl = 1 TO 1000: NEXT dl
220 PRINT "    You Ate All The Cheese! "
230 PRINT "       *** You Win! ***"
240 PRINT "Would you like to play again? ";
250 GET a$
260 IF a$ = "y" OR a$ = "Y" THEN 92
270 TEXT
280 END
1000 GR
1010 COLOR  = bc
1020 VLIN 0, 39 AT 0
1025 HLIN 0, 39 AT 39
1030 VLIN 39, 0 AT 39
1040 HLIN 39, 0 AT 0
1045 PLOT 39, 39
1050 FOR j = 5 TO 35 STEP 8: VLIN 9, 15 AT j: NEXT j
1056 FOR j = 5 TO 35 STEP 8: VLIN 27, 33 AT j: NEXT j
1070 RETURN
1100 DIM x(5), y(5)
1110 FOR n = 1 TO 5
1120 x(n) = (n-1)*4+20
1130 y(n) = 5
1140 NEXT n
1142 ce = 0
1145 pc = 11: rc = 10: bc = 2: cc = 13
1150 RETURN
1200 FOR n = 1 TO 4
1210 xc = 1: yc = 1
1220 IF x(n)-x > 0 THEN  xc = -1
1225 IF x(n) = x THEN  xc = 0
1230 IF y(n)-y > 0 THEN  yc = -1
1240 s = SCRN(x(n)+xc, y(n)+yc)
1242 IF s > 0 THEN 1250
1245 COLOR  = 0: PLOT x(n), y(n)
1246 x(n) = x(n)+xc: y(n) = y(n)+yc
1247 COLOR  = rc: PLOT x(n), y(n)
1250 IF s = pc THEN  RETURN
1255 NEXT n
1260 RETURN
1300 COLOR  = 1
1310 FOR n = 1 TO 4
1315 COLOR  = rc
1320 PLOT x(n), y(n)
1330 NEXT n
1340 RETURN
```

```
1400 x = 20
1410 y = 30
1430 COLOR  = pc: PLOT x, y
1440 RETURN
1500 pd = PDL(5)
1510 IF pd = 15 THEN  RETURN
1515 xc = 0: yc = 0
1520 IF pd = 1 THEN  yc = -1: GOTO 1550
1525 IF pd = 2 THEN  xc = 1: GOTO 1550
1530 IF pd = 4 THEN  yc = 1: GOTO 1550
1535 IF pd = 8 THEN  xc = -1
1550 s = SCRN(x+xc, y+yc)
1555 IF s = cc THEN  ce = ce+1: GOSUB 1700: GOTO 1560
1556 IF s > 0 THEN  RETURN
1560 COLOR  = 0: PLOT x, y
1570 x = x+xc: y = y+yc
1580 COLOR  = pc: PLOT x, y
1590 RETURN
1600 tx = INT(RND(1)*38+1)
1610 ty = INT(RND(1)*38+1)
1620 IF SCRN(tx, ty) > 0 THEN 1600
1630 COLOR  = cc: PLOT tx, ty
1640 RETURN
1700 HOME
1710 PRINT " Cheese Eaten:"; ce
1720 tx = 0: RETURN
1800 HOME
1810 PRINT " The Rats Got You!!"
1820 PRINT "Would you like to play again? ";
1830 GET a$
1840 IF a$ = "y" OR a$ = "Y" THEN 92
1850 TEXT
1860 END
```



Hangman

Program 4

Here is the familiar word game of Hangman. You try to guess a secret word before a body is hanged from the gallows. The secret word is chosen by the Adam and is of a random length. You try to discover the secret word by guessing each letter in the word. If the letter you guessed is in the secret word, it will be shown on the screen each place it occurs. If the letter you have chosen is not in the secret word, a part will be added to the body displayed on the gallows. There are 12 parts of the body that can be drawn on the gallows: head, torso, two arms, two upper legs, two lower legs, two feet, eyes, and finally a hat. If the entire body is drawn before you have guessed the secret word, you lose the game.

The alphabet is displayed at the bottom of the screen. You select letters for your guesses by using the joystick to position the flashing cursor under the letter you wish to select and then pressing the right controller button. Each letter you select is removed from the list to prevent you from choosing it again.

Technical Description

Hangman uses the low-resolution screen to draw the gallows and the body. The letters the player can choose from are displayed in the text window, which is made up of the four bottom rows of the screen.

The secret word is chosen at random from a list of words contained in lines 9000-9999. These words have been *encoded* so you will not become familiar with them while typing them into the computer or listing the program.

The formula to encode the words is to replace each letter with the letter that precedes it in the alphabet. In the case of the letter *a*, which has no letter before it in the alphabet, an apostrophe (') is used to represent it. After the program picks a secret word from the data, it must be decoded before it can be used. The decoding is performed in lines 1050-1090. Each letter appearing in the word is isolated using the MID$ function. Then the letter's numerical value is found by using the ASC command. One is added to this value, and the value is converted back to a character using the CHR$ command. This character is now the decoded letter from the word. After this is done to all of the letters, the word is decoded and can be used in the game.

You can add your own words to this list, and the program will help you encode them. Just type RUN 7000 and enter your word. Next the program will print your word in encoded form. Now add a line after line 9190 (but before line 9999) that reads *line number* DATA *encoded word* and press RETURN. Your encoded word has been added to the list and can be chosen by the computer during the game.

Important Variables

Variable	Function
a$	Encoded secret word.
b$	Decoded secret word.
i$	Player's current letter choice.
l(*number*)	Array to indicate which letter of the alphabet has been chosen. Each entry, starting with 1, corresponds to a letter in alphanumeric order. Hence, the size of *number* is 26.
mk	Number of body parts added to the gallows.

Program Description

Lines	Function
1-95	Standard introduction.
100-195	Main program; enter low-resolution mode.

200-250	Read the user's guess and see if it is in the secret word.
260-330	Inform the player that he or she has lost.
400-460	Inform the player that he or she has won.
1000-1100	Subroutine to pick the secret word randomly from the list.
1200-1380	Subroutine to allow the player to select a letter using the joystick.
1400-1450	Subroutine to print the alphabet at the bottom of the screen.
2000-2020	Subroutine to print the letter in the space.
4000-4080	Subroutine to draw the gallows.
5000-6230	Subroutine to draw the next body part.
6500-6610	Subroutines that call other subroutines to draw the gallows.
7000-7030	Subroutine to encode the words chosen by the user.
9000-9999	Encoded secret word list.

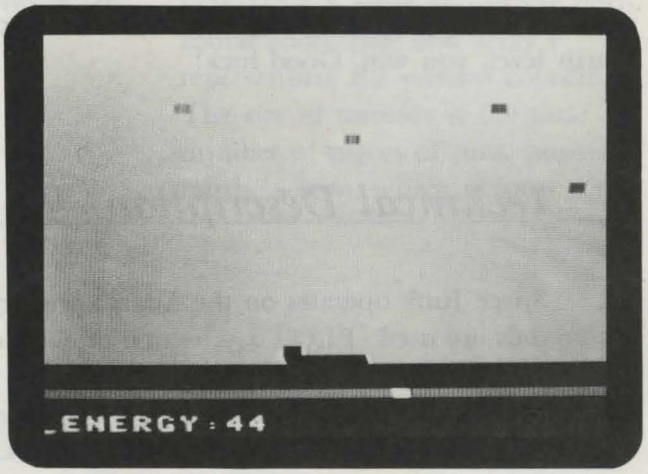
```
  1 REM  (c) 1984 by Brian Sawyer
  5 TEXT
 10 pm$ = "Hangman"
 15 FOR n = 1 TO 4
 20 PRINT CHR$(162)
 25 NEXT n
 30 tb = 16-INT(LEN(pm$)/2)
 35 PRINT TAB(tb); pm$
 40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
 45 PRINT " Hit Controller Button to Play"; CHR$(160)
 50 iv = 0: r = 0
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
    AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 92 NORMAL
 95 CLEAR
100 GOSUB 1000
120 GR
122 GOSUB 4000
```

```
 126 GOSUB 1400
 130 ln = LEN(b$)
 150 tb = 16-ln
 159 VTAB 20
 160 PRINT TAB(tb);
 170 FOR n = 1 TO ln
 180 PRINT "- ";
 190 NEXT n
 195 rt = 0: x = 1
 200 GOSUB 1200
 205 w = 0
 210 FOR n = 1 TO ln
 220 IF MID$(b$, n, 1) = i$ THEN  GOSUB 2000
 230 NEXT n
 240 IF w = 0 THEN  GOSUB 5000
 245 IF rt = LEN(b$) THEN 400
 250 IF mk < 13 THEN 200
 260 HOME
 270 PRINT " You are hung!"
 280 PRINT " The word was "; b$
 290 PRINT "Would you like to play again? ";
 300 GET a$
 310 IF a$ = "y" OR a$ = "Y" THEN 95
 320 TEXT
 330 END
 400 FOR dl = 1 TO 1500: NEXT dl
 405 HOME
 410 PRINT "    You Guessed The Word !!!"
 420 PRINT "Would you like to play again? ";
 430 GET a$
 440 IF a$ = "y" OR a$ = "Y" THEN 95
 450 TEXT
 460 END
1000 FOR nw = 1 TO 100
1002 READ a$: IF a$ = "999" THEN 1004
1003 NEXT nw
1004 i = INT(RND(1)*(nw-1)+1)
1005 DIM l(26): FOR n = 1 TO 26
1006 l(n) = 0: NEXT n
1010 RESTORE
1020 FOR n = 1 TO i
1030 READ a$
1040 NEXT n
1045 b$ = ""
1050 FOR n = 1 TO LEN(a$)
1060 c$ = MID$(a$, n, 1)
1065 IF c$ = " " THEN 1095
1070 ch = ASC(c$)
1080 b$ = b$+CHR$(ch+1)
1090 NEXT n
1095 mk = 1
1100 RETURN
1200 REM      Select letter
```

```
1210 GOSUB 1300
1220 i$ = CHR$(x+96)
1230 RETURN
1300 d$ = CHR$(x+96)
1301 IF l(x) = 1 THEN d$ = " "
1305 HTAB x+2: VTAB 22+nb
1310 PRINT d$; CHR$(163);
1320 FOR d1 = 1 TO 100: NEXT d1
1330 PRINT d$; CHR$(160)
1340 FOR d1 = 1 TO 100 NEXT d1
1350 IF PDL(5) = 2 THEN x = x+1
1360 IF PDL(5) = 8 THEN x = x-1
1365 IF x < 1 THEN x = 1
1366 IF x > 26 THEN x = 26
1370 IF PDL(9) = 0 THEN 1300
1372 IF l(x) = 1 THEN 1300
1375 l(x) = 1
1376 HTAB x+2: PRINT " "; CHR$(160); CHR$(160);
1380 RETURN
1400 HOME
1405 PRINT: PRINT: PRINT " ";
1410 FOR n = 1 TO 26
1420 PRINT CHR$(n+96);
1430 NEXT
1440 PRINT CHR$(160)
1450 RETURN
2000 w = w+1
2005 rt = rt+1
2010 HTAB tb+(n-1)*2
2012 VTAB 20
2015 PRINT i$; CHR$(162)
2017 PRINT CHR$(7);
2020 RETURN
4000 COLOR = 13
4010 VLIN 30, 38 AT 10
4020 HLIN 10, 30 AT 38
4030 VLIN 38, 5 AT 29
4040 HLIN 10, 30 AT 30
4050 HLIN 20, 30 AT 5
4060 VLIN 5, 8 AT 20
4065 FOR y = 30 TO 38
4070 HLIN 10, 30 AT y
4075 NEXT y
4080 IF SCRN(30, 5) > 0 THEN nb = 1
4090 RETURN
5000 ON mk GOSUB 5100, 5200, 5300, 5400, 5500, 5600,
     5700, 5800, 5900, 6000, 6100, 6200
5010 mk = mk+1
5020 RETURN
5100 COLOR = 13
5110 FOR y = 9 TO 11
5120 x1 = 19: x2 = 2: y1 = y: GOSUB 6600
5130 NEXT
```

```
5140 RETURN
5200 COLOR  = 9
5210 y1 = 12: y2 = 7: x1 = 20: GOSUB 6500
5220 RETURN
5300 COLOR  = 9
5310 x1 = 20: x2 = 4: y1 = 13: GOSUB 6600
5320 RETURN
5400 COLOR  = 9
5410 x1 = 16: x2 = 4: y1 = 13: GOSUB 6600
5420 RETURN
5500 COLOR  = 9
5510 FOR t = 0 TO 2
5520 PLOT 20+t, 20+t
5530 NEXT t
5540 RETURN
5600 COLOR  = 9
5610 FOR t = 0 TO 2
5620 PLOT 20-t, 20+t
5630 NEXT t
5640 RETURN
5700 COLOR  = 9
5710 y1 = 22: y2 = 3: x1 = 22: GOSUB 6500
5720 RETURN
5800 COLOR  = 9
5810 y1 = 22: y2 = 3: x1 = 18: GOSUB 6500
5820 RETURN
5900 COLOR  = 4
5910 x1 = 22: x2 = 1: y1 = 25: GOSUB 6600
5920 RETURN
6000 COLOR  = 4
6010 x1 = 17: x2 = 1: y1 = 25: GOSUB 6600
6020 RETURN
6100 COLOR  = 2
6110 PLOT 19, 10: PLOT 21, 10
6120 RETURN
6200 COLOR  = 10
6210 x1 = 19: x2 = 2: y1 = 8: GOSUB 6600
6220 x1 = 17: x2 = 6: y1 = 9: GOSUB 6600
6230 RETURN
6500 VLIN y1, y1+y2+(1-nb) AT y1
6510 RETURN
6600 HLIN x1, x1+x2+(1-nb) AT y1
6610 RETURN
7000 PRINT: INPUT a$
7005 PRINT "  ";
7010 FOR n = 1 TO LEN(a$)
7020 PRINT CHR$(ASC(MID$(a$, n, 1))-1);
7030 NEXT n
7040 END
9000 DATA lnmjdx
9010 DATA b`mnd
9020 DATA   sxodvqhsdq
```

```
 9030 DATA `c`l
 9040 DATA `unb`cn
 9050 DATA roghmw
 9060 DATA uhnkd's
 9070 DATA uhs`lhm
 9080 DATA inxr'shbj
 9090 DATA qnnr'sdq
 9100 DATA g`lr'sdq
 9110 DATA bknbj
 9120 DATA mdogdv
 9130 DATA `artqc
 9140 DATA  sdkduhrhnm
 9150 DATA  od'oodq
 9160 DATA `mbdrsnq
 9170 DATA fdk`'shm
9180 DATA mdb's`q
 9190 DATA rvhmd
9999 DATA 999
```



Space Junk

Program 5

Your home planet is being threatened by debris falling from decaying space stations. You control a laser cannon on the planet's surface that disintegrates the falling junk before it hits the ground. You use the joystick on game controller 1 to move the cannon left or right, and you press the left controller button to fire.

Because the atmosphere of your planet is very dense, it is difficult for the laser to penetrate. You must clear areas of the atmosphere by shooting at it before you can hit the space junk. These clear spaces within the atmosphere make it possible for you to hit junk at a higher altitude. But if a piece of junk encounters such a space in the atmosphere and is not immediately disintegrated, it will fill in the area with the same dense atmosphere.

You start with 50 energy points, which provide the power for your cannon. Each shot absorbed by the atmosphere uses 1 point, and you lose 10 points each time a piece of junk hits the ground. You score 20 points for each piece of junk you hit in the air. If you run out of energy at any time, you lose.

This game has four different levels of play. You advance to the next level after you have successfully defended the planet from 16 pieces of space junk. Hitting the debris becomes more difficult with each level. If you can survive the fourth level, you win. Good luck!

Technical Description

Space Junk operates on the Adam's low-resolution screen. Two important commands are used: PLOT *x,y*, which draws a colored box at location *x,y*, and SCRN(*x,y*), which is used to identify the color of the block at position *x,y*. Here *x* represents the horizontal coordinate and *y* represents the vertical coordinate. Both *x* and *y* are replaced by the appropriate variables when used within the program.

The positions of the four pieces of space junk are contained in two arrays of size 4. One of the arrays, named x, stores the horizontal coordinates of each piece of junk, while the other array, named y, stores the vertical coordinates. The method used to simulate the falling debris operates as follows: each piece of junk is printed on the screen and erased; the vertical coordinate of each piece is increased by 1, and then each piece is printed on the screen again.

The SCRN command is used in line 1500 to identify, by color, whether atmosphere or junk has been shot.

The PDL command is used in lines 1405-1410 to check if the joystick on game controller 1 has been moved. If it has been moved to the right, the value of PDL(5) equals 8. If the joystick is moved left, the value of PDL(5) changes to 2.

The Adam's random number function, RND, is used to pick locations for the space junk to appear at the top of the screen. In line 1605 the RND function is used to position a new piece of junk anywhere from the 4th to the 34th column.

Important Variables

Variable	Function
bk	Screen background color.
sc	Player's current number of energy points.
sr	Level of play.
sz	Number of pieces of junk.
x,y	Position of the laser cannon on the screen. The variable x represents the horizontal coordinate. (Since the cannon never moves vertically, y is a constant value.)

x(*number*), y(*number*)	Current position of a piece of junk with array x representing the horizontal coordinate and array y representing the vertical coordinate. The size of *number* is the total number of pieces of junk possible on the screen, which is four.

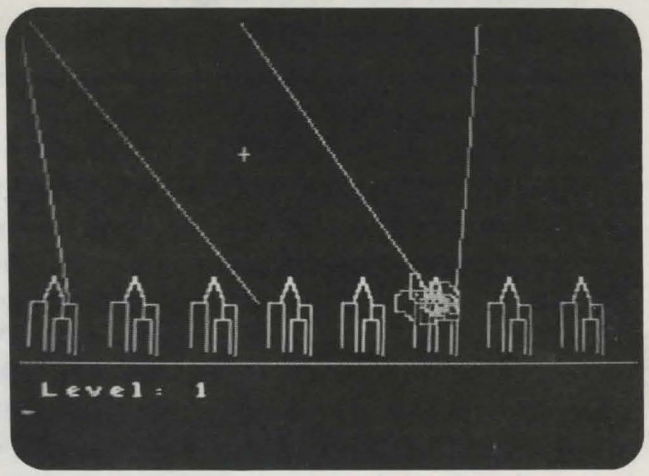
Program Description

Lines	Function
1-95	Standard introduction.
100-150	Main program.
1000-1020	Subroutine to enter low-resolution graphics mode.
1100-1180	Subroutine to initialize the variables and draw the background on the screen.
1200-1230	Subroutine to initialize the position of each piece of junk.
1300-1320	Subroutine to move the space junk down one row.
1400-1440	Subroutine to check the joystick position and button and fire or move the cannon.
1500-1530	Subroutine to check what has been hit by the cannon.
1600-1610	Subroutine to create a new piece of junk after one has been shot.
1700-1720	Subroutine to print the energy score.
1800-1810	Subroutine to print the energy level number.
1900-1950	Subroutine to inform you when you lose.
2000-2040	Subroutine to advance to the next screen of play.
3000-3010	Subroutine to inform you when you win.

```
  1 REM  (c) 1984 by Brian Sawyer
  5 TEXT
 10 pm$ = "Space Junk"
 15 FOR n = 1 TO 4
 20 PRINT CHR$(162)
 25 NEXT n
 30 tb = 16-INT(LEN(pm$)/2)
 35 PRINT TAB(tb); pm$
 40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
 45 PRINT " Hit Controller Button to Play"; CHR$(160)
 50 iv = 0: r = 0
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
    AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 GOSUB 1000
110 GOSUB 1100
120 GOSUB 1200
125 GOSUB 1700
130 GOSUB 1300
135 IF sc < 0 THEN 1900
137 IF i > 0 THEN  GOSUB 1800
139 IF nd > 15 THEN 2000
140 GOSUB 1400
145 GOSUB 1400
150 GOTO 130
1000 GR
1010 COLOR  = 1
1020 RETURN
1100 sz = 4
1105 DIM x(sz), y(sz)
1110 nd = 0
1114 sr = 1
1115 DIM sc(5): sc(1) = 13: sc(2) = 14: sc(3) = 11: sc(4) = 15
1120 x = 20: y = 39
1130 bk = 13
1131 bc = 9
1132 sc = 50
1135 COLOR  = bk
1140 FOR n = 0 TO 39
1150 VLIN 0, 36 AT n
1160 NEXT n
1165 COLOR  = bc
1166 HLIN 0, 39 AT 39
1167 PLOT 39, 39
1170 x = 20: y = 39
1180 RETURN
```

```
1200 FOR n = 1 TO sz
1210 x(n) = INT(RND(1)*30+5)
1220 y(n) = INT(RND(1)*10+1)
1230 NEXT n: RETURN
1300 i = 0
1302 FOR n = 1 TO sz
1305 COLOR = bk: PLOT x(n), y(n)
1310 y(n) = y(n)+1: IF y(n) > 37 THEN i = n: GOSUB 1600:
     sc = sc-10
1315 COLOR = n: PLOT x(n), y(n)
1320 NEXT n: RETURN
1400 xc = 0: COLOR = 4: IF PDL(9) = 0 THEN 1405
1401 FOR m = 38 TO 1 STEP -1: IF SCRN(x, m) <> 0 THEN
     GOSUB 1500: GOTO 1404
1403 PLOT x, m: NEXT m
1404 COLOR = 0: VLIN 39, m AT x
1405 IF PDL(5) = 8 AND x > 2 THEN xc = -1
1410 IF PDL(5) = 2 AND x < 37 THEN xc = 1
1420 COLOR = bc: PLOT x, y
1422 x = x+xc
1425 IF SCRN(x, y) <> 0 THEN a = a
1430 COLOR = 15: PLOT x, y
1440 RETURN
1500 s = SCRN(x, m)
1505 IF s = bk THEN sc = sc-1: GOSUB 1800: RETURN
1510 i = s: GOSUB 1600
1520 sc = sc+10: GOSUB 1800
1530 RETURN
1600 nd = nd+1
1605 x(i) = INT(RND(1)*30+5)
1610 y(i) = 0: RETURN
1700 HOME
1705 PRINT
1710 PRINT " ENERGY:"; sc; CHR$(160)
1720 RETURN
1800 IF sc <= 0 THEN RETURN
1810 HTAB 9: PRINT sc; "  "; CHR$(160): RETURN
1900 sc = 0
1910 HOME
1930 PRINT "          OUT OF ENERGY!"
1940 PRINT "       *** You Lose ***"
1950 END
2000 HOME
2001 IF sr = 4 THEN 3000
2002 sr = sr+1
2005 sc = 50
2006 bk = sc(sr)
2015 GR: COLOR = 1
2020 GOSUB 1135
2025 GOSUB 1200
2030 GOSUB 1700
2035 nd = 0
```

```
2040 GOTO 130
3000 FOR dl = 1 TO 1000: NEXT
3001 PRINT
3005 PRINT "          *** You Won!! ***"
3010 END
```



Missile Monitor

Program 6

A city is being besieged by missiles, and you must destroy them before they strike the buildings of the city. You are in control of a defense base for shooting down the missiles. Using the joystick on game controller 1, you can move the *indicator* (a small orange cross) on the screen. When the indicator is near the head of an incoming missile, press the right controller button to cause an explosion. If the head of the missile is within the cloud of the explosion, it will be destroyed. As some of the missiles descend, they split into two, three, or sometimes four missiles, so it is to your advantage to destroy the missiles as soon as possible.

If you have successfully destroyed 25 missiles, you will advance to the next level. There are five levels, and each successive level is more difficult because the missiles move faster. If the city is struck by more than two missiles at any level, you lose. If you have survived after the fifth level, you win.

Technical Description

This program is a good example of the use of shape tables in an action game. Three shape tables are used: one for the indicator, one for the explosion cloud, and one for the city.

A shape table is used to draw the buildings at the bottom of the screen. The DRAW 3 AT n,150 in line 4010 will print a single building. When the DRAW 3 AT n,150 command is executed five times, the variable n is increased by 30, and five equally spaced buildings are drawn across the screen. Notice that the vertical coordinate is always constant at 150. The explosion is accomplished by using the DRAW 2 AT x,y command, followed by XDRAW 2 AT x,y to erase it.

The value of PDL(5) is checked to see if the joystick is being moved. First the variable a in line 1400 is given the value of PDL(5). If this variable is found to contain the value 1, the indicator is moved up. A value of 2 causes the indicator to be moved to the right, 4 causes it to be moved down, and 8 causes it to be moved to the left. The testing of variable a occurs from line 1410 to 1440.

Important Variables

Variable	Function
a	Contains the most recent value of PDL(5). This is the position of the joystick on game controller 1.
c(*number*)	Array containing the value that will be added to the horizontal coordinate of a missile during its descent. The size of *number* is 4.
d(*number*)	Array containing the value that will be added to the vertical coordinate of a missile during its descent. The size of *number* is 4.
gm	Level currently being played.
ht	Number of missiles that have hit the city.
nm	Number of missiles destroyed by the player.
x,y	Screen coordinates of the indicator, where x is the horizontal coordinate and y is the vertical coordinate.
x(*number*), y(*number*)	Arrays containing the screen coordinates of the missiles. The array x is for the horizontal coordinate and the array y is for the vertical coordinate. The size of *number* is 4.

xc,yc	Values added to the variables x and y in order to move the indicator. The variable xc indicates direction on the horizontal axis and the variable yc indicates direction on the vertical axis.

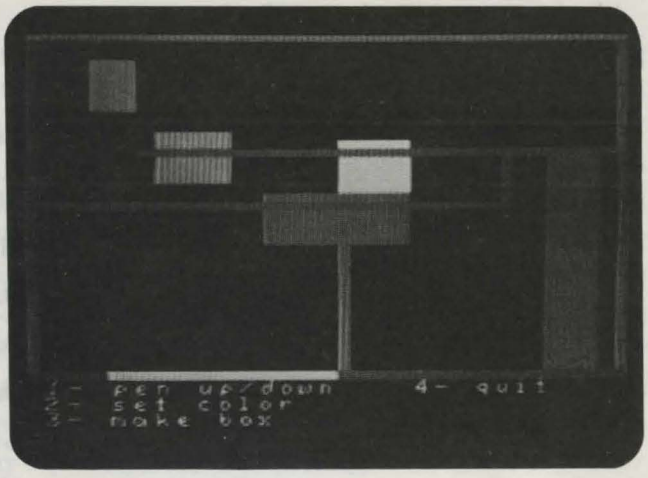
Program Description

Lines	Function
0	Shape table memory pointer.
1-95	Standard introduction.
100-230	Main program.
300-320	Inform you that you have won.
400-460	Inform you that you have lost.
1000-1099	Subroutine to enter high-resolution graphics mode.
1100-1199	Subroutine to initialize the arrays and set colors.
1200-1240	Subroutine to initialize the missile positions.
1300-1340	Subroutine to move and draw the missiles.
1400-1470	Subroutine to allow the joystick to move the indicator.
1500-1570	Subroutine to make the explosion and check to see if any missiles are affected.
1600-1660	Subroutine to create a new missile at a random position at the top of the screen.
2500-2520	Subroutine to make the missile explode on the ground.
4000-4040	Subroutine to draw the cities at the bottom of the screen.
8000-8130	Subroutine to load shape table data into memory.
9510-9830	Shape table data.

```
  0 HIMEM :41000
  1 REM (c) 1984 by Brian Sawyer
  5 TEXT
 10 pm$ = "Missile Monitor"
 15 FOR n = 1 TO 4
 20 PRINT CHR$(162)
 25 NEXT n
 30 tb = 16-INT(LEN(pm$)/2)
 35 PRINT TAB(tb); pm$
 40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
 41 NORMAL
 42 d = d+1: IF d/2 = INT(d/2) THEN  INVERSE
 45 PRINT " Hit Controller Button to Play"; CHR$(160)
 50 iv = 0: r = 0
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
    AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 GOSUB 8000
150 GOSUB 1000
160 GOSUB 1100
170 GOSUB 1200
171 HOME: PRINT: PRINT " Level: "; gm+1
172 GOSUB 4000
173 ht = 0: nm = 0
175 GOSUB 1400
177 IF PDL(9) THEN  GOSUB 1500
180 GOSUB 1300: IF ht < 3 AND nm < 25 THEN 175
190 IF ht > 2 THEN 400
200 gm = gm+1: IF gm > 4 THEN 300
205 HGR
210 SCALE  = 1
220 ROT  = 0
230 GOTO 170
300 HOME
310 PRINT " ****  You Win!  ****"
320 PRINT "Would you like to play again? ";
330 GET a$
340 IF a$ = "y" OR a$ = "Y" THEN 95
350 TEXT
360 END
400 HOME
410 PRINT " ****  You Lose!  ****"
420 PRINT "Would you like to play again? ";
430 GET a$
440 IF a$ = "y" OR a$ = "Y" THEN 95
450 TEXT
```

```
 460 END
1000 HGR
1005 SCALE  = 1
1006 ROT  = 0
1010 HCOLOR  = 1
1099 RETURN
1100 DIM x(4), c(4), d(4)
1105 DIM cl(4): cl(0) = 8: cl(1) = 15: cl(2) = 5:
     cl(3) = 6: cl(4) = 1
1110 x = 128: y = 100
1199 RETURN
1200 FOR n = 1 TO 4
1220 x(n) = INT(RND(10)*256)
1225 y(n) = 1
1230 c(n) = RND(1)
1235 IF x(n) > 128 THEN  c(n) = -c(n)
1237 d(n) = 1: IF n < gm THEN  d(n) = 2
1240 NEXT n: RETURN
1300 HCOLOR  = 1
1305 FOR n = 1 TO 4
1310 HPLOT x(n), y(n)
1320 x(n) = x(n)+c(n): y(n) = y(n)+d(n)
1325 IF y(n) > 130 THEN  GOSUB 2500
1335 NEXT n
1340 RETURN
1400 a = PDL(5)
1405 xc = 0: yc = 0
1410 IF a = 1 THEN  yc = -8
1420 IF a = 2 THEN  xc = 8
1430 IF a = 4 THEN  yc = 8
1440 IF a = 8 THEN  xc = -8
1442 IF (x+xc > 250) OR (x+xc < 5) OR (y+yc > 108)
     OR (y+yc < 5) THEN  RETURN
1447 XDRAW 1 AT x, y
1450 y = y+yc: x = x+xc
1460 DRAW 1 AT x, y
1470 RETURN
1500 HCOLOR  = 2
1515 DRAW 2 AT x+5, y
1520 FOR n = 1 TO 4
1530 IF ABS(x(n)-x)+ABS(y(n)-y) < 10 THEN  GOSUB 1600
1540 NEXT n
1550 HCOLOR  = 0
1565 XDRAW 2 AT x+5, y
1570 RETURN
1600 nm = nm+1
1605 IF RND(1) > .5 THEN 1630
1610 y(n) = 1: x(n) = INT(RND(1)*255+1): c(n) = RND(1)
1620 IF x(n) > 128 THEN  c(n) = -c(n)
1625 RETURN
1630 IF n = 1 THEN 1610
1640 x(n) = x(n-1): y(n) = y(n-1): c(n) = RND(1)
1650 IF x(n) > 128 THEN  c(n) = -c(n)
```

```
1660 RETURN
2500 HCOLOR  = 15
2501 ht = ht+1: IF x(n) > 255 THEN  x(n) = 255
2502 DRAW 4 AT x(n), y(n)
2503 XDRAW 4 AT x(n), y(n)
2504 DRAW 2 AT x(n), y(n)
2505 HCOLOR  = 5
2506 SCALE  = 2: DRAW 2 AT x(n), y(n)
2510 XDRAW 2 AT x(n), y(n)
2515 SCALE  = 1: XDRAW 2 AT x(n), y(n)
2520 GOSUB 1600: RETURN
4000 HCOLOR  = cl(gm)
4005 FOR n = 10 TO 230 STEP 30
4010 DRAW 3 AT n, 150
4020 NEXT n
4025 HCOLOR  = 6
4030 HPLOT 0, 158 TO 255, 158
4040 RETURN
8000 qs = 4: pt = 16766: tl = 43000
8005 of = qs*2+3
8010 by = tl: GOSUB 8100
8015 POKE pt, lo: POKE pt+1, hi
8020 POKE tl, qs
8030 FOR n = 1 TO qs
8040 by = of: GOSUB 8100
8050 POKE tl+n*2, lo: POKE tl+n*2+1, hi
8060 READ a: POKE tl+of, a: of = of+1
8065 IF a <> 0 THEN 8060
8070 NEXT n
8080 RETURN
8100 hi = INT(by/256): lo = by-hi*256
8130 RETURN
9510 DATA  45,45,221,147,34,36,36,36,0
9610 DATA  60,60,60,60,55,55,55,46,46,46,37
9620 DATA  37,60,60,60,39,63,55,46,46,54,54
9630 DATA  46,45,36,36,45,54,46,45,36,36,36
9640 DATA  60,60,60,63,55,54,39,60,36,60,55
9650 DATA   54,46,54,54,54,62,60,60,39,36,44
9660 DATA  44,45,46,0
9710 DATA  36,36,36,36,36,36,36,36,36,36,44
9720 DATA 45,45,53,54,54,54,54,54,54,54,54
9730 DATA 54,54,54,77,36,36,36,36,36,36,36
9740 DATA 44,45,45,53,54,54,54,54,54,54,54
9750 DATA 118,33,36,36,36,36,36,36,36,36,36
9760 DATA 36,36,60,63,63,54,54,54,36,36,36
9770 DATA 36,36,39,36,39,36,39,36,55,54,55
9780 DATA  54,55,54,55,54,0
9810 DATA 39,39,127,73,33,36,39,103,137,146,41
9820 DATA 44,45,172,210,27,54,45,46,222,219,46
9830 DATA 46,238,219,39,36,4,0
```



Junior Artist

Program 7

With this program you can draw simple color pictures using the screen as your canvas and a special *pen* as your paintbrush. The commands listed below this canvas allow you to draw many different shapes in a variety of colors.

The screen is divided into an invisible 40 × 40 grid and is surrounded by a white border. Each position in the grid can contain one of 16 possible colors. The position of the special pen on the screen is indicated by a flashing cursor. This pen can be moved around the screen by using the joystick on game controller 1. The pen has two positions: up and down. If the pen is down, a path of color will be painted when the pen is moved. If the pen is up, it can be moved without drawing anything. Pressing 1 on the controller keypad will lift the pen up if it is down or put it down if it is up.

You can change the color of the pen by pressing 2 on the keypad. You will then be asked which color you want. Press the number of the color on the keypad and enter it by pressing the right controller button. If you make a mistake while

entering the number, use the left controller button as a backspace. The colors and their corresponding numbers are as follows:

Numbers	Color
0	Black
1	Magenta
2	Dark blue
3	Dark red
4	Dark green
5	Grey 1
6	Medium green
7	Light blue
8	Light yellow
9	Medium red
10	Grey 2
11	Light red
12	Light green
13	Light yellow
14	Cyan
15	White

You can make your pen erase any part of the drawing by simply setting the color to black (0).

To draw the boxes on the screen, press 3 on the keypad. Enter the width of the box by pressing a number on the keypad and then the right controller button. Follow the same procedure for the height. If you make a mistake while entering either number, use the left controller button as a backspace. Both the height and width can range from 1 to 39 squares. The box will be printed on the screen with the pen in the upper-left corner and will be filled with the current pen color. If you request a size larger than the canvas is capable of displaying, the box will be trimmed so that it can be shown on the screen. The box can be drawn with the pen either up or down.

You quit the program by selecting 0 from the menu. You will then be asked if you really want to quit. Press Y on the keypad to confirm your selection or N if you decide not to quit.

When the session begins, the pen is located in the center of the canvas, the color is white, and the pen is up.

Technical Description

In this program the pen's path is drawn on the low-resolution screen using the PLOT command. The joystick on game controller 1 is used to move the pen. The value of PDL(5) changes with the position of the joystick. Its value is 1 for the upward direction, 4 for the downward direction, 2 for right, and 8 for left. In lines 1202-1222 the position of the pen is changed according to the value of PDL(5). The SCRN command is used in line 1225 to find the color of the square that the cursor is about to be moved upon. In the pen up mode, this color must be remembered before the cursor occupies the square so that the original color can be restored after the cursor has left the square. This ensures that moving the cursor while in the up mode will not alter the drawing. The cursor is not a problem when the pen is in the down mode, since the cursor is the same color as the pen and the pen will paint over any square that it occupies.

Important Variables

Variable	Function
cl	Current pen color. Value of cl ranges from 0 to 15.
d	Color of the cursor.
fl	Indicates whether the cursor is on or off.
no	Number entered in the keypad.
p1	Contains the current value of PDL(5), which is the position of the joystick on game controller 1.
p2	Contains the current value of PDL(7), the position of the left controller button.
p3	Contains the current value of PDL(9), the position of the right controller button.
p4	Contains the current value of PDL(13), the number pressed on the keypad.
x,y	Screen coordinates of the pen. The variable x represents the horizontal coordinate, and the variable y represents the vertical coordinate.
xl	Length of the box to be drawn.
yl	Height of the box to be drawn.

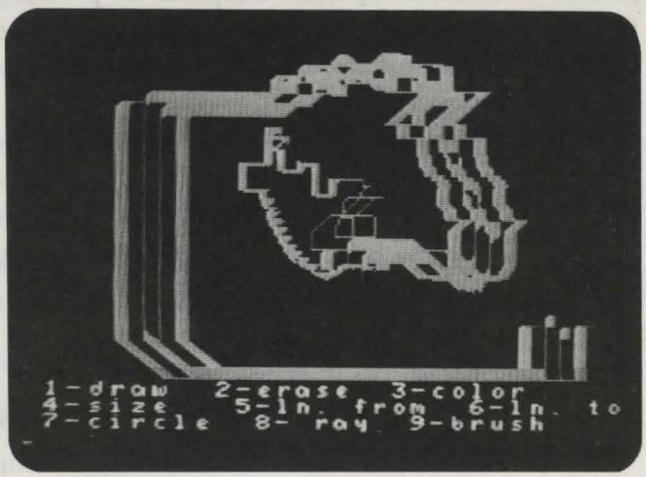
Program Description

Lines	Function
1-95	Standard introduction.
100-170	Main program.
1000-1070	Subroutine to enter low-resolution graphics mode, initialize variables, and draw the screen border.
1100-1140	Subroutine to print the menu.
1200-1230	Subroutine to move the pen, leaving a mark if the pen is in the down mode.
1300-1350	Subroutine to change the pen color.
1400-1460	Subroutine to check if the player has selected any menu options on the keypad.
1500-1540	Subroutine to change the pen between the up mode and the down mode.
1600-1620	Subroutine to flash the cursor.
1700-1799	Subroutine to draw the box.

```
  1 REM  (c) 1984 by Brian Sawyer
  5 TEXT
 10 pm$ = "Junior Artist"
 15 FOR n = 1 TO 4
 20 PRINT CHR$(162)
 25 NEXT n
 30 tb = 16-INT(LEN(pm$)/2)
 35 PRINT TAB(tb); pm$
 40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
 45 PRINT " Hit Controller Button to Play"; CHR$(160)
 50 iv = 0: r = 0
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
    AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 GOSUB 1000
105 GOSUB 1100
110 p1 = PDL(5)
```

```
 115 f = f+1: IF f/9 = INT(f/9) THEN  GOSUB 1600
 120 p2 = PDL(13)
 130 IF p1 = 0 AND p2 = 15 THEN 110
 140 IF p1 > 0 THEN  GOSUB 1200
 150 IF p2 = 1 THEN  GOSUB 1500
 160 IF p2 = 2 THEN  GOSUB 1300
 165 IF p2 = 3 THEN  GOSUB 1700
 167 IF p2 = 4 THEN  TEXT: END
 170 GOTO 110
1000 GR
1005 DIM i(10)
1010 cl = 15
1015 COLOR  = cl
1020 x = 20: y = 20
1030 VLIN 0, 39 AT 0
1035 HLIN 0, 39 AT 39
1040 VLIN 39, 0 AT 39
1050 HLIN 39, 0 AT 0
1060 PLOT 39, 39
1070 RETURN
1100 HOME
1110 PRINT " 1- pen up/down    4- quit"
1120 PRINT " 2- set color"
1130 PRINT " 3- make box"
1140 RETURN
1200 xc = 0: yc = 0
1202 IF p1 = 1 THEN  yc = -1
1205 IF p1 = 2 THEN  xc = 1
1210 IF p1 = 4 THEN  yc = 1
1215 IF p1 = 8 THEN  xc = -1
1217 IF x+xc > 39 OR x+xc < 0 THEN  RETURN
1218 IF y+yc > 39 OR y+yc < 0 THEN  RETURN
1219 COLOR  = cl
1220 IF pn = 0 THEN  COLOR  = s
1221 PLOT x, y
1222 x = x+xc: y = y+yc
1225 s = SCRN(x, y)
1226 COLOR  = cl
1230 PLOT x, y: RETURN
1300 HOME
1310 PRINT " which color? (0-15) ";
1315 FOR t = 1 TO 500: NEXT t
1320 GOSUB 1400
1330 IF no > 15 THEN 1300
1340 cl = no
1345 GOSUB 1100
1350 RETURN
1400 REM  test joystick
1410 k = 1
1415 pd = PDL(13)
1417 p2 = PDL(9)
1420 p1 = PDL(7)
```

```
1423 IF p2 = 1 AND k > 1 THEN 1450
1425 IF p1 = 1 AND k > 1 THEN  k = k-1: PRINT CHR$(163); " ";
     CHR$(163);
1426 IF p1 = 1 AND PDL(7) = 1 THEN 1426
1430 IF pd > 9 OR k > 9 THEN 1415
1435 PRINT pd;
1440 i(k) = pd
1442 k = k+1
1443 IF PDL(13) <> 15 THEN 1443
1445 GOTO 1415
1450 no = 0: q = 0: FOR k1 = k-1 TO 1 STEP -1
1455 no = no+i(k1)*10^q: q = q+1
1460 NEXT k1: PRINT: RETURN
1500 REM  pen up or down
1510 pn = 1-pn
1520 HOME
1525 IF pn = 0 THEN  PRINT " pen up"
1526 IF pn = 1 THEN  PRINT " pen down"
1530 IF PDL(13) <> 15 THEN 1530
1535 FOR t = 1 TO 500: NEXT t
1536 GOSUB 1100
1540 RETURN
1600 fl = 1-fl
1603 d = cl
1604 IF cl = s THEN  d = 0: IF cl = 0 THEN  d = 15
1610 IF fl = 1 THEN  d = s
1615 COLOR  = d: PLOT x, y
1620 RETURN
1700 HOME
1720 FOR t = 1 TO 500: NEXT t
1725 PRINT " width of box ? (1-39) ";
1730 GOSUB 1400
1735 IF no > 39 THEN 1710
1740 IF no = 0 THEN 1796
1745 xl = x+no
1746 IF xl > 39 THEN  xl = 39
1747 HOME
1750 PRINT "height of box ? (1-39) ";
1755 GOSUB 1400
1760 IF no > 39 THEN 1747
1765 IF no = 0 THEN 1796
1770 yl = y+no
1775 IF yl > 39 THEN  yl = 39
1776 COLOR  = cl
1777 s = cl
1780 FOR nx = x TO xl-1
1790 VLIN y, yl AT nx
1795 NEXT nx
1796 GOSUB 1100
1799 RETURN
```



Senior Artist

Program 8

This program helps you draw detailed and colorful pictures using the screen as your canvas. This time the controller is your paintbrush. The joystick on game controller 1 is used to control the direction of your brush, and the keypad is used to select different painting commands.

The screen is divided into two parts. The largest portion is used for the canvas. It occupies the upper portion of the screen and contains 256 tiny points called *pixels* (picture elements) in the horizontal direction and 156 in the vertical direction. In the lower part of the screen you will see a menu of the painting commands. These commands allow you to perform a variety of painting functions.

You begin with your brush in the center of the screen, which is indicated by a flashing dot. If you move the brush on the screen using the joystick, a thin trail of white paint will follow.

You can use the painting commands to change colors and the size or shape of the brush or to draw lines and circles. To select a command from the menu, press its number on the keypad. When you draw a circle or change colors or size

of the brush, you will be asked to input an additional number to determine how the command will operate. Respond by pressing the number on the keypad. If you make a mistake, use the left controller button as a backspace key and then type the correct number. Finally, enter the number into the computer by pressing the right button.

Here is a description of the painting commands:

Command	Function
0-QUIT	Lets you quit the painting session. You will be asked if you really want to quit. Press Y to confirm your selection of this command or N if you decide not to quit.
1-DRAW	Returns you from erase mode to draw mode. Moving the brush will leave a trail of paint.
2-ERASE	Causes your brush to erase any painted area that you move it over.
3-COLOR	Allows you to select the color of paint. You are asked to enter a number from 1 to 15 corresponding to the chosen color. Here are the colors you can choose from:

Number	Color
0	Black
1	Green
2	Violet
3	White 1
4	Black 2
5	Orange
6	Blue
7	White 2
8	Brown
9	Dark blue
10	Grey
11	Pink
12	Dark green
13	Yellow
14	Aqua
15	Magenta

4-SIZE	Changes the size of the brush. Enter a number from 1 to 5, with 1 being the smallest (about the size of a pencil tip) and 5 being the largest (about the size of a quarter). This command does not alter the style of the brush; it simply enlarges or shrinks it.
5-LN FROM	Defines the present location of your brush as the *beginning point* for a line to be drawn. The line will actually be drawn when you define the *endpoint* of the line in the LN TO or the RAY command.
6-LN TO	Draws a line from the beginning point (defined in the LN FROM command) to the present location or endpoint of the brush. Whenever this command is executed, the endpoint becomes the beginning point. If you use this command without setting a new beginning point, a line will be drawn from the endpoint of the previously drawn line. If no lines have been drawn, LN TO will draw a line from the center of the screen to the current location of the brush.
7-CIRCLE	Draws a circle that is centered on the present location of the brush. You are asked to enter the radius of the circle. This can range from 1 (smallest) to 80 (largest). A circle of radius 80 takes up about half of the canvas. The inside of the circle is not filled with paint.
8-RAY	Draws a line from the beginning point (defined in the LN FROM command) to the present location of the brush. This command differs from the LN TO command in that it does not change the beginning point after the line has been drawn.

Thus, you only need to define the beginning point once with the LN TO command and then move the brush around the canvas while executing the RAY command. Each RAY command will draw a line from the present location of the brush to the original beginning point. This can be used to make flower petals, sun rays, or any design that has many lines originating from a single point.

9-BRUSH — Allows you to select the style of brush to paint with. There are five styles of brush, each of which leaves a slightly different trail of paint.

Technical Description

The difference between Senior Artist and Junior Artist is that Senior Artist is done in high-resolution graphics and Junior Artist is done in low-resolution graphics. High-resolution graphics allows Senior Artist to use a variety of painting commands like circles and lines that could not be performed satisfactorily on Junior Artist. You should compare the differences between the programs.

All of the painting in this program is done by using shape tables (which, incidentally, would not be possible in low-resolution graphics). The brush is a shape table, and each of the different painting commands will cause this shape to move or change color. In line 2325 the size of the brush is changed with the SCALE command. The color of the brush is defined in the command HCOLOR. In the erase mode, the XDRAW command is used to clear the area under the brush.

As the brush moves from point to point, it fills each point with paint, thereby leaving a continuous trail. This method of painting can be changed by modifying the movement of the brush so that it leaves a broken trail. For example, changing the value assigned to the variable sp in line 1014 to 2 will paint every other point.

An interesting modification can be made to the program by removing the REM statement from line 1276. This will cause the brush to rotate constantly, creating a spiral pattern. This effect is produced by using the ROT command and incrementing the value of the variable rt each time it is executed.

Important Variables

Variable	Function
cl	Current color of the brush.
fx,fy	Screen coordinates of the beginning point for a line to be drawn. The variable fx represents the horizontal coordinate and fy the vertical coordinate.
pd	Contains value of PDL(5), the position of the joystick on game controller 1.
pe	Indicates which pen is being used.
pn	Indicates erase mode.
r	Radius of the circle.
sp	Size of step that the brush moves.
sx,sy	Current point being plotted on the circle, where the variable sx represents the horizontal coordinate and the variable sy represents the vertical coordinate.
sz	Current size of the brush.
x,y	Current screen coordinates of the brush, where variable x represents the horizontal coordinate and variable y the vertical coordinate.
xc,yc	Values added on the x and y to move the indicator.

Program Description

Lines	Function
0	Shape table memory pointer.
1-95	Standard introduction.
100-199	Main program.
500-550	Message to ask you whether you would like to continue to play.
1000-1050	Subroutine to enter high-resolution graphics mode.
1200-1290	Subroutine to move the brush.

1300-1390	Subroutine to execute the painting command that has been selected.
1400-1470	Subroutine to allow the player to enter a number from the controller keypad.
1500-1520	Subroutine to flash the brush position.
2000-2010	Subroutine to exit from the erase mode.
2100-2110	Subroutine to enter the erase mode.
2200-2230	Subroutine to choose a new color of paint for the brush.
2300-2330	Subroutine to choose a new size of brush.
2400-2410	Subroutine to set the beginning position for a line to be drawn.
2500-2545	Subroutine to draw a line to the brush.
2600-2690	Subroutine to draw a circle.
2800-2830	Subroutine to choose a new style of brush.
2900-2940	Subroutine to print the menu in the text window.
8000-8130	Subroutine to load the shape table data into memory.
9000-9503	Shape table data.

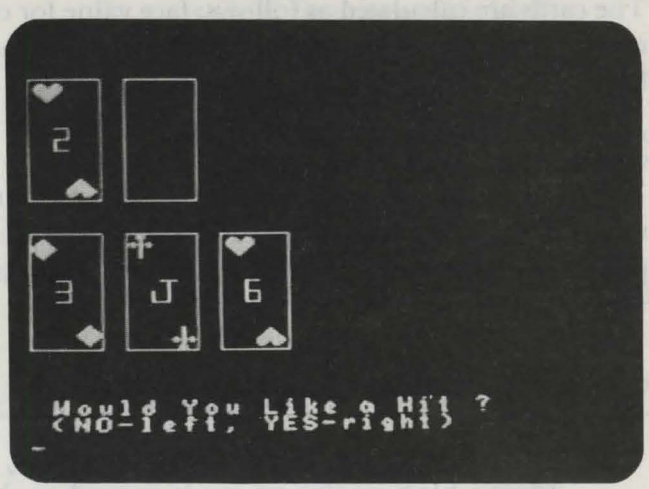
```
 0 HIMEM :41000
 1 REM  (c) 1984 by Brian Sawyer
 5 TEXT
10 pm$ = "Senior Artist"
15 FOR n = 1 TO 4
20 PRINT CHR$(162)
25 NEXT n
30 tb = 16-INT(LEN(pm$)/2)
35 PRINT TAB(tb); pm$
40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
45 PRINT " Hit Controller Button to Play"; CHR$(160)
50 iv = 0: r = 0
55 NORMAL: IF iv = 1 THEN  INVERSE
60 PRINT " Hit"; CHR$(160)
65 c = 0
```

```
 70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
    AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 GOSUB 1000
105 GOSUB 8000
107 GOSUB 2900
110 GOSUB 1200
120 GOSUB 1300
122 IF qt = 1 THEN 500
125 GOSUB 1500
130 GOTO 110
199 END
500 HOME
510 PRINT " Do You Want to Quit ?"
520 PRINT " (Type 'y' or 'n')"
530 GET a$
540 IF a$ <> "Y" AND a$ <> "y" THEN  qt = 0:
    GOSUB 2900: GOTO 125
550 TEXT: END
1000 REM  initialize variables
1005 DIM i(10)
1010 x = 128: y = 80
1014 sp = 1
1015 cl = 15
1016 pe = 1
1017 pn = 1
1018 sz = 1
1020 fx = x: fy = y
1030 HGR: HCOLOR  = 15
1040 ROT  = 0: SCALE  = 1
1050 RETURN
1200 p1 = PDL(5)
1205 xc = 0: yc = 0
1210 IF p1 = 0 THEN  RETURN
1220 IF p1 = 1 THEN  yc = -1
1225 IF p1 = 3 THEN  yc = -1: xc = 1
1230 IF p1 = 2 THEN  xc = 1
1235 IF p1 = 6 THEN  xc = 1: yc = 1
1240 IF p1 = 4 THEN  yc = 1
1245 IF p1 = 12 THEN  xc = -1: yc = 1
1250 IF p1 = 8 THEN  xc = -1
1255 IF p1 = 9 THEN  xc = -1: yc = -1
1257 IF pn = 0 THEN  XDRAW pe AT x, y
1258 xc = xc*sp: yc = yc*sp
1260 IF (x+xc > 2) AND (x+xc < 255) THEN  x = x+xc
1270 IF (y+yc > 2) AND (y+yc < 159) THEN  y = y+yc
1275 IF cl = 0 THEN  HCOLOR  = 15
1276 REM  rot=rt:rt=rt+1:if rt>64 then rt=0
```

```
1280 DRAW pe AT x, y
1285 IF cl = 0 THEN  HCOLOR  = 0
1290 RETURN
1300 p4 = PDL(13)
1305 IF p4 = 0 THEN  qt = 1: RETURN
1310 IF p4 = 15 THEN  RETURN
1320 IF p4 = 1 THEN  GOSUB 2000
1325 IF p4 = 2 THEN  GOSUB 2100
1330 IF p4 = 5 THEN  GOSUB 2400
1340 IF p4 = 6 THEN  GOSUB 2500
1345 IF p4 = 8 THEN  GOSUB 2500
1350 IF p4 = PDL(13) THEN 1350
1355 IF p4 = 3 THEN  GOSUB 2200
1360 IF p4 = 4 THEN  GOSUB 2300
1370 IF p4 = 7 THEN  GOSUB 2600
1375 IF p4 = 9 THEN  GOSUB 2800
1390 GOSUB 2900
1395 RETURN
1400 k = 1
1415 p4 = PDL(13)
1420 p2 = PDL(7)
1422 p3 = PDL(9)
1423 IF p3 = 1 AND k > 1 THEN 1450
1425 IF p2 = 1 AND k > 1 THEN  k = k-1: PRINT CHR$(163); " ";
     CHR$(163);
1426 IF p2 = 1 AND PDL(7) = 1 THEN 1426
1430 IF p4 > 9 OR k > 9 THEN 1415
1435 PRINT p4;
1440 i(k) = p4
1442 k = k+1
1443 IF PDL(13) <> 15 THEN 1443
1445 GOTO 1415
1450 no = 0: q = 0: FOR k1 = k-1 TO 1 STEP -1
1455 no = no+i(k1)*10^q: q = q+1
1460 NEXT k1: PRINT: RETURN
1470 END
1500 IF PDL(5) > 0 OR PDL(13) <> 15 THEN  RETURN
1501 IF cl = 0 THEN  HCOLOR  = 15
1502 XDRAW pe AT x, y
1505 FOR t = 1 TO 20: NEXT t
1510 DRAW pe AT x, y
1515 FOR t = 1 TO 20: NEXT t
1517 IF cl = 0 THEN  HCOLOR  = 0: XDRAW pe AT x, y
1520 GOTO 1500
2000 pn = 1
2005 cl = 15: HCOLOR  = cl
2010 RETURN
2100 pn = 0
2105 HCOLOR  = 0: cl = 0
2110 RETURN
2200 HOME
2205 PRINT " what color  (1-15)? ";
2210 GOSUB 1400
```

```
2215 IF no < 1 OR no > 15 THEN 2200
2220 cl = no
2225 HCOLOR  = cl
2230 RETURN
2300 HOME
2305 PRINT " what size   (1-15)? ";
2310 GOSUB 1400
2315 IF no < 1 OR no > 15 THEN 2300
2320 sz = no
2325 SCALE  = no
2330 RETURN
2400 fx = x: fy = y
2410 RETURN
2500 IF (fx = x) AND (fy = y) THEN  RETURN
2510 dx = fx-x: dy = fy-y
2512 fx = x: fy = y
2515 e = ABS(dx): IF e < ABS(dy) THEN  e = ABS(dy)
2520 dx = dx/e: dy = dy/e
2525 FOR n = 1 TO e
2530 DRAW pe AT fx, fy
2535 fx = fx+dx  fy = fy+dy
2540 NEXT n
2542 IF pd = 6 THEN  fx = x: fy = y
2545 RETURN
2600 HOME
2605 PRINT " radius size (1-80)? ";
2610 GOSUB 1400
2615 IF no < 1 OR no > 80 THEN 2600
2617 r = no
2620 FOR n = 0 TO 3.14159*2 STEP .05
2630 sx = x+COS(n)*r
2640 sy = y+SIN(n)*r
2650 IF sx > 255 THEN  sx = 255
2652 IF sy > 160 THEN  sy = 160
2654 IF sx < 2 THEN  sx = 2
2656 IF sy < 2 THEN  sy = 2
2670 DRAW pe AT sx, sy
2680 NEXT n
2690 RETURN
2800 HOME
2805 PRINT " which brush (1-5)? ";
2810 GOSUB 1400
2815 IF no < 1 OR no > 15 THEN 2800
2825 pe = no
2830 RETURN
2900 HOME
2910 PRINT " 1-draw  2-erase  3-color"
2920 PRINT " 4-size  5-ln. from  6-ln. to"
2930 PRINT " 7-circle  8- ray  9-brush"
2940 RETURN
8000 qs = 5: pt = 16766: tl = 41000
8005 of = qs*2+3
```

```
8010 by = tl: GOSUB 8100
8015 POKE pt, lo: POKE pt+1, hi
8020 POKE tl, qs
8030 FOR n = 1 TO qs
8040 by = of: GOSUB 8100
8050 POKE tl+n*2, lo: POKE tl+n*2+1, hi
8060 READ a: POKE tl+of, a: of = of+1
8065 IF a <> 0 THEN 8060
8070 NEXT n
8080 RETURN
8100 hi = INT(by/256): lo = by-hi*256
8130 RETURN
9000 DATA 1
9010 DATA 4,0
9200 DATA 52,38,47,61,0
9303 DATA 213,227,123,97,0
9403 DATA 45,62,47,0
9503 DATA 6,0
```



Blackjack

Program 9

Your Adam challenges you to a game of high-stakes Blackjack. You begin with a mere $500, and it is up to you to win as much money as you can from Adam, the dealer.

To start each hand, you place your bet by pressing the keypad numbers on game controller 1. If you make a mistake, use the left controller button as a backspace and then type in the correct number. Finally, enter your bet into the computer by pressing the right controller button. There is no limit on the amount you can bet, but be careful not to bet more than you have.

After placing your bet you are dealt two cards face up. The Adam is dealt two cards: one face up, one face down. You may now choose to be dealt additional cards. This is referred to as being *hit*. To be hit, press the right controller button. When you want to stop, or *stand,* press the left controller button. You can be dealt as many as five additional cards. If your cards' total point value exceeds 21, however, you have *busted* and lost the game. When you have finished playing your hand, and if you have not busted, the Adam must play its hand. But the rules it follows are stricter: if the Adam's hand totals 16, it must take a hit; if its hand is over 16, it must *stand.*

To win the round, the total of your hand must be higher than the Adam's. The cards are calculated as follows: face value for cards 2 through 10, 10 points for royalty (jack, queen, king), and 11 points for an ace. If a hand containing an ace has a total over 21, the ace is counted as 1 point instead of 11.

A *push* occurs when your total is the same as the Adam's. This is considered a tie, and no money is lost or gained.

If the first two cards dealt are an ace, and either a ten, jack, queen, or king, this is Blackjack. Whoever gets Blackjack wins the hand automatically. When the player gets Blackjack, the dealer pays one and a half times the bet.

Technical Description

Adam's high-resolution mode is used to show the cards in detail. Shape tables are used to draw the suits on the cards. Notice that the suit for each card appears twice: at the top of the card and inverted at the bottom of the card. Only one shape table is needed to draw both. The top suit is produced by printing the shape table, and the bottom suit is produced by rotating the shape table 180 degrees and then printing it. This is accomplished in line 2734, which reads ROT=32. The number that a shape can be rotated ranges from 0 (no rotation) to 64 (full rotation).

The program uses arrays to represent the deck of cards. The arrays no( ) and su( ) hold the rank and suit of each of the 52 cards in the deck. Lines 1100-1180 create an ordered deck using two FOR-NEXT loops. Lines 1200-1295 shuffle the deck by randomly permutating cards in the deck 52 times. The RND function is used in line 1220 to pick a random card in the deck and to exchange it for another card.

Important Variables

Variable	Function
be	Amount bet by the player.
bt	Indicates if either player has a Blackjack.
ca	Number of aces in the Adam's hand.
cx,cy	Position on which to draw the suit on the card.
cl	Total of the dealer's hand.
k	Random place at which the deck is cut (or split).
mo	Total of a player's money.

nf	Indicates if the number and suit should be drawn on a card.
no(*number*), su(*number*)	The number and suit of cards in the deck where *number* equals 52 entries, or the total number of cards in a deck.
x,y	Position where the number is drawn on a card. Variable x represents the horizontal coordinate and variable y the vertical coordinate.
ya	The number of aces in the player's hand.
y1	Total of the player's hand.

Program Description

Lines	Function
0	Shape table memory pointer.
1-95	Standard introduction.
100-999	Main loop program.
1000-1050	Subroutine to enter high-resolution mode and initialize variables.
1100-1180	Subroutine to build the deck of cards.
1200-1295	Subroutine to shuffle the deck.
1300-1340	Subroutine to request the player bet.
1400-1470	Subroutine to input a number from the keypad.
1700-1770	Subroutine to deal the Adam one card.
1800-1870	Subroutine to deal the player a card.
2400-2450	Subroutine to read in the data for drawing the numbers on the cards.
2700-2860	Subroutine to draw a card on the screen.
8000-8130	Subroutine to load the data for the shape table into memory.
9103-10053	Shape table data.
10090-10230	Data for drawing card numbers.

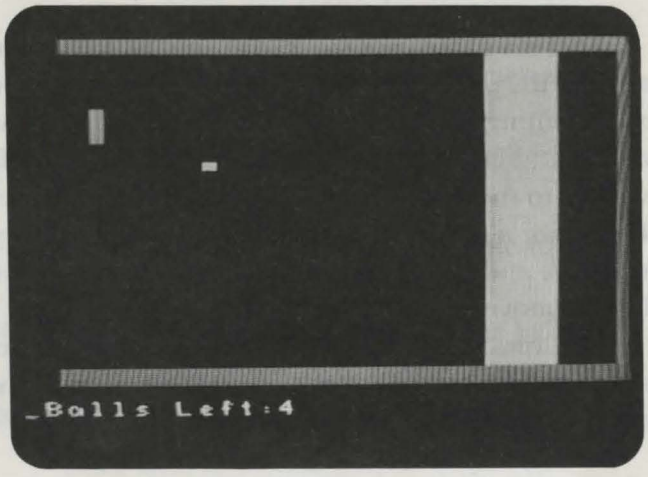
```
  0 HIMEM :41000
  1 REM  (c) 1984 by Brian Sawyer
  5 TEXT
 10 pm$ = "Blackjack"
 15 FOR n = 1 TO 4
 20 PRINT CHR$(162)
 25 NEXT n
 30 tb = 17-INT(LEN(pm$)/2)
 35 PRINT TAB(tb); pm$
 40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
 45 PRINT " Hit Controller Button to Play"; CHR$(160)
 50 iv = 0: r = 0
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
    AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 GOSUB 1000
115 GOSUB 2400
120 GOSUB 1100
130 GOSUB 1200
140 GOSUB 1300
142 IF be = 0 THEN 700
145 GOSUB 1018
150 GOSUB 1700
160 nf = 1: GOSUB 1700: nf = 0
165 bq = 0: IF bt = 1 THEN  bq = 1
170 GOSUB 1800
180 GOSUB 1800
185 IF bq = 1 THEN  PRINT " Dealer Ha's Blackjack":
    GOTO 300
190 IF bt = 1 THEN  PRINT "You have Blackjack!":
    mo = mo+INT(be*1.5): GOTO 400
200 HOME: PRINT " Would You Like a Hit ?"
205 PRINT " (NO-left, YES-right)"
210 p1 = PDL(7): p2 = PDL(9):
    IF p1 = 0 AND p2 = 0 THEN 210
215 HOME
220 IF p1 = 1 THEN 240
225 GOSUB 1800
235 IF y1 < 22 THEN 239
237 IF ya = 0 THEN  PRINT " Busted!": GOTO 300
238 ya = ya-1: y1 = y1-10
239 IF yn < 7 THEN 200
240 no = no(2): su = su(2): t = cn: cl = 2:
    rw = 1: GOSUB 2700
275 IF cn > 6 THEN 290
280 IF c1 < 22 THEN 285
```

```
282 IF ca > 0 THEN  ca = ca-1: c1 = c1-10: GOTO 285
284 PRINT " Dealer Busts!": GOTO 293
285 IF c1 > 16 THEN 290
286 GOSUB 1700: GOTO 275
290 IF c1 > y1 THEN 300
292 IF c1 = y1 THEN  PRINT " Push- Nobody Wins": GOTO 400
293 PRINT " You Win the Hand!"
295 mo = mo+be
296 GOTO 400
300 PRINT " Dealer Wins!"
310 mo = mo-be
320 IF mo <= 0 THEN 600
330 GOTO 400
400 REM  play new hand
405 FOR d1 = 1 TO 1000: NEXT d1
420 GOTO 120
500 PRINT " You Win!"
510 mo = mo+be
520 GOTO 400
600 FOR d1 = 1 TO 1000: NEXT d1
605 HOME
610 PRINT " You Are Broke!"
620 PRINT " You Lose!"
630 END
700 TEXT: END
999 END
1000 REM  initialize variables
1015 DIM c(14, 20): mo = 500: DIM i(10)
1017 DIM no(52), su(52): RETURN
1018 c1 = 0: y1 = 0
1019 cn = 0: yn = 0: i = 0
1021 ca = 0: ya = 0
1030 HGR: HCOLOR  = 1
1040 ROT  = 0: SCALE  = 1
1050 RETURN
1100 REM  make deck
1110 q = 1
1120 FOR n = 1 TO 13
1130 FOR m = 1 TO 4
1150 no(q) = n
1155 su(q) = m
1157 q = q+1
1160 NEXT m
1170 NEXT n
1180 RETURN
1200 REM  shuffle
1210 FOR n = 1 TO 52
1220 k = INT(RND(1)*52+1)
1230 t = no(n)
1240 no(n) = no(k)
1250 no(k) = t
1260 t = su(n)
```

```
1270 su(n) = su(k)
1280 su(k) = t
1290 NEXT n
1295 RETURN
1300 HOME
1310 PRINT " You Have "; mo; " Dollars"
1320 PRINT " Bet How Much? ";
1325 GOSUB 1400: be = no
1330 IF be > mo THEN 1300
1340 RETURN
1400 REM  check paddle
1410 k = 1
1415 pd = PDL(13)
1420 p1 = PDL(7)
1422 p2 = PDL(9)
1423 IF p2 = 1 AND k > 1 THEN 1450
1425 IF p1 = 1 AND k > 1 THEN  k = k-1:
     PRINT CHR$(163); " "; CHR$(163);
1426 IF p1 = 1 AND PDL(7) = 1 THEN 1426
1430 IF pd > 9 OR k > 9 THEN 1415
1435 PRINT pd;
1440 i(k) = pd
1442 k = k+1
1443 IF PDL(13) <> 15 THEN 1443
1445 GOTO 1415
1450 no = 0: q = 0: FOR k1 = k-1 TO 1 STEP -1
1455 no = no+i(k1)*10^q: q = q+1
1460 NEXT k1: PRINT: RETURN
1470 END
1700 REM  computer hit
1705 bt = 0
1710 i = i+1
1720 cn = cn+1
1730 no = no(i): su = su(i)
1735 IF (c1 = 11 AND no > 8 AND no < 13) THEN  bt = 1
1736 IF (no = 13 AND c1 = 10) THEN  bt = 1
1740 rw = 1: cl = cn: GOSUB 2700
1750 IF no = 13 THEN  c1 = c1+1: IF c1+10 < 22
     THEN  c1 = c1+10: ca = ca+1
1755 IF no = 13 THEN 1770
1760 IF no < 9 THEN  c1 = c1+no+1: GOTO 1770
1765 IF no < 13 THEN  c1 = c1+10: GOTO 1770
1767 c1 = c1+1
1770 RETURN
1800 REM  player hit
1805 bt = 0
1810 i = i+1
1820 yn = yn+1
1830 no = no(i): su = su(i)
1835 IF (y1 = 11 AND no > 8 AND no < 13) THEN  bt = 1
1836 IF (no = 13 AND y1 = 10) THEN  bt = 1
1840 rw = 2: cl = yn: GOSUB 2700
```

```
1850 IF no = 13 THEN  ya = ya+1
1860 IF no < 9 THEN  y1 = y1+no+1: GOTO 1870
1865 IF no < 13 THEN  y1 = y1+10: GOTO 1870
1867 y1 = y1+11
1870 RETURN
2400 GOSUB 8000
2410 FOR n = 1 TO 13
2415 m = 0
2420 READ c(n, m)
2426 HPLOT cx, cy+50
2430 IF c(n, m) <> 999 THEN  m = m+1: GOTO 2420
2440 NEXT n
2450 RETURN
2700 REM  print card,nu,su,cl,rw
2710 cx = (cl-1)*36+7: cy = (rw-1)*70+10
2715 HCOLOR  = 11: wd = 27
2720 HPLOT cx, cy TO cx+wd, cy
2722 HPLOT cx+wd, cy TO cx+wd, cy+55
2724 HPLOT cx+wd, cy+55 TO cx, cy+55
2727 HPLOT cx, cy+55 TO cx, cy
2728 IF nf = 1 AND bt = 0 THEN  RETURN
2729 f = 0: IF su > 2 THEN  f = 9
2730 HCOLOR  = 15: IF su > 2 THEN  HCOLOR  = 6
2732 ROT  = 0
2733 DRAW su AT cx+5, cy+2+f
2734 ROT  = 32
2735 DRAW su AT cx+22, cy+53-f
2800 REM  pass nu,cx,cy
2801 HCOLOR  = 8
2805 cx = cx+10: cy = cy+22
2810 n = 0
2815 HPLOT c(nu, 0)+cx, c(nu, 1)+cy
2820 IF c(nu, n) = 999 THEN 2860
2830 x = c(nu, n)+cx: y = c(nu, n+1)+cy
2840 HPLOT  TO x, y: HPLOT x, y
2850 n = n+2: GOTO 2820
2860 RETURN
8000 qs = 4: pt = 16766: tl = 41000
8005 of = qs*2+3
8010 by = tl: GOSUB 8100
8015 POKE pt, lo: POKE pt+1, hi
8020 POKE tl, qs
8030 FOR n = 1 TO qs
8040 by = of: GOSUB 8100
8050 POKE tl+n*2, lo: POKE tl+n*2+1, hi
8060 READ a: POKE tl+of, a: of = of+1
8065 IF a <> 0 THEN 8060
8070 NEXT n
8080 RETURN
8100 hi = INT(by/256): lo = by-hi*256
8130 RETURN
9103 DATA 62,45,55,63,45,45,46,63,63,63,62
```

```
 9106 DATA    45,45,45,45,46,63,63,63,63,63,173
 9109 DATA    61,47,45,45,45,61,62,63,63,47,46
 9112 DATA    45,61,62,47,38,0
 9553 DATA   77,9,53,63,45,221,219,27,63,45,173
 9556 DATA   63,63,47,45,45,45,45,45,60,39,173
 9559 DATA   218,219,227,60,175,45,151,63,63,45,45
 9562 DATA   45,45,45,245,63,63,63,63,119,45,45
 9565 DATA   45,245,63,63,119,45,245,123,45,0
10003 DATA   36,36,36,36,36,53,54,54,54,54 52
10006 DATA   38,36,36,36,36,47,61,44,63,61,47
10009 DATA   62,53,54,61,45,47,47,61,37,53,62
10012 DATA   60,47,61,45,47,61,63,63,63,63,37
10015 DATA   54,52,52,37,52,0
10041 DATA   45,45,61,60,63,37,45,55,61,45 55
10044 DATA   45,63,60,36,38,60,63,62,39,45,45
10047 DATA   45,45,46,37,63,39,63,63,63,45,45
10050 DATA   45,45,39,63,63,63,37,45,45,39,63
10053 DATA   37,6,0
10090 DATA   0,0, 5,0, 5,5, 0,5, 0,10, 5,10, 999
10110 DATA   0,0, 5,0, 5,5, 0,5, 5,5, 5,10, 0,10 ,999
10120 DATA   0,0, 0,5, 5,5, 5,0, 5,10, 999
10130 DATA      5,0, 0,0, 0,5, 5,5, 5,10, 0,10 .999
10140 DATA       5,0, 0,0, 0,5, 5,5, 5,10, 0,10, 0,5, 999
10150 DATA   0,0, 5,0, 5,10, 999
10160 DATA  0,0, 0,10, 5,10,5,0, 0,0, 0,5, 5,5, 999
10170 DATA     0,0, 5,0, 5,10, 5,5, 0,5, 0,0,999
10190 DATA  2,0, 2,10, 0,10, 10,10, 10,0, 6,0, 6,10, 999
10200 DATA   3,0, 9,0, 6,0, 6,10, 0,10, 0,5, 999
10210 DATA  0,0, 0,10, 3,10, 3,8, 3,12, 3,10, 5,10, 5,0,
            0,0 , 999
10220 DATA   0,0, 0,10, 0,6, 5,10, 0,6, 5,0, 999
10230 DATA     0,0, 0,10, 0,0, 5,0, 5,10, 5,5, 0,5, 999
```



Barrier Ball

Program 10

In Barrier Ball you try to break a hole in one wall of the court by swatting a ball with your racket. The court is built of two light blue side walls and a thick light blue barrier wall at the opposite end. There is no wall behind you. When a ball is released into the court and you hit it, the ball will bounce off the walls and the barrier. Every time the ball hits the barrier, a piece of the wall is chipped loose (this is indicated in yellow). If the ball hits a place on the barrier that is already chipped, it will break it off (this is indicated in black). The object of this game is to break a hole in the barrier and then hit the ball through the hole.

After the ball bounces off the barrier, you must prevent it from leaving the court by swatting it with your racket. You can maneuver your racket with the joystick from game controller 1. If you miss the ball, you get another one, but if you miss the ball three times before breaking the barrier, you lose the game.

Technical Description

This game uses the PLOT and SCRN commands to make the ball bounce around the court. The variables x,y contain the screen coordinates of the ball with x containing the horizontal coordinate and y the vertical coordinate. Variables xc and yc indicate the direction the ball is moving. If xc has a value of 1, movement will be to the right, −1 will be movement to the left, and 0 will be no movement. If yc has a value of 1, movement will be downward, −1 will be movement upward, and 0 will be no movement. Adding xc to x and yc to y will cause the ball to move.

Lines 1305-1310 use the SCRN command to determine whether the ball is about to hit the wall, barrier, or racket. If it is, the value of xc or yc is changed, causing the ball to bounce in a new direction.

Important Variables

Variable	Function
p	Contains the value of PDL(5), the direction of the joystick on game controller 1.
px,py	Screen coordinates of the racket; px contains the horizontal coordinate and py the vertical coordinate.
s	Color of the space the ball is about to move upon.
x,y	Screen coordinates of the ball; x contains the horizontal coordinate and y the vertical coordinate.
xc,yc	Direction the ball is moving. Used in combination, the variables will give a diagonal direction.

Program Description

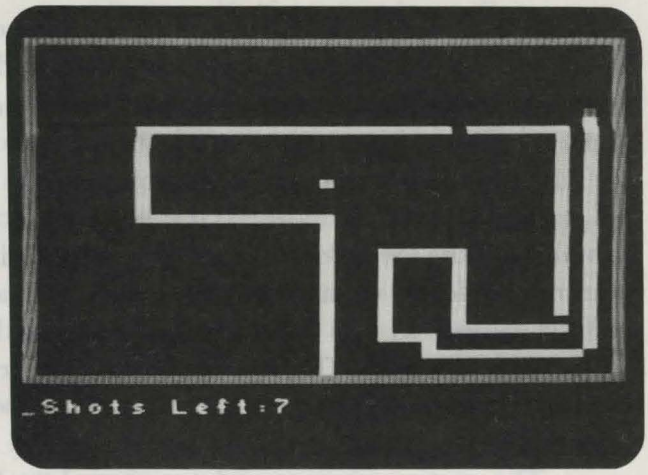
Lines	Description
1-95	Standard introduction.
100-199	Main program.
1000-1080	Subroutine to enter low-resolution mode and draw the border and barriers.

1100-1130	Subroutine to initialize the variables and place the ball at a random position.
1200-1250	Subroutine to draw the racket using the VLIN command.
1300-1420	Subroutine to bounce the ball off the wall or barrier.
2000-2030	Subroutine to break a hole in the barrier.
2100-2110	Subroutine to get another ball if the first one is lost.
3000-3060	Subroutine to announce your victory.
4000-4020	Subroutine to print the statistics.
5000-5060	Subroutine to announce your loss.

```
  1 REM  (c) 1984 by Brian Sawyer
  5 TEXT
 10 pm$ = "Barrier Ball"
 15 FOR n = 1 TO 4
 20 PRINT CHR$(162)
 25 NEXT n
 30 tb = 16-INT(LEN(pm$)/2)
 35 PRINT TAB(tb); pm$
 40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
 41 NORMAL
 42 d = d+1: IF d/2 = INT(d/2) THEN  INVERSE
 45 PRINT " Hit Controller Button to Play"; CHR$(160)
 50 iv = 0: r = 0
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0)
    AND (PDL(9) = 0) AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 GOSUB 1000
105 GOSUB 1100
107 GOSUB 4000
110 GOSUB 1200
115 GOSUB 1200
120 GOSUB 1300
```

```
 125 IF x = 2 THEN  GOSUB 1110: GOSUB 2100
 127 IF bx > 37 THEN 3000
 129 IF bl = 0 THEN 5000
 130 GOTO 110
 199 END
1000 GR
1005 PRINT
1010 COLOR  = 2
1015 c = 0
1020 VLIN 0, 39 AT 39
1030 HLIN 2, 39 AT 0
1031 HLIN 2, 39 AT 1
1040 HLIN 2, 39 AT 39
1041 HLIN 2, 39 AT 38
1045 PLOT 39, 39
1046 COLOR  = 14
1049 FOR n = 30 TO 34
1050 y1 = 2: y2 = 35: x1 = n: GOSUB 1500
1057 c = c+1
1070 NEXT n
1080 RETURN
1100 py = 2: bl = 4
1110 xc = 1: yc = 1-INT(RND(1))*2
1120 x = 4: y = 8+INT(RND(1)*26)
1130 RETURN
1200 p = PDL(5)
1205 COLOR  = 0
1220 IF (p = 1) AND (py > 2) THEN  py = py-1: PLOT 4, py+4
1230 IF (p = 4) AND (py < 34) THEN  py = py+1: PLOT 4, py-1
1235 COLOR  = 1
1240 y1 = py: y2 = 3: x1 = 4: GOSUB 1500
1250 RETURN
1300 s = SCRN(x+xc, y+yc): IF s = 0 THEN 1400
1301 h = 0
1302 bx = x+xc: by = y+yc
1305 IF SCRN(x, y-1) > 0 OR SCRN(x, y+1) > 0
     THEN  yc = -yc: h = 1
1310 IF SCRN(x+1, y) > 0 OR SCRN(x-1, y) > 0
     THEN  xc = -xc: h = 1
1315 IF h = 0 THEN  xc = -xc: yc = -yc
1320 IF s > 7 THEN  GOSUB 2000
1400 COLOR  = 0: PLOT x, y
1405 x = x+xc: y = y+yc
1407 IF x = 2 THEN  RETURN
1410 COLOR  = 8: PLOT x, y
1420 RETURN
1500 FOR y3 = y1 TO y1+y2: PLOT x1, y3: NEXT y3: RETURN
2000 by = INT((by-2)/4)*4+2
2010 IF s = 8 THEN  COLOR  = 0: c = c-1: GOTO 2020
2015 COLOR  = 8
2020 y1 = by: y2 = 3: x1 = bx: GOSUB 1500
2030 RETURN
2100 bl = bl-1: GOSUB 4000
```

```
2102 IF bl = 0 THEN  RETURN
2105 IF PDL(5) = 0 THEN 2105
2110 RETURN
3000 HOME
3010 PRINT "       *** You Win! ***"
3020 PRINT "Would you like to play again? ";
3030 GET a$
3040 IF a$ = "y" OR a$ = "Y" THEN 95
3050 TEXT
3060 END
4000 IF bl = 0 THEN  RETURN
4005 PRINT " Balls Left:"; bl
4010 PRINT CHR$(160);
4020 RETURN
5000 HOME
5010 PRINT "       *** Game Over ***"
5020 PRINT "Would you like to play again? ";
5030 GET a$
5040 IF a$ = "y" OR a$ = "Y" THEN 95
5050 TEXT
5060 END
```



Arena

Program 11

You are a gladiator locked in an arena with a killer bee. You are armed with a gun containing ten shots. When you move around in the arena, you leave a trail of netting through which the bee usually cannot pass. You try to surround the bee with enough netting to get a shot at it. But be warned: when the bee is confined in a narrow area, it becomes angry enough to break through the netting.

You move using the joystick on game controller 1. The right controller button fires your gun in the direction you are moving. Your gun can also shoot holes in the netting you have laid so you can move through it freely.

If you run into the netting you have laid, collide with the bee, or run out of ammunition, you die. You win when you shoot and hit the killer bee.

Technical Description

This action game operates on the low-resolution screen. Variables px and py are the coordinates of the player; py contains the vertical coordinate and px the horizontal coordinate. The direction of the player is contained in the variables kx and ky. In lines 1210-1240 the value of the joystick control, PDL(5), is checked to see whether the joystick has been moved. If the joystick has been moved to the left, variable kx will contain −1; if it has been moved to the right, variable kx will contain 1. If the joystick has been moved up, variable ky will contain −1; if it has been moved down, variable ky will contain 1. The value of kx will be added to px and the value of ky will be added to py, causing the player to move.

The values of PDL(7) and PDL(9) are checked in line 2000 to see if either the left or right controller button has been pressed, indicating that a shot should be fired.

The starting position and direction of the killer bee is set in lines 1110-1120. The RND function picks a random vertical coordinate for the starting position of the bee in line 1120.

Important Variables

Variable	Function
cr	Indicates if a player has crashed into the netting.
dl	The number of shots.
f	Counts the intervals at which a player is allowed to move.
ht	Indicates if the bee has been shot.
kx,ky	The direction that the player is moving. The variable kx indicates direction along the horizontal axis and ky indicates direction along the vertical axis.
p	Contains the value of PDL(5), the position of the joystick on game controller 1.
px,py	Screen coordinates of the player; px represents the horizontal coordinate and py the vertical coordinate.
s	Color of the space that the player is about to move upon.

x,y	Screen coordinates of the killer bee; x contains the horizontal coordinate and y the vertical coordinate.
xc,yc	Contains the direction that the ball is moving. The variable xc contains the direction along the horizontal axis and variable yc contains the direction of the vertical axis.

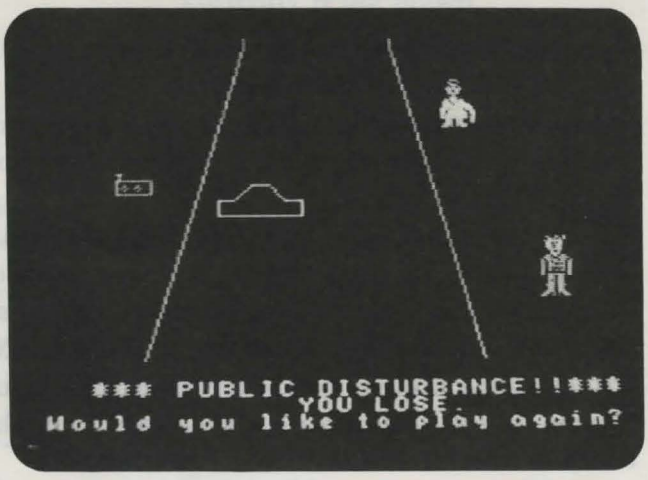
Program Description

Lines	Function
1-95	Standard introduction.
100-199	Main program.
1000-1046	Subroutine to enter low-resolution graphics mode and draw the border.
1050-1080	Subroutines to draw the border.
1100-1130	Subroutine to initialize variables.
1200-1250	Subroutine to check for joystick movement and move player accordingly.
1300-1420	Subroutine to move the killer bee.
2000-2110	Subroutine to shoot in the same direction a player is moving.
4000-4020	Subroutine to print the score.
6000-6010	Subroutine to inform you that you lost.
7000-7070	Subroutine to inform you that you won.

```
 1 REM  (c) 1984 by Brian Sawyer
 5 TEXT
10 pm$ = "Arena"
15 FOR n = 1 TO 4
20 PRINT CHR$(162)
25 NEXT n
30 tb = 16-INT(LEN(pm$)/2)
35 PRINT TAB(tb); pm$
40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
45 PRINT " Hit Controller Button to Play"; CHR$(160)
50 iv = 0: r = 0
```

```
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
    AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 GOSUB 1000
105 GOSUB 1100
107 GOSUB 4000
115 f = f+1: IF f/3 = INT(f/3) THEN  GOSUB 1200
120 GOSUB 1300
125 GOSUB 2000
129 IF cr = 1 THEN 5000
130 IF dl = 0 THEN 3000
138 IF ht = 1 THEN 7000
140 GOTO 115
199 END
1000 GR
1005 PRINT
1010 COLOR  = 2
1015 c = 0
1020 VLIN 0, 39 AT 39
1025 VLIN 0, 39 AT 0
1030 HLIN 0, 39 AT 0
1040 HLIN 0, 39 AT 39
1045 PLOT 39, 39
1046 COLOR  = 14: RETURN
1050 FOR n = 24 TO 36 STEP 3
1055 FOR m = 3 TO 36 STEP 4
1056 VLIN m, m+3 AT n
1057 c = c+1
1060 NEXT m
1070 NEXT n
1080 RETURN
1100 py = 38: dl = 10: px = 19
1105 kx = 0: ky = -1
1106 ht = 0
1110 xc = 1: yc = 1-INT(RND(1))*2
1120 x = 4: y = 2+INT(RND(1)*37)
1130 RETURN
1200 p = PDL(5)
1210 IF p = 1 THEN  ky = -1: kx = 0: GOTO 1241
1220 IF p = 2 THEN  kx = 1: ky = 0: GOTO 1241
1230 IF p = 4 THEN  ky = 1: kx = 0: GOTO 1241
1240 IF p = 8 THEN  kx = -1: ky = 0
1241 COLOR  = 5: PLOT px, py
1244 px = px+kx: py = py+ky
1245 IF SCRN(px, py) > 0 THEN  cr = 1: RETURN
1247 COLOR  = 4: PLOT px, py
```

```
1250 RETURN
1300 s = SCRN(x+xc, y+yc): IF s = 0 THEN 1400
1301 h = 0
1305 IF SCRN(x, y-1) > 0 OR SCRN(x, y+1) > 0 THEN  yc =
     -yc: h = 1
1310 IF SCRN(x+1, y) > 0 OR SCRN(x-1, y) > 0 THEN  xc =
     -xc: h = 1
1320 IF h = 0 THEN  xc = -xc: yc = -yc
1400 COLOR  = 0: PLOT x, y
1405 x = x+xc: y = y+yc
1410 COLOR  = 8: PLOT x, y
1420 RETURN
2000 IF PDL(7) = 0 AND PDL(9) = 0 THEN  RETURN
2005 dl = dl-1
2006 GOSUB 4000
2010 tx = px: ty = py
2020 tx = tx+kx: ty = ty+ky
2025 s = SCRN(tx, ty)
2026 IF s = 2 THEN  RETURN
2027 IF s = 8 THEN  ht = 1: RETURN
2029 COLOR  = 9
2030 PLOT tx, ty
2034 GOSUB 1300
2035 COLOR  = 0
2040 PLOT tx, ty
2050 GOTO 2020
2100 bl = bl-1: GOSUB 4000
2105 IF PDL(5) = 0 THEN 2105
2110 RETURN
3000 HOME
3010 PRINT "           OUT OF SHOTS!"
3020 GOTO 6000
4000 IF dl = 0 THEN  RETURN
4005 PRINT " Shots Left:"; dl
4010 PRINT CHR$(160);
4020 RETURN
5000 HOME
5010 PRINT "              CRASH!!!"
5020 GOTO 6000
6000 PRINT "        **** You Lose ***"
6010 GOTO 7030
7000 HOME
7010 PRINT "           YOU GOT HIM!"
7020 PRINT "       **** You Win! ****"
7030 PRINT "Would you like to play again? ";
7040 GET a$
7050 IF a$ = "y" OR a$ = "Y" THEN 95
7060 TEXT
7070 END
```



Mr. Helpful

Program 12

In this program, you are Mr. Helpful, a public-spirited citizen trying to aid his fellow humans. Your mission is to prevent a public disturbance that is brewing on a downtown street. An elderly woman and a punk rocker stand on one side of the street along with a radio. You have a car, and you must move all three to the other side of the street. Unfortunately, this is not as simple as it sounds.

Your compact car can only hold one item or person at a time, so you can transport no more than one of the three across the street during each trip. You can also cross the street alone. A word of caution: you must not leave the woman alone with the blaring radio or she will smash it; if you leave the punk and the woman alone, they will fight over the finer points of musical taste. Both of these events are public disturbances and must be avoided!

When you begin the game, there will be a slight delay while the items are drawn on the screen. To move any of the items, press the number on the keypad corresponding to the following: 1=elderly lady, 2=radio, and 3=punk rocker. Pressing 4 on the keypad will move the car to the opposite side of the street. Remember, however, that leaving two incompatible items together while you cross the street will result in a public disturbance.

Technical Description

This program presents an alternative method of drawing detailed images on the high-resolution screen. Instead of using shape tables, the items in this game are drawn using the HPLOT command on line 1230. Each item is made by plotting a number of individual dots. These dot patterns appear in data statements in lines 9000-9222.

Notice that periods and x's form the actual pictures of the items as these appear in the data statements. A period represents a blank space, while an x represents a dot. Each data line comprised of x's and periods is treated as one data string and is read into the variable a$ on line 1120. Notice that some of these data strings have a number preceding them. This number determines the color of the x's and periods within each string and is not changed until another number is encountered.

Since there is no pattern between string and number, the program cannot determine if it is reading a number or a string. Therefore each number is read into a$ as a string. The string variable a$ is then tested on line 1125 to determine whether it is a string or a string that is a number. If it is a string that is a number, line 1125 converts it to its numerical value. Lines 9000-9034 contain the data to display the punk, 9100-9109 the data for the radio, and 9200-9222 the data for the woman. You can modify the items yourself by changing this data. Put a period in an area that you want blank and an x where you want a dot plotted.

The car is drawn on the high-resolution screen using the HPLOT TO command. The data statement in lines 9310-9320 contains the coordinates of the points that, when connected, make up the car. Line 9300 tells how many points are in the car. Line 1420 draws a line from the last point shown in the car to the current point.

Important Variables

Variable	Function
a1,a2,a3	Position on the street of the woman, the radio, and the punk, respectively.
b(*number1*, *number2*, *number3*)	Array containing dot patterns for each of the three items. The number of entries for *number1* is 3, for *number2* is 15, and for *number3* is 29.
cp	Position of the car on the street.
pd	Contains the value of PDL(13), the number pressed on the keypad of game controller 1.

s	The number representing the item currently being drawn.
x,y	Screen position of the character being drawn; the variable x contains the horizontal coordinate and variable y the vertical coordinate.
x1,y1	Screen position of the car; variable x1 contains the horizontal coordinate and y1 the vertical coordinate.

Program Description

Lines	Function
0	Shape table memory pointer.
1-95	Standard introduction.
100-300	Main program; checks for keypad input.
400-599	Messages to inform you of your loss: lady smashes radio or lady and punk rocker left alone.
600-620	Messages to inform you that you prevented a public disturbance.
1000-1040	Subroutine to enter high-resolution graphics mode and draw the street using the HPLOT TO command.
1100-1199	Subroutine to read in points to be plotted for each character.
1200-1260	Subroutine to draw a character on the screen using repeated HPLOT commands.
1400-1430	Subroutine to draw the car using the HPLOT TO command.
1600-1650	Subroutine that calls other subroutines that will draw a character.
9000-9222	Data for characters.
9300-9320	Data to draw the lines that connect the points of the car.

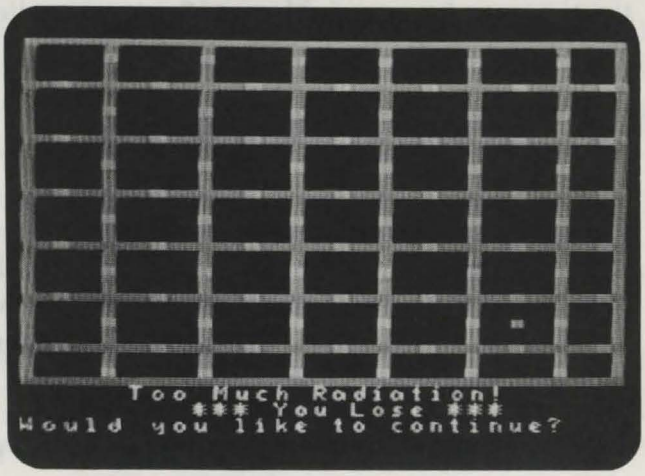
```
 0 HIMEM :41000
 1 REM  (c) 1983 by Brian Sawyer
 5 TEXT
10 pm$ = "Mr. Helpful"
```

```
 15 FOR n = 1 TO 4
 20 PRINT CHR$(162)
 25 NEXT n
 30 tb = 16-INT(LEN(pm$)/2)
 35 PRINT TAB(tb); pm$
 40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
 45 PRINT " Hit Controller Button to Play"; CHR$(160)
 50 iv = 0: r = 0
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
    AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 GOSUB 1000
110 GOSUB 1100
120 FOR s = 1 TO 3
130 GOSUB 1600
140 NEXT s
150 x1 = 80+cp*45: y1 = 80: GOSUB 1400
155 IF (a1 = a2) AND (cp <> a1) THEN 400
156 IF (a1 = a3) AND (cp <> a1) THEN 500
157 IF (a1 = 0) AND (a1 = 0) AND (a3 = 0) THEN 600
160 HOME
170 HOME
175 PRINT " take who across? (1,2,3, or 4)"
180 pd = PDL(13): IF pd = 15 THEN 180
190 IF pd < 1 OR pd > 4 THEN 170
191 IF pd = 1 AND a1 <> cp THEN 170
192 IF pd = 2 AND a2 <> cp THEN 170
193 IF pd = 3 AND a3 <> cp THEN 170
195 es = 1
196 IF pd > 0 AND pd < 4 THEN  s = 4-pd: GOSUB 1600
200 IF pd = 1 THEN  a1 = 1-a1
210 IF pd = 2 THEN  a2 = 1-a2
220 IF pd = 3 THEN  a3 = 1-a3
230 cp = 1-cp: GOSUB 1400
235 es = 0
236 IF pd > 0 AND pd < 4 THEN  s = 4-pd: GOSUB 1600
300 GOTO 150
400 HOME
405 PRINT " OOPS- lady & radio left alone!"
406 GOSUB 2000
430 HOME
440 PRINT "   LADY SMASHES RADIO!!!"
450 GOSUB 2000
460 GOSUB 2100
462 FOR dl = 1 TO 4000: NEXT dl
464 PRINT "Would you like to play again? "
466 GET a$
```

```
468 IF a$ = "y" OR a$ = "Y" THEN 95
470 TEXT: END
500 HOME
510 PRINT " OOPS- lady & punk left alone!"
520 GOSUB 2000
530 HOME
540 PRINT " LADY: 'Your music-- it's just"
550 PRINT " noise!'"
560 GOSUB 2000
565 HOME
570 PRINT " PUNK: 'Buzz off !!!'"
580 GOSUB 2000
590 GOSUB 2100
592 FOR dl = 1 TO 4000: NEXT dl
594 PRINT "Would you like to play again? "
596 GET a$
598 IF a$ = "y" OR a$ = "Y" THEN 95
599 TEXT: END
600 HOME
610 PRINT "   *** YOU SAVED THE DAY! ***"
615 FOR dl = 1 TO 5000: NEXT dl
617 TEXT
620 END
1000 HGR
1010 HCOLOR  = 1
1015 HPLOT 90, 0 TO 50, 150
1016 HPLOT 150, 0 TO 190, 150
1040 RETURN
1100 READ ns
1102 DIM b(ns, 15, 29)
1105 DIM cl(ns, 29), x(10), y(10)
1107 DIM xs(ns), ys(ns)
1108 FOR n = 1 TO ns: READ xs(n), ys(n): NEXT n
1110 FOR s = 1 TO ns
1115 FOR n = 1 TO ys(s)
1120 READ a$
1125 IF VAL(a$) > 0 THEN  cl(s, n) = VAL(a$): READ a$
1130 FOR m = 1 TO xs(s)
1135 IF MID$(a$. m, 1) = "x" OR MID$(a$, m, 1) = "."
     THEN 1140
1136 GOTO 1135
1140 IF MID$(a$, m, 1) = "x" THEN  b(s, m, n) = 1
1150 NEXT m
1160 NEXT n
1165 NEXT s
1170 READ q
1172 FOR n = 1 TO q
1175 READ x(n), y(n)
1176 x(n) = INT(x(n)/1.5)
1177 y(n) = INT(y(n)/1.5)
1180 NEXT n
1190 a1 = 1
1192 a2 = 1
```

```
1194 a3 = 1
1196 cp = 1
1199 RETURN
1200 HCOLOR  = 1
1210 FOR cy = 1 TO ys(s)
1220 FOR cx = 1 TO xs(s)
1225 by = INT(cy/1)
1227 IF cl(s, by) > 0 THEN  HCOLOR  = cl(s, by)
1228 IF es = 1 THEN  HCOLOR  = 0
1230 IF b(s, cx, by) = 1 THEN  HPLOT x+cx, y+cy
1240 NEXT cx
1250 NEXT cy
1260 RETURN
1400 HCOLOR  = 1: IF es = 1 THEN  HCOLOR  = 0
1405 HPLOT x1, y1
1410 FOR n = 1 TO q
1420 HPLOT  TO x(n)+x1, y(n)+y1
1422 HPLOT x(n)+x1, y(n)+y1
1425 NEXT n
1430 RETURN
1600 IF s = 3 THEN  x = 55+a1*115: y = 18
1630 IF s = 2 THEN  x = 40+a2*160: y = 60
1640 IF s = 1 THEN  x = 30+a3*180: y = 90
1650 GOSUB 1200: RETURN
2000 FOR dl = 1 TO 3000
2010 NEXT dl
2020 RETURN
2100 HOME
2105 PRINT "   *** PUBLIC DISTURBANCE!!***"
2110 PRINT "              YOU LOSE."
2120 RETURN
8000 DATA 3
8010 DATA          14,29, 14,10, 14,22
9000 DATA 8, ...x...x..x...
9001 DATA    ....x.x.x.x...
9002 DATA    ..x.xxxxxx....
9003 DATA    ...xx.x.xx....
9004 DATA    ...xxx.xxx....
9005 DATA    ...xxx.xxx....
9006 DATA    ....xxxxxx...
9007 DATA    .....x..x... .
9008 DATA    .....xxxx.....
9009 DATA 5, ...xx.xx.xxx..
9010 DATA    ..xxxx..xxxxx.
9011 DATA    .x.xxxxx.xxx.x
9012 DATA    .xx.x.xxxxx.xx
9013 DATA    .xx.xxx.x.x.xx
9014 DATA    .xx.xxxxxxx.xx
9015 DATA    .xx.x.x.x.x.xx
9016 DATA    .xx.xx.x.xx.xx
9017 DATA    .xx.........xx
9018 DATA    .xx.xxxxxxx.xx
9019 DATA  6,....xxx.xxx...
```

```
9025 DATA        ....xxx.xxx...
9026 DATA        ....x.x.xxx...
9027 DATA        ....xxx.xxx...
9028 DATA  15,   ...xxxx.xxxx..
9029 DATA        ....xxx.xxx...
9030 DATA        ....xxx.xxx...
9031 DATA        ...xxxx.xxxx..
9033 DATA        ..xxx.x.x.xxx.
9034 DATA        .xxx..x.x..xxx
9100 DATA     xxx...........
9101 DATA     .x...........
9102 DATA     .x...........
9103 DATA     xxxxxxxxxxxxx
9104 DATA     x...........x
9105 DATA     x..x... x....x
9106 DATA     x.x.x..x.x...x
9107 DATA     x..x....x....x
9108 DATA     x...........x
9109 DATA     xxxxxxxxxxxxx
9200 DATA        ... xxx.....
9201 DATA        ...x...x....
9202 DATA        ....xxx.....
9203 DATA        ...x.x.x....
9204 DATA        ..xxxxxx....
9205 DATA        ...xx.xx....
9206 DATA        ...xxxxx....
9207 DATA        .....x......
9208 DATA   10,....xxx.....
9209 DATA        ..x.xxx.x....
9210 DATA        .xxx.x.xxx....
9212 DATA        .xxxx.xxxxxx..
9213 DATA        .xxxxxxxxxxx.
9214 DATA        xxx.xxxxx.xxx.
9215 DATA        xx.xxxxxxx..xx
9216 DATA        xx.xxxxxxxx.xx
9217 DATA        x.xxxxxxxxx..x
9218 DATA        xxxxxxxxxxx.xxx
9219 DATA 11,    ..xxx.xxx...xxx
9220 DATA        ..xxx.xxx...xxx
9221 DATA       .xxxx.xxxxx....
9222 DATA      xxx.x.x.xxx...
9300 DATA     8
9310 DATA     50,0, 50,-10, 40,-10, 30,-20, 20,-20,
             10,-10
9320 DATA   0,-10, 0,0
```



Robot Hunt

Program 13

A radioactive robot is loose in an underground building and your objective is to capture it. Since you cannot see the robot, you must figure out where it is by using a Geiger counter. The Geiger counter emits beeps at a rate that varies depending on how far you are from the robot. The nearer the robot is to you, the faster the Geiger counter beeps. The intensity of the radiation is displayed on the screen, and this also indicates how far you are from the robot.

The building is made of so many rooms that the walls (which are purple) form a 7 × 7 grid. Each wall has a yellow door through which you or the robot can pass, but there are no doors leading out of the building. You can move from room to room by using the joystick on game controller 1. Your position is marked by a blue square. All of the doors in each room are open, but you can seal them shut by moving on top of them and pressing either controller button. This will permanently shut the door so that neither you nor the robot can pass. Try to figure out where the robot is and trap it in a single room by moving through the building and shutting doors behind you, but be careful not to seal yourself in. A door that is shut is indicated in light red.

The radiation given off by the robot is lethal, and the amount you have absorbed is displayed in rems on the screen. If you have not trapped the robot by the time you have received over 50 rems, you die, and the game is over.

Technical Description

Often in games using low-resolution graphics, it is important to find the color of a position on the screen. For example, the SCRN(x,y) command on line 1244 is used to find the color of the position x,y, which is the position about to be moved upon by the player. If this color corresponds to the color of a sealed door or a wall, movement is prevented.

The distance of the player from the robot determines how fast the Geiger counter beeps and the intensity of the radiation. This distance is calculated in line 1300 by the Pythagorean theorem, which uses the general equation of $x^2 + y^2 = z^2$.

Through algebraic manipulation, the general equation is transformed into the BASIC equation d=SQR((x−rx)^2+(y−ry)^2), which will give the distance between the robot and the player. Here the variables rx and x contain the horizontal coordinates of the robot and player, respectively, and correspond to the *x* in the general equation. The variables ry and y contain the vertical coordinates of the robot and player, respectively, and correspond to the *y* in the general equation.

The sound of the Geiger counter is produced with the command PRINT CHR$(7). This causes the Adam to emit a short beep.

Important Variables

Variable	Function
bk	Color of the doors and walls.
cx,cy	Direction the robot is moving. The variable cx contains the horizontal direction with −1 indicating a leftward direction, 1 indicating a rightward direction, and 0 indicating no direction along the horizontal axis. The variable cy contains the vertical direction with −1 indicating an upward direction, 1 indicating a downward direction, and 0 indicating no direction along the vertical axis.
d	Distance of the player from the robot.

cl	Color of the position about to be moved upon by the player.
pd	Contains the latest value of PDL(5), the position of the joystick.
rm	Amount of radiation received by the player.
rx,ry	Screen coordinates of the robot; variable rx contains the horizontal coordinate and ry the vertical coordinate.
sx,sy	Direction of the robot from the player.
wn	Indicates whether the robot is trapped.
x,y	Screen coordinates of the player, where variable x contains the horizontal coordinate and variable y contains the vertical coordinate.

Program Description

Lines	Function
1-95	Standard introduction.
100-140	Main program.
1000-1055	Subroutine to enter low-resolution graphics mode, draw border, and initialize the variables.
1100-1120	Subroutine to pick a random beginning position for the robot.
1200-1260	Subroutine to move the player.
1300-1310	Subroutine to calculate the distance of the robot from the player.
1500-1540	Subroutine to beep the Geiger counter.
1600-1699	Subroutine to draw the grid (rooms of the building).
1700-2180	Subroutine to move the robot away from the player and check for sealed doors.
2200-3070	Routines to print the various messages, such as rem dosage, your loss, and your win.

```
  1 REM  (c) 1984 by Brian Sawyer
  5 TEXT
 10 pm$ = "Robot Hunt"
 15 FOR n = 1 TO 4
 20 PRINT CHR$(162)
 25 NEXT n
 30 tb = 16-INT(LEN(pm$)/2)
 35 PRINT TAB(tb); pm$
 40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
 45 PRINT " Hit Controller Button to Play"; CHR$(160)
 50 iv = 0: r = 0
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
    AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 GOSUB 1000
102 GOSUB 2200
105 GOSUB 1600
107 COLOR  = 7: PLOT x, y
110 GOSUB 1100
120 GOSUB 1900
121 GOSUB 1200
122 GOSUB 2300
125 GOSUB 1300
127 GOSUB 1500
130 GOSUB 1700
132 IF wn = 1 THEN 3000
135 IF rm > 50 THEN 2500
140 GOTO 120
1000 GR
1005 wn = 0
1010 bk = 1
1015 DIM s(39, 39)
1020 x = 20: y = 20
1023 COLOR  = bk
1025 rm = 0
1030 VLIN 0, 39 AT 0
1035 HLIN 0, 39 AT 39
1040 VLIN 39, 0 AT 39
1045 HLIN 39, 0 AT 0
1050 PLOT 39, 39
1055 RETURN
1100 rx = 2+INT(RND(1)*5)*6
1110 ry = 2+INT(RND(1)*5)*6
1115 IF rx = 20 AND ry = 20 THEN 1100
1120 RETURN
1200 REM
1205 pd = PDL(5)
```

```
1206 IF pd = 0 THEN  RETURN
1207 xc = 0
1208 yc = 0
1210 IF pd = 1 THEN  yc = -1: GOTO 1244
1220 IF pd = 2 THEN  xc = 1: GOTO 1244
1230 IF pd = 4 THEN  yc = 1: GOTO 1244
1240 IF pd = 8 THEN  xc = -1: GOTO 1244
1242 RETURN
1244 cl = SCRN(x+xc, y+yc)
1245 IF cl = bk OR cl = 11 THEN  RETURN
1246 COLOR  = s: PLOT x, y
1247 s = cl
1250 x = x+xc: y = y+yc
1255 COLOR  = 7: PLOT x, y
1260 RETURN
1300 d = SQR((x-rx)^2+(y-ry)^2)
1310 RETURN
1500 c = c+2
1510 IF c < d THEN  RETURN
1520 c = 0
1525 rm = rm+1
1530 PRINT CHR$(7);
1540 RETURN
1600 COLOR  = bk
1610 FOR n = 5 TO 35 STEP 6
1620 VLIN 0, 39 AT n
1630 NEXT n
1635 FOR n = 5 TO 38 STEP 6
1640 HLIN 0, 39 AT n
1650 NEXT n
1655 COLOR  = 8
1660 FOR n = 2 TO 39 STEP 6
1670 FOR m = 2 TO 38
1672 IF SCRN(n, m) > 0 THEN  PLOT n, m
1674 NEXT m
1680 NEXT n
1685 FOR n = 2 TO 39 STEP 6
1690 FOR m = 2 TO 38
1692 IF SCRN(m, n) > 0 THEN  PLOT m, n
1694 NEXT m
1695 NEXT n
1699 RETURN
1700 REM   Move robot; check for sealed wall.
1705 cx = 0: cy = 0
1707 sx = SGN(x-rx)
1708 IF sx = 0 THEN  sx = 1
1710 sy = SGN(y-ry)
1712 IF sy = 0 THEN  sy = 1
1714 cy = -sy
1715 GOSUB 2100
1720 IF ok = 1 THEN 1780
1725 cx = -sx
1730 GOSUB 2100
```

```
1735 IF ok = 1 THEN 1780
1740 cx = sx
1745 GOSUB 2100
1750 IF ok = 1 THEN 1780
1755 cy = sy
1760 GOSUB 2100
1765 IF ok = 1 THEN 1780
1770 wn = 1: RETURN
1780 rx = rx+cx: ry = ry+cy
1785 RETURN
1790 PLOT rx, ry: RETURN
1800 cx = INT(RND(1)*3)-1
1810 cy = INT(RND(1)*3)-1
1820 RETURN
1900 IF PDL(7) = 0 AND PDL(9) = 0 THEN  RETURN
1910 IF s = 8 THEN  s = 11
1920 RETURN
2100 ok = 0
2105 cx = cx*6: cy = cy*6
2110 IF cx+rx < 0 OR cx+rx > 39 THEN 2170
2115 IF cy+ry < 0 OR cy+ry > 39 THEN 2170
2120 IF SCRN(rx+cx/2, ry+cy/2) < 11 THEN  ok = 1: RETURN
2170 cx = 0: cy = 0
2180 RETURN
2200 HOME
2210 PRINT " REMS DOSAGE:    INTENSITY:"
2215 PRINT CHR$(160); CHR$(160);
2220 RETURN
2300 HTAB 14
2310 PRINT CHR$(160); rm; "  "; CHR$(160);
2320 HTAB 29
2330 PRINT 60-INT(d); " "; CHR$(160);
2340 RETURN
2500 HOME
2520 PRINT "     Too Much Radiation!"
2530 PRINT "        *** You Lose ***"
2540 GOTO 3030
3000 HOME
3020 PRINT "        *** You Win! ***"
3030 PRINT "Would you like to continue?"
3040 GET a$
3050 IF a$ = "y" OR a$ = "Y" THEN 95
3060 TEXT
3070 END
```

World Conquest

Program 14

World Conquest is an exciting game of military strategy. Your rival is the Adam, and your troops must battle the Adam's to dominate the world.

In this game the world is divided into four continents, which are displayed on the screen: North America, South America, Africa, and Asia. Each continent is divided into a specific number of regions. Each region can hold one to ten armies, which belong either to you or the Adam. In each region there is a colored block whose size depends on the number of armies in the region. The orange blocks are your armies, and the blue blocks are the Adam's armies. The game begins with the armies and the regions divided evenly between you and the Adam. The size of each region and the ownership of the regions are determined randomly.

A flashing cursor indicates your position on the screen. Use the joystick to move the cursor from region to region. You and the Adam will alternate turns; each player can either attack his opponent or move troops. These maneuvers can only be performed between adjacent regions. You can test whether two regions are adjacent by moving the cursor from one region to another. If this can be done in one move, the regions are adjacent. Some regions are considered adjacent even though they do not touch each other.

As you move the cursor over your regions, the number of troops in that region will be indicated on the lower-right corner of the screen. The number of troops in regions the Adam owns is proprietary information, so when the cursor passes those regions, nothing is displayed.

You have the first turn. Select your first maneuver by pressing the left controller button to move troops or the right controller button to attack. You will then be asked which region you wish to attack or move from. You respond to this question by placing the cursor on the chosen region and pressing the right controller button. Now move the cursor to the region you are attacking or moving to, and press the right controller button again. If you are moving troops, you will be asked how many troops you wish to move. Respond to this question by pressing the number on the keypad. If you make a mistake while pressing the number, use the left controller button as a backspace key.

If you have chosen to attack, a battle will follow between the regions. Whoever has more armies has the advantage in the battle, but will not necessarily win. If you lose all but one army, you must retreat. If the Adam's armies are all lost, you win the region. You will be asked how many armies you want to move into the conquered region. Respond to this question by pressing the number on the keypad.

Now the Adam attacks you from one of the regions it owns. The same rules of battle apply: the Adam must retreat if it loses all but one army, and it wins when all of your armies are defeated.

During each turn, you are given bonus armies if you own all the regions of a continent. The number of bonus armies awarded is determined by which continent you possess. There is one bonus army for Africa and South America, two for North America, and three for Asia. The same bonus rule applies to the Adam.

Whoever can conquer all of the regions of the world wins the game.

Technical Description

This high-resolution game uses five shape tables: one for each of the four continents, and one to draw the armies. The shape table for each continent is first loaded into memory from the data statements. In line 4025 each continent is drawn on the screen by using the DRAW 1 at x,y command in a color set by the HCOLOR command. Each shape is drawn twice its normal size, since the command SCALE=2 has been given in line 4000. Because of the large amount of memory needed for each shape table, only one is loaded into Adam's memory at a time. After each is drawn, the next shape table replaces the previous one in memory.

Arrays are used to represent the number of armies and the ownership of regions. The array p( ) contains 21 elements, one for each region. If a region is owned by the player, the element equals 0; if it is owned by the Adam, it equals 1. The array s( ) holds the number of armies in each region.

Important Variables

Variable	Function
ab	Number of bonus armies for the Adam.
ac	The attacking region.
dc	The defending region.
er	Flag for an illegal move.
p(*number*)	Array to store the ownership of each region. The size of *number* reflects the total number of regions, which is 21. Each entry in array p corresponds to a region in the world. A value of 0 in an entry indicates that the player owns the region, and a value of 1 indicates that the Adam owns the region.
m	The number of armies moved into a conquered region.
mf	Region an army movement is being made from.
mt	Region an army is being moved to.
pb	Number of bonus armies given to the player.
s(*number*)	Array to store the number of armies in each region. The size of *number* reflects the total number of regions in the world, which is 21. Each entry in array s corresponds to a region in the world.
tr	The number of troops lost by the defending region.
ym	Indicates if a player is moving.
x(*number*), y(*number*)	Arrays containing the coordinates of the regions on the screen. The size of *number* equals the number of regions, which is 21.
u(*number*), d(*number*), l(*number*), r(*number*)	Arrays containing the regions that are adjacent to each region. The size of *number* equals the number of regions, which is 21.

Program Description

Lines	Function
0	Shape table memory pointer.
1-95	Standard introduction.
100-205	Main program.
207-256	Player's turn.
260-350	The Adam's turn.
400-500	Routine to move troops.
700-870	Subroutine to wage combat between two regions.
900-970	Message to inform you whether you won or lost the game.
1000-1050	Subroutine to enter high-resolution mode.
1400-1460	Subroutine to allow player to input numbers on the controller keypad.
2400-2450	Subroutine to draw the four continents.
2600-2710	Subroutine to initialize arrays and read in the table of adjacent regions from data.
2800-2890	Subroutine to set up starting ownership of regions.
3000-3650	Subroutine to allow a player to enter a move on the joystick while checking for mistakes.
3700-3830	Subroutine to inform you whether you won or lost the battle.
4000-4040	Subroutine to draw a continent on the screen at position x,y.
6000-6340	Subroutine to calculate bonuses for continent ownership.
7000-7090	Subroutine to allow the joystick to move cursor from region to region.
7100-7160	Subroutine to flash the cursor.
7200-7240	Subroutine to print the number of armies in a region.
7300-7320	Subroutine to redraw the armies.

7400-7430	Subroutine to draw the armies on the board.
7500-7530	Subroutine to draw armies on a region.
8000-8130	Subroutine to load the shape table into memory.
9010-9700	Shape table data for the continents.
10000-10410	Table of adjacent regions.

```
 0 HIMEM :41000
 1 REM  (c) 1984 by Brian Sawyer
 5 TEXT
10 pm$ = "World Conquest"
15 FOR n = 1 TO 4
20 PRINT CHR$(162)
25 NEXT n
30 tb = 16-INT(LEN(pm$)/2)
35 PRINT TAB(tb); pm$
40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
45 PRINT " Hit Controller Button To Play"; CHR$(160)
50 iv = 0: r = 0
55 NORMAL: IF iv = 1 THEN  INVERSE
60 PRINT " Hit"; CHR$(160)
65 c = 0
70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
   AND (c < 40) THEN 70
80 IF c = 40 THEN  iv = 1-iv: GOTO 55
90 r = RND(-r)
92 NORMAL
95 CLEAR
100 GOSUB 1000
130 GOSUB 2400
137 GOSUB 2600
160 GOSUB 2800
170 GOSUB 7400
180 ln = 1
200 GOSUB 6000
202 IF ab = 8 THEN 950
205 ct = 0: n = ln
207 GOSUB 3000
208 IF ym = 1 THEN 400
210 GOSUB 3100
220 IF er = 1 THEN 207
230 GOSUB 3200
240 IF er = 1 THEN 207
250 GOSUB 700
255 n = ac: GOSUB 7500
256 n = dc: GOSUB 7500
```

```
260 HOME: PRINT " adam's turn"
262 GOSUB 6000
264 IF pb = 8 THEN 900
265 d = -999: ct = 1
270 FOR n = 1 TO 21
275 IF p(n) = 0 THEN 330
280 FOR m = 1 TO 21
290 IF p(m) = 1 THEN 310
295 IF m <> u(n) AND m <> d(n) AND m <> l(n)
    AND m <> r(n) THEN 310
300 IF s(n)-s(m) > d THEN  d = s(n)-s(m): ac = m: dc = n
310 NEXT m
330 NEXT n
335 HOME: PRINT " adam attacks!"
336 GOSUB 7100: GOSUB 7100
340 GOSUB 700
345 n = ac: GOSUB 7500
346 n = dc: GOSUB 7500
350 GOTO 200
400 REM move troops
410 GOSUB 3300
412 IF er = 1 THEN 207
415 GOSUB 3400
417 IF er = 1 THEN 207
435 GOSUB 3500
450 s(mf) = s(mf)-t: s(mt) = s(mt)+t:
    IF s(mt) > ma THEN  s(mt) = ma
455 n = mf: GOSUB 7500
456 n = mt: GOSUB 7500
460 GOTO 260
500 END
700 REM attack from ac to dc
705 IF ct = 1 THEN  s(ac) = s(ac)+ab: GOTO 710
706 s(ac) = s(ac)+pb
710 GOSUB 7100
730 IF s(ac)+RND(1)*s(dc) > s(dc)+RND(1)*s(ac) THEN 745
735 s(ac) = s(ac)-1: GOTO 750
745 s(dc) = s(dc)-1
750 IF s(ac) <= 0 THEN 800
760 IF s(dc) <= 0 THEN 850
770 n = ac: GOSUB 7500
780 n = dc: GOSUB 7500
790 GOTO 710
800 GOSUB 3800
805 IF ct = 0 THEN  s(ac) = 1: RETURN
810 p(ac) = 1
825 tr = INT(s(dc)/2)
826 s(dc) = s(dc)-tr: s(ac) = tr
830 RETURN
850 GOSUB 3700
855 IF ct = 1 THEN  s(dc) = 1: RETURN
860 p(dc) = 0: s(dc) = 0
862 GOSUB 3600
```

```
867 s(dc) = s(dc)+i: s(ac) = s(ac)-i
870 RETURN
900 HOME
910 PRINT " you win the game!"
920 END
950 HOME
960 PRINT " adam wins the game!"
970 END
1000 HGR
1010 HCOLOR  = 3
1020 ROT  = 0: SCALE  = 1
1025 ma = 13: an = 1
1050 RETURN
1400 REM  Check paddles
1410 k = 1
1415 pd = PDL(13)
1420 p1 = PDL(7)
1422 p2 = PDL(9)
1423 IF p2 = 1 AND k > 1 THEN 1450
1425 IF p1 = 1 AND k > 1 THEN  k = k-1: PRINT CHR$(163);
     " "; CHR$(163);
1426 IF p1 = 1 AND PDL(7) = 1 THEN 1426
1430 IF pd > 9 OR k > 9 THEN 1415
1435 PRINT pd;
1440 i(k) = pd
1442 k = k+1
1443 IF PDL(13) <> 15 THEN 1443
1445 GOTO 1415
1450 no = 0: q = 0: FOR k1 = k-1 TO 1 STEP -1
1455 no = no+i(k1)*10^q: q = q+1
1460 NEXT: PRINT: RETURN
2400 cl = 12
2410 FOR m = 1 TO 5
2420 GOSUB 8000
2422 HCOLOR  = cl: cl = cl+1: IF cl = 16 THEN  cl = 15
2425 x = 40: y = 4: IF m = 2 THEN  y = 92: x = 55
2426 IF m = 3 THEN  x = 120: y = 32
2427 IF m = 5 THEN 2440
2428 IF m = 4 THEN  x = 119: y = 98
2430 GOSUB 4000
2440 NEXT m
2450 RETURN
2600 REM read in data
2610 DIM x(21), y(21), u(21), d(21), l(21), r(21)
2615 DIM i(10)
2620 FOR n = 1 TO 21
2670 READ a: IF n <> a THEN  STOP
2680 READ x(n), y(n)
2690 READ u(n), d(n), l(n), r(n)
2700 NEXT n
2710 RETURN
2800 DIM p(21), s(21)
```

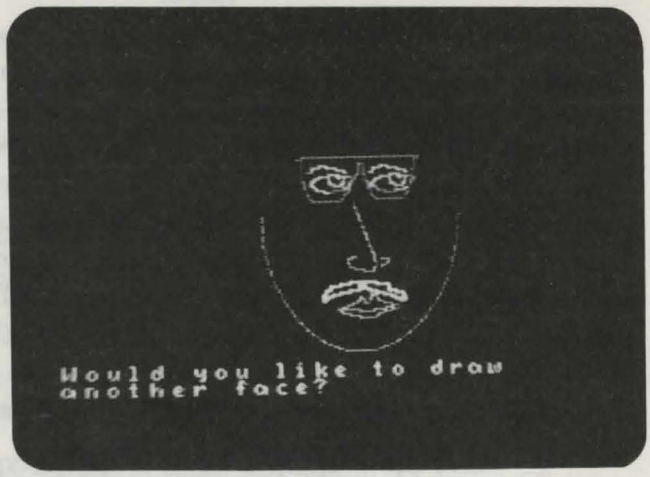
```
2810 FOR n = 1 TO 21
2815 p(n) = 1: NEXT n
2820 FOR n = 1 TO 10
2830 i = INT(RND(1)*21+1)
2835 IF p(i) = 0 THEN 2830
2840 p(i) = 0
2845 NEXT n: m1 = 1: m2 = 21
2850 FOR n = 1 TO 10
2855 FOR n1 = m1 TO 21
2860 IF p(n1) = 1 THEN  NEXT n1: END
2862 m1 = n1+1
2865 FOR n2 = m2 TO 1 STEP -1
2870 IF p(n2) = 0 THEN  NEXT n2: END
2880 m2 = n2-1
2885 s(n2) = INT(RND(1)*6+3): s(n1) = s(n2)
2886 NEXT n
2887 FOR n = 1 TO 21: IF s(n) = 0 THEN  s(n) = 3
2888 NEXT n
2890 RETURN
3000 HOME
3005 ym = 0
3010 PRINT " move or attack ? "
3020 PRINT " (hit left or right button)": PRINT
3030 GOSUB 7000
3040 IF pm = 1 THEN  ym = 1
3045 IF PDL(7) = 1 OR PDL(9) = 1 THEN 3045
3050 RETURN
3100 HOME: er = 0
3105 PRINT " attack from what country ?"
3110 PRINT " (go to region, hit rt. button)"
3120 GOSUB 7000
3130 IF p(n) = 1 THEN  HOME: PRINT " you don't own that!":
     GOSUB 5000: er = 1:RETURN
3140 ac = n
3150 RETURN
3200 HOME: er = 0
3205 PRINT " attack what country ? "
3210 PRINT " (go to region, hit rt. button)"
3218 IF PDL(9) = 1 THEN 3218
3220 GOSUB 7000
3230 IF p(n) = 0 THEN  HOME: PRINT " you own that!":
     GOSUB 5000: er = 1: RETURN
3240 dc = n
3250 IF ac <> u(n) AND ac <> d(n) AND ac <> l(n) AND
     ac <> r(n) THEN 3270
3260 RETURN
3270 HOME
3275 PRINT " not adjacent!": GOSUB 5000: er = 1: RETURN
3300 HOME: er = 0
3305 PRINT " move from what country?"
3310 PRINT " (go to region, hit rt. button)"
3320 GOSUB 7000
```

```
3330 IF p(n) = 1 THEN  HOME: PRINT " you don't own that!":
     GOSUB 5000: er = 1: RETURN
3340 mf = n
3350 RETURN
3400 HOME: er = 0
3405 PRINT " move to what country? "
3410 PRINT " (go to region, hit rt. button)"
3418 IF PDL(9) = 1 THEN 3418
3420 GOSUB 7000
3430 IF p(n) = 1 THEN  HOME: PRINT " you don't own that!":
     GOSUB 5000: er = 1: RETURN
3440 mt = n
3450 IF mf <> u(n) AND mf <> d(n) AND mf <> l(n) AND
     mf <> r(n) THEN 3470
3460 RETURN
3470 HOME
3475 PRINT " not adjacent!": GOSUB 5000: er = 1: RETURN
3500 HOME
3505 PRINT " move how many armies?"
3510 PRINT " (0-"; s(mf)-1; ") ? ";
3520 GOSUB 1400
3530 IF no > s(mf)-1 THEN  HOME: PRINT "too many":
     GOSUB 5000: GOTO 3500
3540 t = no
3545 HOME
3550 RETURN
3560 PRINT " (0-"; s(mf)-1; ") ? ";
3600 HOME
3606 PRINT " move how many armies?"
3610 PRINT " (1-"; s(ac)-1; ") ? ";
3620 GOSUB 1400
3630 IF no > s(ac)-1 THEN  HOME: PRINT "too many":
     GOSUB 5000: GOTO 3600
3635 IF no = 0 THEN  HOME: PRINT " not enough!":
     GOSUB 5000: GOTO 3600
3640 i = no
3645 HOME
3650 RETURN
3700 HOME
3710 PRINT " you win the battle!"
3720 GOSUB 5000
3730 RETURN
3800 HOME
3810 PRINT " adam wins the battle!"
3820 GOSUB 5000
3830 RETURN
4000 SCALE  = 2
4025 DRAW 1 AT x, y
4040 RETURN
5000 FOR dl = 1 TO 2400
5010 NEXT dl
5020 RETURN
```

```
6000 pb = 1: ab = 1
6005 pf = 1: af = 1
6010 FOR m = 1 TO 5
6015 IF p(m) = 0 THEN  af = 0
6020 IF p(m) = 1 THEN  pf = 0
6025 NEXT m
6030 pb = pb+pf*2: ab = ab+af*2
6105 pf = 1: af = 1
6110 FOR m = 6 TO 8
6115 IF p(m) = 0 THEN  af = 0
6120 IF p(m) = 1 THEN  pf = 0
6125 NEXT m
6130 pb = pb+pf: ab = ab+af
6205 pf = 1: af = 1
6210 FOR m = 9 TO 12
6215 IF p(m) = 0 THEN  af = 0
6220 IF p(m) = 1 THEN  pf = 0
6225 NEXT m
6230 pb = pb+pf: ab = ab+af
6305 pf = 1: af = 1
6310 FOR m = 13 TO 21
6315 IF p(m) = 0 THEN  af = 0
6320 IF p(m) = 1 THEN  pf = 0
6325 NEXT m
6330 pb = pb+pf*3: ab = ab+af*3
6340 RETURN
7000 SCALE  = 1
7002 n = an
7005 GOSUB 7200
7010 pd = PDL(5): IF pd = 0 AND PDL(7) = 0 AND PDL(9) = 0
     THEN  GOSUB 7300: GOTO 7010
7030 pm = PDL(7): pn = PDL(9)
7035 IF pm = 1 OR pn = 1 THEN  PRINT CHR$(7); : RETURN
7040 IF pd = 1 THEN  i = u(n): IF i = 0 THEN 7010
7050 IF pd = 2 THEN  i = r(n): IF i = 0 THEN 7010
7060 IF pd = 4 THEN  i = d(n): IF i = 0 THEN 7010
7070 IF pd = 8 THEN  i = l(n): IF i = 0 THEN 7010
7071 an = i
7072 n = i: GOSUB 7300
7090 GOTO 7005
7100 HCOLOR  = 3
7110 DRAW 1 AT x(ac)+4, y(ac)+8
7120 DRAW 1 AT x(dc)+4, y(dc)+8
7130 FOR u = 1 TO 600: NEXT u
7140 XDRAW 1 AT x(ac)+4, y(ac)+8
7150 XDRAW 1 AT x(dc)+4, y(dc)+8
7155 FOR u = 1 TO 600: NEXT u
7160 RETURN
7200 PRINT CHR$(160); "                    ": PRINT CHR$(160);
7215 IF p(n) = 1 THEN  PRINT " adam's region": RETURN
7230 PRINT " armies in region:"; s(n)
7240 RETURN
7300 HCOLOR  = 3: GOSUB 7501
```

```
7310 GOSUB 7500
7320 RETURN
7400 FOR n = 1 TO 21
7410 GOSUB 7500
7420 NEXT n
7430 RETURN
7500 SCALE  = 1: HCOLOR  = 1+p(n)+4
7501 IF s(n) > ma THEN  s(n) = ma
7502 FOR m = 1 TO ma
7504 XDRAW 1 AT x(n), y(n)+m
7505 NEXT m
7506 FOR m = 1 TO s(n)
7510 DRAW 1 AT x(n), y(n)+m
7520 NEXT m
7530 RETURN
8000 qs = 1: pt = 16766: tl = 41000
8005 of = qs*2+3
8010 by = tl: GOSUB 8100
8015 POKE pt, lo: POKE pt+1, hi
8020 POKE tl, qs
8030 FOR n = 1 TO qs
8040 by = of: GOSUB 8100
8050 POKE tl+n*2, lo: POKE tl+n*2+1, hi
8060 READ a: POKE tl+of, a: of = of+1
8065 IF a <> 0 THEN 8060
8070 NEXT n
8080 RETURN
8100 hi = INT(by/256): lo = by-hi*256
8120 RETURN
9010 DATA 44,37,45,46,53,45,53,53,53,53,53
9020 DATA 45,44,44,37,44,46,46,62,62,62,62
9030 DATA 54,55,46,54,46,54,45,36,37,36,45
9040 DATA 46,54,53,45,62,62,54,55,62,54,63
9050 DATA 62,55,63,62,54,55,62,55,62,55,53
9060 DATA 46,54,54,39,39,60,36,63,60,62,55
9070 DATA 55,55,46,46,54,53,54,39,39,63,60
9080 DATA 36,39,60,36,39,60,36,39,60,36,60
9090 DATA 36,36,60,36,36,37,36,37,36,36,39
9100 DATA 63,36,39,36,44,36,44,45,36,45,46
9110 DATA 46,46,0
9210 DATA 53,45,46,53,53,45,46,54,62,54,55
9220 DATA 62,54,62,62,62,54,54,55,55,55,62
9230 DATA 62,54,55,47,46,53,62,63,63,36,36
9240 DATA 60,36,39,36,44,36,36,37,60,60,36
9250 DATA 39,36,37,36,44,44,37,44,36,45,45,0
9310 DATA 36,37,44,37,45,45,46,54,55,55,45
9320 DATA 46,37,37,37,36,37,45,36,45,45,37
9330 DATA 37,45,44,36,44,44,53,54,46,46,54
9340 DATA 46,37,44,44,45,37,37,44,45,37,45
9350 DATA 37,45,45,37,45,45,45,44,45,46,54
9360 DATA 62,54,54,62,54,54,54,53,54,54,55
9370 DATA 54,54,39,39,52,55,54,53,54,53,54
9380 DATA 53,62,62,54,55,55,55,63,62,62,54
```

```
 9390 DATA 62,54,63,60,36,60,55,55,55,55,55
 9400 DATA 39,39,39,60,36,60,55,62,63,62,55
 9410 DATA 55,53,53,54,55,55,63,60,60,60,60
 9420 DATA 39,36,39,36,39,37,37,63,62,63,54
 9430 DATA 55,54,39,47,60,63,54,53,53,63,39
 9440 DATA 39,60,55,62,55,63,38,39,36,63,36
 9450 DATA 37,44,44,44,44,36,44,36,37,36,37
 9460 DATA 36,60,39,60,62,62,54,39,60,60,39
 9470 DATA 36,45,36,44,36,0
 9510 DATA 53,45,37,45,53,45,45,53,45,36,45
 9520 DATA 37,53,53,53,53,53,53,45,62,62,54
 9530 DATA 63,62,54,62,54,54,53,62,54,23,52
 9540 DATA 62,62,55,62,54,62,62,63,60,39,39
 9550 DATA 60,36,60,36,36,36,39,39,63,63,39
 9560 DATA 63,39,60,36,36,36,44,37,44,36,45
 9570 DATA 37,0
 9700 DATA 45,45,0
10000 DATA 1,31,14
10010 DATA 0,2,19,0
10020 DATA 2,38,39
10030 DATA 1,3,0,5
10040 DATA 3,40,62
10050 DATA 2,6,0,4
10060 DATA 4,55,58
10070 DATA 5,0,3,0
10080 DATA 5,58,36
10090 DATA 0,4,2,13
10100 DATA 6,49,94
10110 DATA 3,7,0,0
10120 DATA 7,49,120
10130 DATA 6,8,0,9
10140 DATA 8,42,134
10150 DATA 7,0,0,0
10160 DATA 9,114,112
10170 DATA 0 ,0,7,10
10180 DATA 10,138,110
10190 DATA 15,11,9,0
10200 DATA 11,135,132
10210 DATA 10,12,0,0
10220 DATA 12,140,145
10230 DATA 11,0,0,0
10240 DATA 13,122,35
10250 DATA 0,14,5,16
10260 DATA 14,135,58
10270 DATA 13,15,0,17
10280 DATA 15,125,79
10290 DATA 14,10,0,18
10300 DATA 16,161,30
10310 DATA 0,17,13,19
10320 DATA 17,162,53
10330 DATA 16,18,14,20
10340 DATA 18,162,82
10350 DATA 17,0,15,21
10360 DATA 19,214,2
10370 DATA  0,20,16
10380 DATA 20,209,5
10390 DATA 19,21,17
10400 DATA 21,209,7
10410 DATA 20,0,18,
```



Face Designer

Program 15

Face Designer puts you in charge of constructing a unique face on your Adam. This is done by choosing from various facial features presented to you by the program.

You are first shown a gallery of six distinctly different noses. These noses are not only to be admired—you must select one for the face. To select a nose, press the number on the keypad of the game controller 1 corresponding to the nose you desire. The noses, as well as all the other facial parts, are numbered from left to right.

Now select one out of four pairs of eyes using the same process. Mouths are next, and you have five to choose from. You may also choose one of two mustaches for your face or none at all. If the person you are drawing is nearsighted, eyeglasses may be used. Finally, select the shape of the head. You have three different sizes to choose from.

Now that you have chosen all of the features for your customized face, it will be drawn on the screen.

Technical Description

This game makes use of shape tables on the high-resolution screen to draw the facial features. Since many shapes are used, only one shape is held in memory at a time. When all the selections have been made and it is time to draw the face, each shape is individually loaded into memory and drawn, then the next is loaded in place of it, and so on.

The HCOLOR command sets the color of the shape that is to be drawn, and the DRAW r AT x,y command draws it at position x,y, where x contains the horizontal coordinate and y contains the vertical coordinate. Here r contains the shape to be drawn. For example, if r equals 1, the shape table for the eyes is drawn.

The keypad number pressed by the player is contained in PDL(13). If its value is 15, no number has yet been selected.

The variable qs in line 8000 is the number of shapes to be loaded into memory at one time. In this program, qs is set to 1.

Important Variables

Variable	Function
ey	Type of eyes chosen by the player.
fs	Shape of face selected.
gf	Indicates if glasses were selected.
mo	Type of mouth chosen.
mu	Type of mustache chosen.
no	Type of nose chosen.
pd	Contains the value of PDL(13), the number pressed on the keypad on game controller 1.
r	The number of the shape being currently drawn.
x,y	Screen position of the shape currently being drawn; variable x contains the horizontal coordinate and variable y the vertical coordinate.

Program Description

Lines	Function
0	Shape table memory pointer.
1-95	Standard introduction.
100-180	Main program.

1000-1050	Subroutine to enter high-resolution mode.
1200-1280	Subroutine to choose type of nose.
1300-1380	Choose type of eyes.
1400-1480	Choose type of mouth.
1500-1580	Choose type of mustache.
1600-1640	Choose glasses.
1700-1760	Choose the type of head.
1800-1880	Subroutine to draw the chosen facial features.
1900-1930	Draw chosen head.
2600-2670	Load in shape from data.
4000-4020	Subroutine to draw the shapes.
8000-8130	Subroutine to load the shapes.
9553-18050	Shape table data.

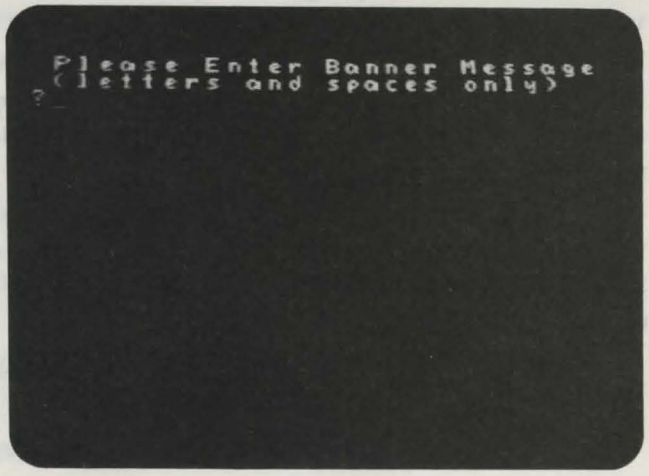
```
 0 HIMEM :41000
 1 REM  (c) 1984 by Brian Sawyer
 5 TEXT
10 pm$ = "Face Designer"
15 FOR n = 1 TO 4
20 PRINT CHR$(162)
25 NEXT n
30 tb = 16-INT(LEN(pm$)/2)
35 PRINT TAB(tb); pm$
40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
45 PRINT " Hit Controller Button to Play"; CHR$(160)
50 iv = 0: r = 0
55 NORMAL: IF iv = 1 THEN  INVERSE
60 PRINT " Hit"; CHR$(160)
65 c = 0
70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
   AND (c < 40) THEN 70
80 IF c = 40 THEN  iv = 1-iv: GOTO 55
90 r = RND(-r)
92 NORMAL
95 CLEAR
100 GOSUB 1000
130 GOSUB 1200
131 GOSUB 1800
140 PRINT "Would you like to draw another face?"
150 GET a$
160 IF a$ = "y" OR a$ = "Y" THEN 95
170 TEXT
180 END
```

```
1000 HGR
1010 HCOLOR  = 3
1020 ma = 13
1030 ROT  = 0: SCALE  = 1
1050 RETURN
1200 HOME
1202 PRINT " select type of nose: (1-6)"
1205 RESTORE
1210 FOR r = 1 TO 6
1220 GOSUB 8000
1230 x = 10+r*34: y = 80
1240 GOSUB 4000
1250 NEXT r
1260 pd = PDL(13): IF pd = 15 THEN 1260
1270 IF pd > 6 OR pd = 0 THEN 1260
1280 no = pd
1300 HOME
1301 HGR: HCOLOR  = 1
1302 PRINT " select type of eyes: (1-4)"
1310 FOR r = 1 TO 4
1320 GOSUB 8000
1330 x = 10+r*34: y = 80
1340 GOSUB 4000
1350 NEXT r
1360 pd = PDL(13): IF pd = 15 THEN 1360
1370 IF pd > 4 OR pd = 0 THEN 1360
1380 ey = pd+6
1400 HOME
1401 HGR: HCOLOR  = 1
1402 PRINT " select type of mouth: (1-5)"
1410 FOR r = 1 TO 5
1420 GOSUB 8000
1430 x = 10+r*34: y = 80
1440 GOSUB 4000
1450 NEXT r
1460 pd = PDL(13): IF pd = 15 THEN 1460
1470 IF pd > 5 OR pd = 0 THEN 1460
1480 mo = pd+10
1500 HOME
1501 HGR: HCOLOR  = 1
1502 PRINT " select type of mustache"
1503 PRINT " (1-2) or (3-none)"
1510 FOR r = 1 TO 2
1520 GOSUB 8000
1530 x = 10+r*34: y = 80
1540 GOSUB 4000
1550 NEXT r
1560 pd = PDL(13): IF pd = 15 THEN 1560
1570 IF pd > 3 OR pd = 0 THEN 1560
1580 mu = pd+15
1600 HOME
1605 HGR: HCOLOR  = 1
1607 GOSUB 8000: x = 100: y = 80
1608 GOSUB 4000
```

```
1610 PRINT " do you want glasses?"
1620 PRINT " (1=yes,2=no)"
1630 pd = PDL(13): IF pd = 15 THEN 1630
1640 gf = 0: IF pd = 1 THEN  gf = 1
1700 HOME
1705 HGR
1706 FOR dl = 1 TO 500: NEXT dl
1710 PRINT " choose shape of head:"
1720 PRINT " (1-fat, 2-medium, 3-thin)"
1730 pd = PDL(13): IF pd = 15 THEN 1730
1740 IF pd = 0 OR pd > 3 THEN 1700
1750 fs = 25+5*(4-pd)
1760 RETURN
1800 HGR: HOME: HCOLOR  = 1
1810 r = ey: GOSUB 2600
1812 HCOLOR  = 14
1815 x = 110: y = 70: GOSUB 4000
1817 x = 135: y = 70: GOSUB 4000
1820 r = no: GOSUB 2600
1822 HCOLOR  = 5
1825 x = 129: y = 83: IF no > 4 THEN  x = 134
1826 GOSUB 4000
1830 r = mo: GOSUB 2600
1832 HCOLOR  = 15
1835 x = 122: y = 134: IF mo = 15 THEN  x = 118
1836 GOSUB 4000
1838 IF mu = 18 THEN 1846
1840 r = mu: GOSUB 2600
1842 i = 0: IF r = 16 THEN  i = 10
1844 HCOLOR  = 3
1845 x = 122+i: y = 124: GOSUB 4000
1846 IF gf = 0 THEN 1860
1850 r = 18: GOSUB 2600
1852 HCOLOR  = 9
1855 x = 109: y = 63: GOSUB 4000
1860 GOSUB 1900
1880 RETURN
1900 HCOLOR  = 15
1905 FOR ci = 0 TO 3.14159 STEP .02
1910 HPLOT 132+COS(ci)*fs, 90+SIN(ci)*65
1920 NEXT ci
1930 RETURN
2600 RESTORE
2610 FOR m = 1 TO r
2620 GOSUB 8000
2630 NEXT m
2670 RETURN
4000 SCALE  = 1
4010 DRAW 1 AT x, y
4020 RETURN
8000 qs = 1: pt = 16766: tl = 41000
8005 of = qs*2+3
```

```
8010 by = tl: GOSUB 8100
8015 POKE pt, lo: POKE pt+1, hi
8020 POKE tl, qs
8030 FOR n = 1 TO qs
8040 by = of: GOSUB 8100
8050 POKE tl+n*2, lo: POKE tl+n*2+1, hi
8060 READ a: POKE tl+of, a: of = of+1
8065 IF a <> 0 THEN 8060
8070 NEXT n
8080 RETURN
8100 hi = INT(by/256): lo = by-hi*256
8130 RETURN
9553 DATA 54,54,54,54,54,54,54,54,54,54,55
9556 DATA 63,54,62,54,55,54,45,45,46,53,45
9559 DATA 45,37,37,45,45,60,36,60,36,63,36
9562 DATA 60,36,36,36,36,36,36,36,36,0
9588 DATA 54,54,54,54,54,54,54,54,54,54,54
9591 DATA 54,62,62,62,62,54,53,45,45,44,54
9594 DATA 54,53,45,37,37,36,36,53,46,45,45
9597 DATA 44,36,39,39,39,39,39,39,63,36,36
9600 DATA 36,36,36,36,0
9626 DATA 54,62,54,54,55,54,63,63,54,46,54
9629 DATA 46,45,46,46,37,44,54,46,46,46,45
9632 DATA 44,44,36,36,53,54,45,37,37,45,45
9635 DATA 37,36,60,60,63,63,36,39,39,60,36
9638 DATA 36,36,0
9664 DATA 54,46,54,46,54,46,54,46,54,46,54
9667 DATA 46,54,46,54,46,54,54,53,54,55,62
9670 DATA 63,39,63,63,60,39,60,36,45,44,53
9673 DATA 77,73,73,73,61,63,79,137,52,62,62
9676 DATA 55,7,0
9702 DATA 54,62,54,54,55,54,62,54,54,62,54
9705 DATA 54,55,54,62,54,54,53,54,45,45,37
9708 DATA 36,44,36,36,60,63,246,27,63,63,54
9711 DATA 54,53,45,0
9737 DATA 54,54,53,54,54,53,54,46,54,54,53
9740 DATA 54,54,53,54,54,54,55,62,54,63,63
9743 DATA 63,60,63,63,60,63,36,36,37,37,45
9746 DATA 53,45,0
9772 DATA 37,45,36,45,37,45,37,45,53,45,46
9775 DATA 53,53,55,55,63,55,63,55,63,39,63
9778 DATA 39,63,79,73,9,36,37,45,53,53,63
9781 DATA 54,63,36,45,60,55,46,45,5,0
9807 DATA 45,36,45,37,45,53,45,45,46,45,46
9810 DATA 149,63,63,60,63,55,63,63,62,55,55
9813 DATA 55,45,46,46,45,46,45,37,45,45,44
9816 DATA 37,44,44,44,39,255,219,219,51,54,45
9819 DATA 46,45,44,36,220,43,54,0
9845 DATA 36,37,44,44,44,44,36,37,45,53,45
9848 DATA 54,45,46,46,54,246,27,39,39,39,63
9851 DATA 55,55,63,62,54,46,53,46,54,45,37
9854 DATA 44,44,44,44,220,59,63,55,45,45,53
```

```
 9857 DATA  63,63,63,0
 9883 DATA  37,37,44,45,44,45,37,45,53,45,46
 9886 DATA  53,45,54,62,60,39,63,60,63,62,55
 9889 DATA  63,63,63,55,45,46,53,45,53,46,45
 9892 DATA  45,44,45,37,228,219,59,127,182,146,146
 9895 DATA  63,60,60,60,60,76,73,18,53,45,45
 9898 DATA  45,37,45,0
 9924 DATA  45,44,37,45,44,37,37,44,46,54,45
 9927 DATA  36,37,44,53,46,46,53,45,53,45,53
 9930 DATA  62,63,62,63,62,55,55,63,63,63,39
 9933 DATA  63,63,60,63,60,37,45,45,45,53,45
 9936 DATA  45,45,44,45,45,44,45,5,0
 9962 DATA  44,44,45,45,45,45,53,45,37,45,45
 9965 DATA  45,45,53,53,53,63,63,63,63,63,63
 9968 DATA  63,63,63,63,63,63,127,73,9,54,45
 9971 DATA  45,46,45,45,45,37,45,37,4,0
 9997 DATA  41,44,36,45,37,45,37,45,53,53,45
10000 DATA  45,46,45,53,109,49,62,62,222,219,219
10003 DATA  27,61,63,60,63,4,0
10029 DATA  53,45,53,45,53,45,53,45,45,37,45
10032 DATA  37,45,37,45,37,45,37,60,36,63,62
10035 DATA  62,55,55,63,55,63,63,60,60,63,60
10038 DATA  63,60,63,62,63,247,155,34,44,36,36
10041 DATA  37,44,5,0
10067 DATA  46,46,46,46,45,46,53,45,46,45,53
10070 DATA  45,45,45,44,45,44,45,44,37,45,44
10073 DATA  44,44,44,60,62,62,62,55,63,62,63
10076 DATA  55,63,63,63,55,63,36,63,63,39,63
10079 DATA  63,60,39,255,27,60,36,44,44,4,0
10105 DATA  45,53,45,53,45,53,45,53,45,46,53
10108 DATA  45,36,39,60,63,60,63,36,60,63,63
10111 DATA  60,63,63,63,54,54,63,63,62,63,55
10114 DATA  63,55,63,62,62,36,44,36,45,36,45
10117 DATA  45,44,45,37,45,45,45,54,6,0
10143 DATA  45,44,45,37,45,45,77,9,45,45,53
10146 DATA  45,45,46,5,0
10216 DATA  45,45,45,45,45,45,45,45,45,45,45
10219 DATA  45,45,45,45,45,45,45,45,45,45,45
10222 DATA  45,53,63,54,54,62,54,54,62,54,54
10225 DATA  62,62,54,63,63,63,63,63,63,60,60
10228 DATA  36,60,36,60,36,60,36,36,60,55,54
10231 DATA  54,55,62,54,55,54,55,55,55,62,62
10234 DATA  63,63,63,63,63,39,39,60,36,60,36
10237 DATA  60,36,36,36,36,36,63,45,44,0
18050 RETURN
```



Banner

Program 16

This program enables you to print banners with giant-sized letters. Insert a piece of paper into the printer so the top is sticking out about one inch. Now enter your message—it can only be letters or spaces—and it will be printed vertically on the paper as a banner. If the banner has not been completely printed, the program will pause at the end of each piece of paper so you can insert a new piece. When you have the new piece in place, press any key, and the printing will resume.

Technical Description

Each letter is contained in numerical form in data statements in lines 9500-9620. These numbers represent the pattern of dots necessary to draw each letter. This information is used in lines 1500-1650 to print the giant letters.

Each letter that is stored in array 1 consists of five numbers. Before each letter is printed, all five numbers are converted to 1's and 0's to form the letter,

and then the numbers are multiplied by sz to either enlarge or reduce the letter. You can modify the program to print taller or shorter letters by changing the value of the variable sz in line 1100. This value is currently set to 9, which causes the letters to be drawn at a height of about 1/2 of a page. Changing the value of sz to 15 will produce letters that take up a full page. Setting sz to 4 will make the letters 1/4 of a page tall. If you set sz to a value below 3, the letters will be so short and fat they will be illegible.

If you are using fan-fold paper (connected sheets of paper), there is no need for the printer to pause each time a page is printed. If you change the value of pf in line 100 from 1 to 0, the printer will print the banner without pausing at the end of each page.

Important Variables

Variable	Function
b(*number*)	Array used as a temporary storage for the letter to be printed while it is being converted into 1's and 0's. Since five numbers are used to represent each letter, the size of *number* is 5.
l(*number1*, *number2*)	Array containing the dot patterns for each of the letters. Since there are 26 letters in the alphabet, the size of *number1* is 26. Each letter is represented by 5 numbers, so the size of *number2* is 5.
ms$	The message to be printed on the banner.
p(*number*)	Array used to store the letter to be printed while modifications are being made to it. The size of *number* is 5.
sz	The size of each letter. The size of each letter in array l is multiplied by sz to give the final size of the letter before it is printed.

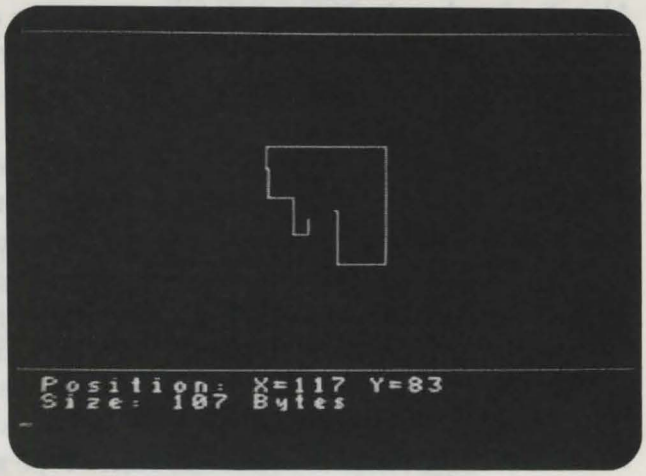
Program Description

Lines	Function
1-95	Standard introduction.
100-180	Input the banner message; main program.
1000-1010	Subroutine to dimension arrays b( ), p( ), and l.
1100-1150	Subroutine to read in each letter's dot pattern from the data.
1200-1230	Subroutine to find the next letter to print.
1500-1590	Subroutine to print a letter.
1600-1650	Subroutine to print the dots of the letter.
1700-1750	Subroutine to halt the printer while a new sheet of paper is inserted.
9500-9620	Dot patterns for each letter.

```
  1 REM (c) 1984 by Brian Sawyer
  5 TEXT
 10 pm$ = "Banner"
 15 FOR n = 1 TO 4
 20 PRINT CHR$(162)
 25 NEXT n
 30 tb = 17-INT(LEN(pm$)/2)
 35 PRINT TAB(tb); pm$
 40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
 45 PRINT " Hit Controller Button to Play"; CHR$(160)
 50 iv = 0: r = 0
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
       AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 HOME: PRINT: PRINT " Reading in Data, Please Wait"
105 GOSUB 1000
110 GOSUB 1100
115 HOME: PRINT
116 PRINT " Please Enter Banner Message "
117 PRINT " (letters and spaces only) "
```

```
 120 INPUT ms$
 125 PR #0: PRINT
 130 FOR n = 1 TO LEN(ms$)
 135 lc = lc+1: IF lc/3 = INT(lc/3) THEN  GOSUB 1700
 140 i$ = MID$(ms$, n, 1)
 141 IF i$ <> " " THEN 145
 142 FOR j = 1 TO 13: PRINT
 144 NEXT j: GOTO 170
 145 FOR j = 1 TO 5: p(j) = 0: NEXT j
 150 GOSUB 1200
 160 GOSUB 1500
 162 PRINT
 165 PRINT
 166 PRINT
 170 NEXT n
 171 PR #0
 176 GET a$
 177 IF a$ = "y" OR a$ = "Y" THEN 95
 180 END
1000 DIM b(5), p(5)
1005 DIM l(26, 5)
1010 RETURN
1100 sz = 9: pf = 1: lc = -1
1105 FOR n = 1 TO 26
1110 FOR x = 1 TO 5
1120 READ a: b = 0
1125 FOR m = 0 TO 4: a = a/2
1126 IF a <> INT(a) THEN  b = b+2^(4-m)
1127 a = INT(a): NEXT m
1128 l(n, x) = b
1130 NEXT x
1140 NEXT n
1150 RETURN
1200 i = ASC(i$)-96
1210 IF i > -32 AND i < -5 THEN  i = i+32
1220 IF i > 26 OR i < 1 THEN  i = 1
1230 RETURN
1500 FOR h = 1 TO 5: b(h) = l(i, h): NEXT h
1510 FOR h = 1 TO 5
1520 FOR s = 5 TO 1 STEP -1
1530 b(s) = b(s)/2
1550 IF b(s) = INT(b(s)) THEN 1555
1551 p(s) = 2
1555 b(s) = INT(b(s))
1570 NEXT s
1575 GOSUB 1600
1580 NEXT h
1590 RETURN
1600 FOR y = 1 TO 3
1605 FOR p = 5 TO 1 STEP -1
1610 b$ = LEFT$("                                                    ", sz)
1612 IF p(p) = 1 THEN  p(p) = 0: GOTO 1615
```

```
1613 IF p(p) = 2 THEN  GOTO 1615
1614 GOTO 1620
1615 b$ = "": FOR j = 1 TO sz: b$ = b$+CHR$(64+i): NEXT
1620 PRINT b$;
1630 NEXT p
1635 PRINT
1640 NEXT y
1645 FOR j = 1 TO 5: IF p(j) = 2 THEN  p(j) = 1
1646 NEXT j
1650 RETURN
1700 IF pf = 0 THEN  RETURN
1705 PR #0
1710 HOME: PRINT
1720 PRINT " Load Paper, then hit any key"
1730 GET g$
1740 HOME
1745 PR #1
1750 RETURN
9500 DATA  4,10,31,17,17,30,17,30,17,31
9510 DATA  15,16,16,16,15,30,17,17,17,30
9520 DATA  31,16,31,16,31,31,16,30,16,16
9530 DATA  14,16,23,18,14,17,17,31,17,17
9540 DATA  28,8,8,8,28,7,2,2,18,14
9550 DATA  18,20,24,20,18,16,16,16,16,31
9560 DATA  10,21,21,21,21,14,17,17,17,17
9570 DATA  14,17,17,17,14,30,17,30,16,16
9580 DATA  14,17,17,21,14,30,17,30,18,17
9590 DATA  14,16,14,1,14,31,4,4,4,4
9600 DATA  17,17,17,17,14,17,17,10,10,4
9610 DATA  21,21,21,21,10,17,10,4,10,17
9620 DATA  17,10,4,4,4,31,2,4,8,31,31
```



Shape Maker

Program 17

Designing high-resolution shapes on a sheet of graph paper can be tedious and can use your precious time and supplies. Shape Maker spares you this drudgery by allowing you to design shapes on the screen using your game controller. These shapes are then converted into data statements for use in your programs.

The flashing cursor in the center of the screen shows where you are drawing. You can draw a shape by moving the cursor with the joystick on game controller 1. If you wish to move the cursor without leaving a trail, hold down the left controller button while moving the cursor. If you want accuracy instead of speed in creating your design, use the keypad to move the cursor. Press 2 to move up, 8 to move down, 6 to move right, and 4 to move left.

The screen on which you are drawing your shape is made of many tiny dots. The drawing area measures 254 dots along the horizontal axis and 155 dots along the vertical axis. Initially these dots are black, but as you move the cursor, the dots change color to indicate where you have been. Encircling the drawing area is a border, which prevents the cursor from leaving the drawing area. The

coordinates that indicate the position of your cursor are displayed at the bottom of the screen; x represents the horizontal coordinate and y the vertical coordinate. Also displayed at the bottom of the screen is the current size of the shape you are drawing. This size is measured in *bytes*. A byte is a single cell in the Adam's memory that is capable of storing one character.

After you have created your shape, you'll want to save it to use in your programs. To do this, press 0 on the controller keypad and then type the letter P. The shape that you have created will be converted to data statements with line numbers. This program code will then be printed on your printer. To use this shape in other programs, you must type the program code that has been printed.

When the printer is finished, you can design additional shapes or quit the program. To make another shape, just repeat the process already described. To quit the program, press 0 on the keypad and then type the letter Q.

Before the program ends, it will print a small subroutine on the printer. This is called the *shape loader* subroutine. Type this subroutine into the computer along with the shape data statements when you enter one of your programs. Including and executing the statement GOSUB 8000 in the program will cause the shapes to be loaded into memory. The shape programs must begin with the statement HIMEM:41000 to make space in the Adam's memory for the shapes. Once the shapes have been loaded into memory, the program should enter high-resolution graphics mode with the HGR command. Now set the color with HCOLOR, and you can use the DRAW command to draw your shapes on the screen.

Technical Description

The shape tables created with this program are stored in the Adam's memory beginning at location 41000. In line 0, the statement HIMEM:49000 is used to reserve memory space for your shape tables so that the Adam will not try to store other information there.

The variable ad keeps track of the last memory location of the shape table. As you design the shape, the size of the shape table increases along with the variable ad. In line 4450, the command POKE ad,pk is used to store the latest part of your shape in the Adam's memory.

Two shape tables are used in this program: the first is the cursor, and the second contains the shape you are designing. Each time you add to the shape, the second shape table is expanded and then drawn on line 4475. The DRAW and XDRAW commands are used to draw the cursor and then erase it, thus creating the flashing effect.

Important Variables

Variable	Function
a$	Indicates the key that you pressed while choosing whether to print or quit.

ad	The memory location at the end of the shape table.
ch	Contains a number from 0 to 7 that corresponds to the direction of the latest move of the cursor.
er	Indicates if the cursor is touching the screen border.
p1	Contains the latest value of PDL(5), the position of the joystick.
p2	Contains the latest value of PDL(13), the key pressed on the keypad.
pk	Contains the latest number to be stored in the shape table memory.
x,y	Screen coordinates of the cursor. The variable x contains the horizontal coordinate and the variable y contains the vertical coordinate.
xc,yc	The direction the cursor is moving. The variable xc indicates the direction along the horizontal axis, and the variable yc indicates the direction along the vertical axis.

Program Description

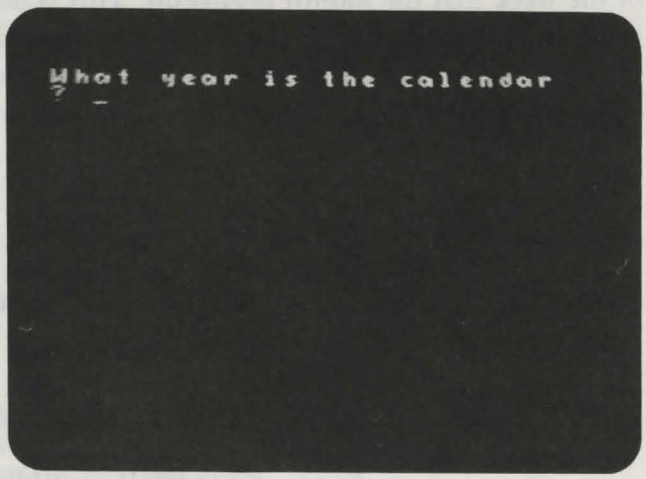
Lines	Function
0	Shape table memory pointer.
1-95	Standard introduction.
100-250	Main program.
1000-1050	Subroutine to enter the high-resolution graphics mode, draw the border, and clear the shape table memory.
2000-2020	Subroutine to print the number of bytes used to draw the shape.
4000-4050	Subroutine to get the latest cursor move.

4100-4120	Subroutine to check if the cursor is within the screen borders.
4200-4210	Subroutine to print the cursor.
4300-4310	Subroutine to erase the cursor. Used with the previous subroutine, it will simulate a flashing cursor.
4400-4620	Subroutine to find the number to store in the shape table memory corresponding to the latest move of the cursor.
5000-5110	Subroutine to print the shape table data on the printer.
5200-5240	Subroutine to print the shape loader subroutine on the printer.
8000-8130	Data containing the shape loader subroutine.
9000-9010	Shape table data.

```
  0 HIMEM :49000
  1 REM (c) 1984 by Brian Sawyer
  5 TEXT
 10 pm$ = "Shape Maker"
 15 FOR n = 1 TO 4
 20 PRINT CHR$(162)
 25 NEXT n
 30 tb = 16-INT(LEN(pm$)/2)
 35 PRINT TAB(tb); pm$
 40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
 45 PRINT " Hit Controller Button to Play"; CHR$(160)
 50 iv = 0: r = 0
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
    AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 POKE 49201, 0
102 CLEAR
105 GOSUB 1000: RESTORE: GOSUB 8000
107 nt = PEEK(sd+201): ln = 8990+nt*1000
110 ad = sd+9: ch = 0: p = 0
```

```
 115 sl = 0: GOSUB 2000
 120 GOSUB 4200
 130 GOSUB 4300
 140 p1 = PDL(5): p2 = PDL(13)
 144 IF p2 = 0 THEN 200
 145 IF p1 = 0 AND p2 = 15 THEN 120
 150 GOSUB 4000
 155 IF ch = 999 THEN 120
 160 GOSUB 4100
 165 IF er = 1 THEN 120
 170 GOSUB 4400
 175 IF ad > sd+195 THEN  PRINT " Error: Out of Room!": END
 180 sl = ad-sd-8: GOSUB 2000
 190 GOTO 120
 200 HOME
 210 PRINT " Print Data or Quit? (p or q)"
 220 INPUT a$
 230 IF a$ = "q" THEN  GOSUB 5200: TEXT: END
 250 GOSUB 5000: nt = nt+1: POKE sd+201, nt: GOTO 102
1000 HGR
1005 x = 128: y = 80
1010 HCOLOR  = 1
1015 SCALE  = 1: ROT  = 0
1020 sd = 49000
1025 FOR n = sd TO sd+200: POKE n, 0: NEXT n
1030 HPLOT 0, 0 TO 255, 0
1035 HPLOT 255, 0 TO 255, 156
1040 HPLOT 255, 156 TO 0, 156
1045 HPLOT 0, 156 TO 0, 0
1050 RETURN
2000 PRINT CHR$(160); CHR$(160);
2005 PRINT " Position: X="; x; " Y="; y
2010 PRINT " Size: "; sl; " Bytes"
2020 RETURN
4000 xc = 0: yc = 0
4005 ch = 999
4010 IF p1 = 1 OR p2 = 2 THEN  yc = -1: ch = 0
4020 IF p1 = 2 OR p2 = 6 THEN  xc = 1: ch = 1
4030 IF p1 = 4 OR p2 = 8 THEN  yc = 1: ch = 2
4040 IF p1 = 8 OR p2 = 4 THEN  xc = -1: ch = 3
4050 RETURN
4100 er = 0
4105 IF x+xc > 254 OR x+xc < 1 THEN  er = 1: RETURN
4110 IF y+yc > 155 OR y+yc < 1 THEN  er = 1: RETURN
4115 x = x+xc: y = y+yc
4120 RETURN
4200 DRAW 1 AT x, y
4205 FOR t = 1 TO 10: NEXT t
4210 RETURN
4300 XDRAW 1 AT x, y
4305 FOR t = 1 TO 10: NEXT t
4310 RETURN
```

```
4400 IF PDL(7) = 0 AND PDL(9) = 0 THEN  ch = ch+4
4410 IF ch > 3 AND p = 2 THEN  p = 0: ad = ad+1
4420 IF ch = 0 THEN  GOSUB 4600: GOTO 4490
4430 ch = ch*8^p
4440 pk = PEEK(ad): pk = pk+ch
4450 POKE ad, pk
4460 p = p+1
4470 IF p = 3 THEN  p = 0: ad = ad+1
4475 DRAW 2 AT 128, 80
4490 RETURN
4600 IF PEEK(ad) > 0 THEN  ad = ad+1
4610 POKE ad, 128
4620 ad = ad+1: p = 0: RETURN
5000 PR #1: PRINT
5002 PRINT "Here is the shape table data:": PRINT
5005 n = sd+9
5010 ln = ln+10
5020 PRINT ln; " Data ";
5030 FOR t = 0 TO 10
5035 IF t > 0 THEN  PRINT ",";
5040 IF PEEK(n+t) = 0 THEN  PRINT "0"; : GOTO 5100
5050 PRINT PEEK(n+t);
5060 NEXT t
5070 n = n+11: PRINT
5080 GOTO 5010
5100 PR #0
5105 PRINT
5110 RETURN
5200 PR #1: PRINT: PRINT
5205 PRINT "Here is the shape loader subroutine:": PRINT
5210 PRINT "8000 qs = "; nt; ": pt = 16766: tl = 41000"
5215 LIST 8005-8199
5230 PR #0
5235 HOME
5240 RETURN
8000 qs = 2: pt = 16766: tl = 49000
8005 of = qs*2+3
8010 by = tl: GOSUB 8100
8015 POKE pt, lo: POKE pt+1, hi
8020 POKE tl, qs
8030 FOR n = 1 TO qs
8040 by = of: GOSUB 8100
8050 POKE tl+n*2, lo: POKE tl+n*2+1, hi
8060 READ a: POKE tl+of, a: of = of+1
8065 IF a <> 0 THEN 8060
8070 NEXT n
8080 RETURN
8100 hi = INT(by/256): lo = by-hi*256
8130 RETURN
9000 DATA            4,0
9010 DATA            0
```



Calendar

Program 18

If you are making plans for the Labor Day weekend in the year 1999 or just want to know what day your birthday falls on this year, this program can help. Calendar uses your Adam's printer to print any month in the twentieth century.

First enter the year by using the keypad on game controller 1. If you make a mistake, use the left controller button as a backspace key. Press the right controller button to enter the number. To enter the month, repeat the same process. Now sit back and watch the printer print you a monthly calendar.

Technical Description

The PR #1 statement in line 1500 causes everything that is sent to the screen to also be printed on the printer. Lines 1500-1960 print the month of the calendar; the printer is stopped with the PR #0 statement in line 160.

In making the calendar for a certain month, it is necessary to know the day of the week that the month begins on. This is done in lines 1200-1260. First the program finds the number of days between the requested date of the calendar and January 1, 1900. Since the program knows that January 1, 1900, was a Monday, it can calculate the day of the week for the month and year that you have requested.

Important Variables

Variable	Function
dy$(*number*)	Array containing the days of the week. Since there are seven days in a week, the size of *number* is 7.
m(*number*)	Array containing each month of the year. Each month is represented as a number, so the size of *number* is 12.
m$(*number*)	Array containing each month of the year. Unlike array m, array m$ stores each month by its name. Since there are 12 months in a year, the size of *number* is 12.

Program Description

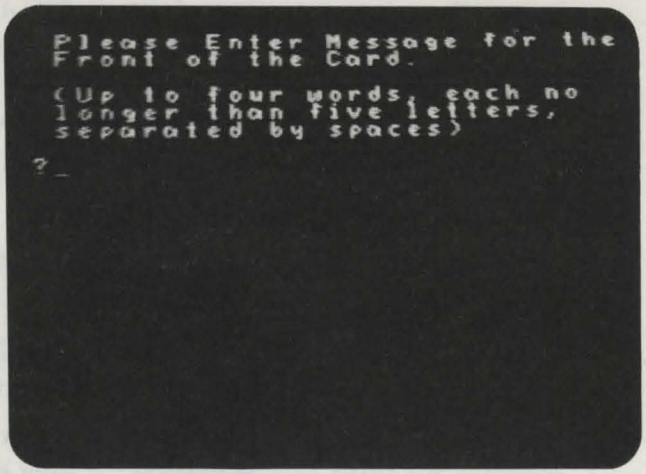
Lines	Function
1-95	Standard introduction.
100-199	Main program.
1000-1050	Subroutine to read in the data and initialize the variables.
1100-1190	Subroutine to allow the user to input the year and the month.
1200-1260	Subroutine to calculate the calendar for the chosen month.
1400-1465	Subroutine to allow the user to input choices on the keypad.
1500-1640	Subroutine to print out the calendar on the printer.
1700-1760	Three separate subroutines to print three different formats within the calendar.

1900-1960	Subroutine to print the day number on the calendar.
9000-9050	Data statements to store the months and days.

```
  1 REM       (c) 1984 by Brian Sawyer
  5 TEXT
 10 pm$ = "Calendar"
 15 FOR n = 1 TO 4
 20 PRINT CHR$(162)
 25 NEXT n
 30 tb = 17-INT(LEN(pm$)/2)
 35 PRINT TAB(tb); pm$
 40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
 45 PRINT " Hit Controller Button to Play"; CHR$(160)
 50 iv = 0: r = 0
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
    AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 GOSUB 1000
110 GOSUB 1100
120 GOSUB 1200
140 mo$ = mo$(mo)
145 sz = m(mo)
150 GOSUB 1500
160 PR #0: HOME
170 PRINT "Would you like to see another calendar?"
180 GET a$
190 IF a$ = "y" OR a$ = "Y" THEN 95
199 TEXT: END
1000 TEXT
1005 yn = 1900
1010 DIM i(10)
1011 DIM mo$(12)
1012 DIM dy$(6)
1015 DIM m(12)
1016 FOR n = 1 TO 12
1017 READ m(n)
1018 NEXT n
1020 FOR n = 1 TO 12
1025 READ mo$(n)
1030 NEXT n
1035 FOR n = 0 TO 6
```

```
1036 READ dy$(n)
1037 NEXT n
1040 st$ = "**************************************************
     **************************************************"
1050 RETURN
1100 HOME
1105 PRINT: PRINT
1110 PRINT " What year is the calendar"
1120 PRINT " ? ";
1130 GOSUB 1400
1140 IF no < 1901 OR no > 2000 THEN 1100
1150 yr = no
1160 PRINT " What month (1-12)"
1165 PRINT " ? ";
1170 GOSUB 1400
1175 IF no < 1 OR no > 12 THEN  HOME: GOTO 1160
1180  mo = no
1190 RETURN
1200  di = yr-yn
1220  ln = INT(di/4)+1
1225 IF yn = yr THEN 1235
1230 IF yr/4 = INT(yr/4) AND mo < 3 THEN  ln = ln-1
1235 IF yn = yr AND mo < 3 THEN  ln = 0
1240  dy = 365*di+ln: IF mo = 1 THEN 1250
1245 FOR n = 1 TO mo-1: dy = dy+m(n): NEXT n
1250  da = dy-INT(dy/7)*7: dy = da
1260 RETURN
1400 REM user input
1410 k = 1
1415 pd = PDL(13)
1420 p1 = PDL(7)
1422 p2 = PDL(9)
1423 IF p2 = 1 AND k > 1 THEN 1450
1425 IF p1 = 1 AND k > 1 THEN  k = k-1:
     PRINT CHR$(163); " "; CHR$(163);
1426 IF p1 = 1 AND PDL(7) = 1 THEN 1426
1430 IF pd > 9 OR k > 9 THEN 1415
1435 PRINT pd;
1440  i(k) = pd
1442  k = k+1
1443 IF PDL(13) <> 15 THEN 1443
1445 GOTO 1415
1450  no = 0: q = 0
1452 FOR k1 = k-1 TO 1 STEP  1
1455  no = no+i(k1)*10^q: q = q+1
1460 NEXT k1
1465 PRINT: RETURN
1500 PR #1: dd = 0
1510 PRINT
1521 GOSUB 1700
1522 yr$ = STR$(yr)
1525  tb = 40-INT(LEN(yr$)/2)
```

```
1530 PRINT TAB( tb); yr$
1535 PRINT
1536 tb = 40-INT(LEN(mo$)/2)
1540 PRINT TAB( tb); mo$
1545 GOSUB 1700
1546 tb = 5
1550 FOR n = 0 TO 6
1554 PRINT SPC( tb);
1555 PRINT "* "; dy$(n);
1556 tb = 7-LEN(dy$(n))
1560 NEXT n
1565 PRINT SPC( tb); "*"
1570 cn = 0
1575 FOR y = 1 TO 6
1580 GOSUB 1750
1585 FOR ly = 1 TO 4
1586 tb = 5
1590 FOR t = 0 TO 6
1594 PRINT SPC( tb);
1595 PRINT "*";
1596 tb = 8
1600 IF ly = 1 THEN  GOSUB 1900
1620 NEXT t
1621 PRINT SPC( tb); "*"
1625 NEXT ly
1626 IF dd = 1 THEN  GOSUB 1750: RETURN
1630 NEXT y
1640 RETURN
1700 REM  three different routines
1710 PRINT LEFT$(st$, 74)
1720 RETURN
1730 PRINT
1740 RETURN
1750 PRINT TAB(6); LEFT$(st$, 64)
1760 RETURN
1900 cn = cn+1
1910 IF cn <= dy THEN  RETURN
1920 IF cn-dy > sz THEN  dd = 1: RETURN
1925 tb = tb-LEN(STR$(cn-dy))
1930 PRINT STR$(cn-dy);
1931 RETURN
1935 FOR bk = 1 TO LEN(STR$(cn-dy))
1940 PRINT CHR$(8);
1945 NEXT bk
1950 PRINT STR$(cn-dy);
1960 RETURN
9000 DATA  31,28,31,30,31,30,31,31,30,31,30,31
9010 DATA  JANUARY,FEBRUARY,MARCH
9020 DATA  APRIL,MAY,JUNE,JULY,AUGUST
9030 DATA  SEPTEMBER,OCTOBER,NOVEMBER,DECEMBER
9040 DATA  Sun,Mon,Tues,Wed,Thurs
9050 DATA  Fri,Sat
```



Greetings

Program 19

Here is a way to make a simple greeting card by using the Adam's printer. First insert an 8 1/2-×11-inch sheet of paper into the printer, making sure that the top of the paper is flush with the three small guide rollers. Now run the program and enter a cover message. The cover message should be four words or less, and each word should not have more than five letters. You can use any letter in the alphabet plus the exclamation point. Type the message into the computer, using spaces to separate each word, and then press RETURN. The Adam will print each word of the message in a column in large letters with each word inverted.

Now enter the lines for the message that will appear inside the card. End each line by pressing RETURN. A line should be no more than 40 characters long, and you can enter as many as 25 lines. When you are finished, type a lowercase Q on a new line and press RETURN. Your message will now be printed on the paper. When the printer stops, remove the paper. Fold it in half lengthwise and then fold it in half widthwise, making sure that the print is always folded toward the outside. The completed card will contain your greeting on the front and your message on the inside.

Technical Description

Notice that each line of the inside message is centered on the right half of the page. A single sheet is 80 columns wide, but your message is centered on the 60th column since the paper will be folded in half. The process of centering the message is accomplished in lines 1720-1730, where the variable tb is set to the tab location where the message is to be printed.

Important Variables

Variable	Function
l(*number1, number2)*	Array containing the dot patterns that make up each of the cover letters. Since there are 26 letters in the alphabet plus the exclamation point, the size of *number1* is 27. Each letter is made from 5 numbers, so the size of *number2* is 5.
ms$	String variable containing the greeting.
ns$(*number*)	Array containing the lines of the message on the inside of the card. Since the maximum number of lines permissible for the inside message is 25, the size of *number* is 25.
tb	Tab location of a line of the message.

Program Description

Lines	Function
1-95	Standard introduction.
100-140	Main program.
1000-1010	Subroutine to dimension arrays c$, n$, and l.
1100-1150	Subroutine to read in the dot patterns for each of the letters.
1200-1250	Subroutine to input the greeting for the front of the card.

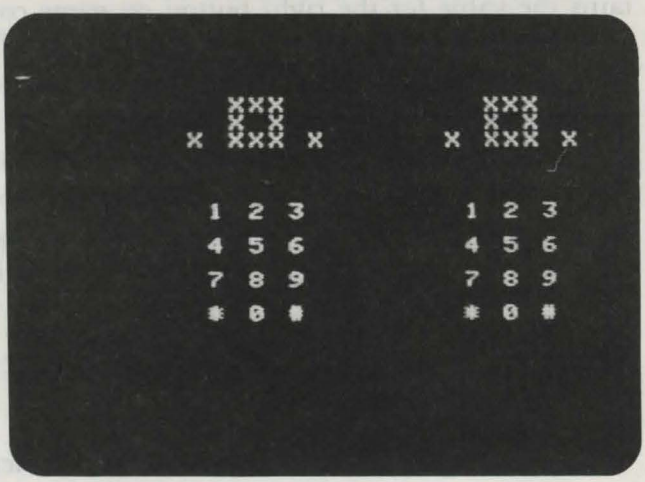
1300-1390	Subroutine to print the greeting in large letters.
1400-1430	Subroutine to advance the paper to the center of the page.
1500-1599	Subroutine to print one letter on the front of the card.
1600-1640	Subroutine to input the message for the inside of the card.
1700-1750	Subroutine to print the inside message.
1900-1925	Subroutine to print the message informing the user that an illegal character has been entered.
2000-2030	Subroutine to print the message informing the user that a word entered is too long.
2100-2130	Subroutine to print the message informing the user that too many words for the greeting were entered.
3000-3060	Subroutine to allow user to input a line of the message for the inside of the card.
9500-9630	Dot patterns for the letters.

```
 1 REM (c) 1984 by Brian Sawyer
 5 TEXT
10 pm$ = "Greetings"
15 FOR n = 1 TO 4
20 PRINT CHR$(162)
25 NEXT n
30 tb = 17-INT(LEN(pm$)/2)
35 PRINT TAB(tb); pm$
40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
45 PRINT " Hit Controller Button to Play"; CHR$(160)
50 iv = 0: r = 0
55 NORMAL: IF iv = 1 THEN  INVERSE
60 PRINT " Hit"; CHR$(160)
65 c = 0
70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
   AND (c < 40) THEN 70
80 IF c = 40 THEN  iv = 1-iv: GOTO 55
90 r = RND(-r)
92 NORMAL
95 CLEAR
100 GOSUB 1000
```

```
 105 GOSUB 1100
 110 GOSUB 1200
 120 GOSUB 1300
 125 GOSUB 1600
 127 yn = yn-16+INT(q/2)
 128 GOSUB 1400
 130 GOSUB 1700
 135 HOME: PRINT " Do you want another one?"
 136 PRINT " (y or n)"
 137 GET a$: IF a$ = "y" OR a$ = "Y" THEN 95
 140 END
1000 DIM c$(10), n$(30)
1005 DIM l(27, 5)
1010 RETURN
1100 REM
1105 FOR n = 1 TO 27
1110 FOR x = 1 TO 5
1120 READ l(n, x)
1130 NEXT x
1140 NEXT n
1150 RETURN
1200 HOME: le = 0: wd = 1: m$ = ""
1210 PRINT " Please Enter Message for the"
1220 PRINT " Front of the Card."
1225 PRINT
1230 PRINT " (Up to four words, each no
     longer than five letters,"
1235 PRINT " separated by spaces)"
1236 PRINT
1237 INPUT ms$
1238 FOR n = LEN(ms$) TO 1 STEP -1: IF MID$(ms$, n, 1) = " "
     THEN  NEXT n
1239 ms$ = MID$(ms$, 1, n)
1240 FOR n = 1 TO LEN(ms$)
1242 c$ = MID$(ms$, n, 1)
1243 IF c$ = " " AND le = 0 THEN 1248
1244 IF c$ = " " THEN  wd = wd+1: le = 0: m$ = m$+" ": GOTO 1248
1245 m$ = m$+c$
1246 le = le+1: IF le > 5 THEN  GOSUB 2000: GOTO 1200
1248 NEXT n: IF wd > 4 THEN  GOSUB 2100: GOTO 1200
1249 ms$ = m$
1250 RETURN
1300 PR #1
1305 m = 0
1310 FOR n = LEN(ms$) TO 1 STEP -1
1315 m = m+1
1330 c$(m) = MID$(ms$, n, 1)
1335 IF c$(m) = "." THEN  c$(m) = " "
1336 IF c$(m) = "!" THEN  c$(m) = "["
1340 IF c$(m) = " " THEN  m = m-1: GOSUB 1500: m = 0
1350 NEXT n
1360 IF m > 0 THEN  GOSUB 1500
1390 RETURN
```

```
1400 PR #1
1405 FOR n = yn TO 28
1410 PRINT
1420 NEXT n
1425 PR #0
1430 RETURN
1500 PRINT
1510 FOR h = 5 TO 1 STEP -1
1520 FOR s = 1 TO m
1540 i = ASC( d$(s))-96
1542 IF i > -32 AND i < -4 THEN  i = i+32
1543 IF i > 27 OR i < 1 THEN  i = 1
1545 b = l( i, h)
1546 FOR t = 5 TO 1 STEP -1
1547 b = b/2
1548 i$ = " "
1550 IF b <> INT(b) THEN  i$ = "*"
1555 b = INT(b)
1560 PRINT i$;
1570 NEXT t
1575 PRINT "  ";
1580 NEXT s
1590 PRINT
1592 yn = yn+1
1595 NEXT h
1596 PRINT
1597 yn = yn+2
1599 RETURN
1600 q = 1
1605 PR #0
1606 HOME
1607 PRINT: PRINT " Please enter the lines of your"
1608 PRINT " message for inside the card."
1610 GOSUB 3000
1615 IF LEN(n$(q)) > 40 THEN  PRINT "line too long.":
     GOTO 1610
1620 IF n$(q) = "q" OR n$(q) = "Q" THEN  RETURN
1625 q = q+1
1630 IF q < 30 THEN 1610
1640 RETURN
1700 PR #1
1710 FOR n = 1 TO q-1
1720 tb = 60-INT(LEN(n$(n))/2)
1730 PRINT TAB( tb); n$(n)
1740 NEXT n
1745 PR #0
1750 RETURN
1900 HOME
1910 PRINT "message contains illegal"
1915 PRINT "characters, try again."
1920 FOR t = 1 TO 1800: NEXT t
1925 RETURN
```

```
2000 HOME
2010 PRINT " word too long (5 letter limit)"
2015 PRINT " try again."
2020 FOR t = 1 TO 1800: NEXT t
2030 RETURN
2100 HOME
2110 PRINT " too many words (4 word limit)"
2115 PRINT " try again."
2120 FOR t = 1 TO 1800: NEXT t
2130 RETURN
3000 GET a$
3005 IF a$ = CHR$(13) THEN 3050
3020 PRINT a$;
3030 n$(q) = n$(q)+a$
3040 GOTO 3000
3050 IF LEN(n$(q)) > 40 THEN  PRINT: PRINT "Line too long.
     Try Again.": n$(q) = "": GOTO 3000
3060 fs = 0: PRINT: RETURN
9500 DATA  4,10,31,17,17,30,17,30,17,31
9510 DATA  15,16,16,16,15,30,17,17,17,30
9520 DATA  31,16,31,16,31,31,16,30,16,16
9530 DATA  14,16,23,18,14,17,17,31,17,17
9540 DATA  28,8,8,8,28,7,2,2,18,14
9550 DATA  18,20,24,20,18,16,16,16,16,31
9560 DATA  10,21,21,21,21,14,17,17,17,17
9570 DATA  14,17,17,17,14,30,17,30,16,16
9580 DATA  14,17,17,14,1 ,30,17,30,18,17
9590 DATA  14,16,14,1,14,31,4,4,4,4
9600 DATA  17,17,17,17,14,17,17,10,10,4
9610 DATA  21,21,21,21,10,17,10,4,10,17
9620 DATA  17,10,4,4,4,31,2,4,8,31
9630 DATA  4,4,4,0,4
```



Controller Program 20

This program helps you understand more about the Adam's game controller by drawing both game controller 1 and game controller 2 on the screen. As you press a number on one of the keypads, move one of the joysticks, or press one of the controller buttons, the corresponding part on the screen is highlighted. The value produced by the activated controller part is also displayed on the screen. To quit the program, press the sequence 123* on the keypad.

Technical Description

Each game controller is separated into four parts: the joystick, the left and right controller buttons, and the keypad. Each part produces a number that indicates its current state, so as a part is moved or a number entered, the value corresponding to that part changes. The value for each part can be found with the PDL function. For example, PDL(7) contains the value for the left controller button for

game controller 1. Normally PDL(7) contains the value 0, but when the button is depressed, the value in PDL(7) changes to 1. As another example, PDL(9) contains the value for the right button on game controller 1. This function works similarly to that of the left controller button.

In order to detect the movement of any part of a game controller, a test must be performed on the PDL corresponding to the part that has been moved. In this program, each PDL is tested within a continuing loop from lines 1300 to 1350. When a controller part is moved or a number entered, the controller part is highlighted on the screen with the INVERSE command, and the corresponding PDL is displayed, as well as the number that it is producing.

Important Variables

Variable	Function
j(*number*)	The current value of the PDL being examined. The size of *number* is 22.
x(*number*), y(*number*)	Coordinates of a controller part on the screen. The size of *number* is 13.

Program Description

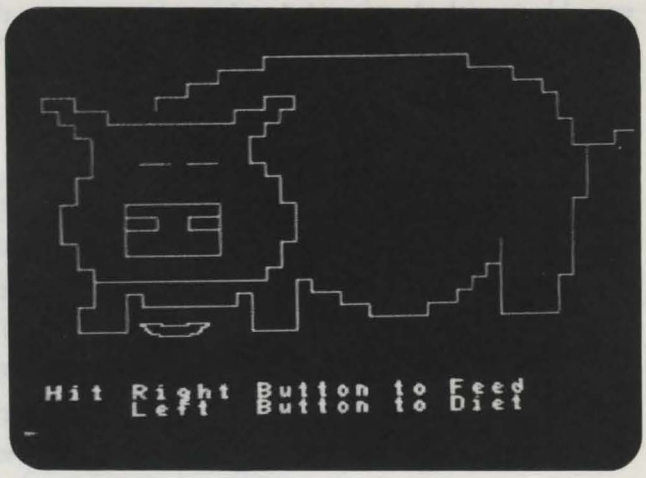
Lines	Function
1-95	Standard introduction.
100-140	Main program.
1000-1040	Subroutine to initialize variables; read in data for printing the controllers.
1100-1150	Subroutine to print the picture of the controller on the screen.
1200-1230	Subroutine to position the cursor on the part that is to be highlighted.
1300-1360	Subroutine to scan through each PDL function, checking to see if any values have changed.
1600-1680	Subroutine to show that the joystick has moved.
1700-1760	Subroutine to show that either controller button has been pressed.

1800-1890	Subroutine to show that a button on the keypad has been pressed.
2000-2030	Subroutine to highlight the portion of the controller that has been activated.
2100-2120	Subroutine to print the value of the PDL function.
2200-2210	Subroutine to move the cursor to the home position and print blank spaces.
9000-9056	Data for drawing the controllers.

```
  1 REM       (c) 1984 by Brian Sawyer
  5 TEXT
 10 pm$ = "Controller"
 15 FOR n = 1 TO 4
 20 PRINT CHR$(162)
 25 NEXT n
 30 tb = 17-INT(LEN(pm$)/2)
 35 PRINT TAB(tb); pm$
 40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
 45 PRINT " Hit Controller Button to Play"; CHR$(160)
 50 iv = 0: r = 0
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
    AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 GOSUB 1000
105 x = 0
110 GOSUB 1100
120 x = 13
130 GOSUB 1100
135 IF qt = 0 THEN  GOSUB 1300: GOTO 135
140 TEXT: END
1000 TEXT
1001 DIM a$(22), x(22), y(22)
1002 DIM j(13), s(4)
1003 h = 1: qt = 0
1005 FOR n = 1 TO 25
1006 dn$ = dn$+CHR$(162)
1007 NEXT n
1010 FOR n = 1 TO 22
1020 READ a$(n), x(n), y(n)
```

```
1030 NEXT n
1035 FOR n = 1 TO 3: s(n) = n: NEXT n: s(4) = 10
1040 RETURN
1100 FOR n = 1 TO 22
1110 dn = y(n)-4: GOSUB 1200
1120 HTAB x(n)+x
1130 PRINT a$(n); CHR$(128)
1140 NEXT n
1150 RETURN
1200 REM     position cursor
1210 PRINT CHR$(128)
1220 PRINT LEFT$(dn$, dn);
1230 RETURN
1300 FOR n = 4 TO 13
1310 j(n) = PDL(n)
1320 IF n < 6 AND j(n) > 0 THEN  GOSUB 1600: GOTO 1360
1330 IF n < 10 AND j(n) > 0 THEN  GOSUB 1700: GOTO 1360
1340 IF n < 12 AND j(n) > 0 THEN  GOSUB 1800: GOTO 1360
1350 NEXT n
1360 RETURN
1600 IF j(n) = 1 THEN  i = 15
1610 IF j(n) = 3 THEN  i = 16
1615 IF j(n) = 2 THEN  i = 17
1620 IF j(n) = 6 THEN  i = 18
1625 IF j(n) = 4 THEN  i = 19
1630 IF j(n) = 12 THEN  i = 20
1635 IF j(n) = 8 THEN  i = 21
1640 IF j(n) = 9 THEN  i = 22
1650 j2 = 0: IF n = 4 THEN  j2 = 1
1655 GOSUB 2100
1660 of = 0: GOSUB 2000
1665 IF PDL(n) = j(n) THEN 1665
1670 of = 1: GOSUB 2000
1676 GOSUB 2200
1680 RETURN
1700 j2 = 0
1705 IF n = 7 THEN  i = 13
1710 IF n = 9 THEN  i = 14
1715 IF n = 6 THEN  i = 13: j2 = 1
1720 IF n = 8 THEN  i = 14: j2 = 1
1725 GOSUB 2100
1730 of = 0: GOSUB 2000
1740 IF PDL(n) = j(n) THEN 1740
1750 of = 1: GOSUB 2000
1755 GOSUB 2200
1760 RETURN
1800 IF j(n) > 48 AND j(n) < 58 THEN  i = j(n)-48
1810 IF j(n) = 48 THEN  i = 11
1820 IF j(n) = 42 THEN  i = 10
1830 IF j(n) = 35 THEN  i = 12
1835 IF i = s(h) THEN  h = h+1: GOTO 1840
1837 h = 1
```

```
1840 j2 = 0
1845 IF h = 5 THEN  qt = 1: RETURN
1850 IF n = 10 THEN  j2 = 1
1855 GOSUB 2100
1860 of = 0: GOSUB 2000
1870 IF PDL(n) = j(n) THEN 1870
1880 of = 1: GOSUB 2000
1885 GOSUB 2200
1890 RETURN
2000 dn = y(i)-4: GOSUB 1200
2010 HTAB x(i)+j2*13
2019 INVERSE
2020 IF of = 1 THEN  NORMAL
2025 PRINT a$(i); CHR$(128)
2030 RETURN
2100 PRINT CHR$(128)
2110 PRINT "  The Value Of PDL("; n; ") = "; PDL(n)
2120 RETURN
2200 PRINT CHR$(128)
2205 PRINT "
2210 RETURN
9000 DATA "1",10,12
9002 DATA "2",12,12
9004 DATA "3",14,12
9006 DATA "4",10,14
9008 DATA "5",12,14
9010 DATA "6",14,14
9012 DATA "7",10,16
9014 DATA "8",12,16
9016 DATA "9",14,16
9018 DATA "*",10,18
9020 DATA "0",12,18
9030 DATA "#",14,18
9040 DATA "X",9,8
9042 DATA "X",15,8
9044 DATA "X",12,6
9046 DATA "X",13,6
9048 DATA "X",13,7
9050 DATA "X",13,8
9052 DATA "X",12,8
9054 DATA "X",11,8
9055 DATA "X",11,7
9056 DATA "X" 11,6
```



Pig Feeder

Program 21

This program contains a tool that allows you to draw and enlarge shapes from the shape table. In this program, the tool is used to show a pig at various sizes. You, the pig's dietician, can either feed the pig or advise it to diet. Each time the pig is fed, it gains weight, and each time the pig diets, it loses weight. Press the left button on game controller 1 to make the pig diet, and press the right controller button to feed the pig. This pig does have some common sense, however; if it becomes too fat, it will stop eating, and if it becomes too thin, it will break its diet. To end the program, press the * on the keypad.

Technical Description

This program contains a tool you can use in your own programs to enlarge high-resolution shapes to a specified amount. The subroutine from 1300 to 1320 draws the shape of the pig to a size specified by the variable n. The

SCALE command in line 1300 is used to set the size. The DRAW command on line 1310 draws the shape on the screen. Before calling the subroutine, set the variable n with the size of the shape you want to draw; then execute GOSUB 1300, and the shape will be enlarged and drawn.

Important Variables

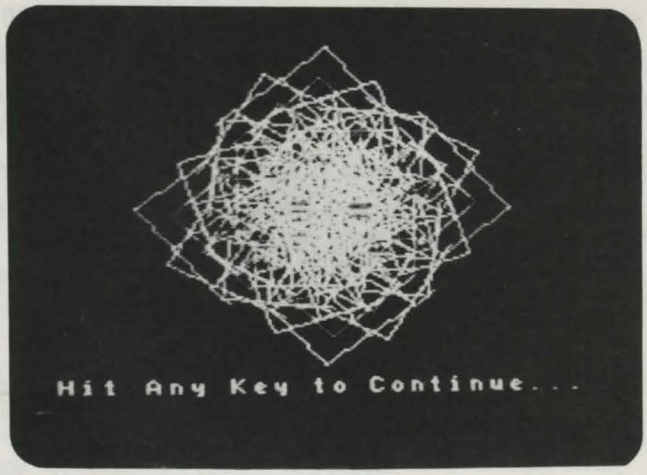
Variable	Function
n	Contains the number that represents how much the pig shape should be enlarged.
x,y	Position of the pig; the variable x represents the horizontal coordinate and the variable y represents the vertical coordinate.

Program Description

Lines	Function
0	Shape table memory pointer.
1-95	Standard introduction.
100-180	Main program.
1000-1050	Subroutine to initialize the variables.
1100-1130	Subroutine to query the user to press either the left or right controller button.
1200-1270	Subroutine to draw shape number 2, which is the bowl that the pig eats from.
1300-1320	Subroutine to draw the pig, which is in shape number 1, and increase or decrease the size by a factor of n.
8000-8130	Subroutine to load the pig shape table from the data into the Adam's memory.
9600-9712	Data for the shape of the pig.

```
  0 HIMEM :41000
  1 REM  (c) 1984 by Brian Sawyer
  5 TEXT
 10 pm$ = "Pig Feeder"
 15 FOR n = 1 TO 4
 20 PRINT CHR$(162)
 25 NEXT n
 30 tb = 16-INT(LEN(pm$)/2)
 35 PRINT TAB(tb); pm$
 40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
 45 PRINT " Hit Controller Button to Play"; CHR$(160)
 50 iv = 0: r = 0
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
    AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 GOSUB 1000
105 GOSUB 8000
120 n = 1
130 GOSUB 1200
140 GOSUB 1100
150 IF PDL(7) = 1 AND n > 1 THEN  n = n-1: GOTO 130
160 IF PDL(9) = 1 AND n < 6 THEN  n = n+1: GOTO 130
170 IF PDL(11) = 42 THEN  TEXT: END
180 GOTO 150
1000 HGR
1005 x = 128: y = 80
1010 HCOLOR  = 1
1015 SCALE  = 1: ROT  = 0
1050 RETURN
1100 HOME
1110 PRINT " Hit Right Button to Feed"
1120 PRINT "      Left  Button to Diet"
1130 RETURN
1200 HGR
1210 HCOLOR  = 1
1250 GOSUB 1300
1260 SCALE  = 1: DRAW 2 AT 52, 70+n*10
1270 RETURN
1300 SCALE  = n
1310 DRAW 1 AT 52, 56
1320 RETURN
8000 qs = 2: pt = 16766: tl = 41000
8005 of = qs*2+3
8010 by = tl: GOSUB 8100
8015 POKE pt, lo: POKE pt+1, hi
8020 POKE tl, qs
```

```
8030 FOR n = 1 TO qs
8040 by = of: GOSUB 8100
8050 POKE tl+n*2, lo: POKE tl+n*2+1, hi
8060 READ a: POKE tl+of, a: of = of+1
8065 IF a <> 0 THEN 8060
8070 NEXT n
8080 RETURN
8100 hi = INT(by/256): lo = by-hi*256
8130 RETURN
9600 DATA 109,173,18,63,63,63,54,54,45,45,45
9606 DATA 36,36,62,55,237,219,27,45,60,7,128
9609 DATA 128,128,128,128,128,128,159,39,39,39,45
9612 DATA 46,53,45,45,45,45,44,37,45,62 62
9615 DATA 62,54,53,53,46,54,62,54,55,63,63
9618 DATA 63,63,63,39,39,60,36,44,36,37,100
9621 DATA 73,9,128,128,27,44,37,45,44,45,44
9624 DATA 45,45,45,45,45,45,53,45 53,45,46
9627 DATA 54,53,46,45,44,229,159,210,38,52,54
9630 DATA 62,54,54,54,55,54,63,63,36,36,36
9633 DATA 54,55,55,55,63,62,63,39,63,60,39
9636 DATA 55,54,62,63,36,60,62,63,63,39,55
9639 DATA 54,63,39,44,36,0
9703 DATA 45,45,45,53,45 45,45,45,45,44,45
9706 DATA 45,45,53,55,63,54,55,63,55,63,63
9709 DATA 63,63,63,63,60,63,60,60,36,63,44
9712 DATA 0
```



Rotator

Program 22

This program chooses one of five simple high-resolution shapes and draws it on the screen. The shape is then rotated in a circular pattern. Once the shape has been rotated, it is enlarged, and its color is changed. The new shape will then rotate again. This process is repeated four times, after which a spiral pattern is produced on the screen. Press Q to quit the program, or press any other key to see another pattern.

Technical Description

Rotator contains a tool that you can use to rotate your own shapes on the high-resolution screen. The subroutine that simulates the rotation of the shape is located in lines 3000 to 3060. The rotation is accomplished by repeatedly increasing the shape's angle with the ROT command and then drawing the shape with the DRAW command. Both of these commands are contained within the body of a

FOR-NEXT loop that will execute 64 times, the number needed to draw the circle. The size of the shape is set in the SCALE command in line 3120, and the shape, which is number n, is drawn in line 3130 using the DRAW command.

Notice that each time the shape is enlarged, it is drawn in a new color. This color is randomly picked on line 4000 from the 16 high-resolution colors. The color for the shape is then set using the HCOLOR command in line 4010. When using this code in your programs, set the variable to the number of the shape in the shape table that you want to have rotated, and then execute GOSUB 3100.

Important Variables

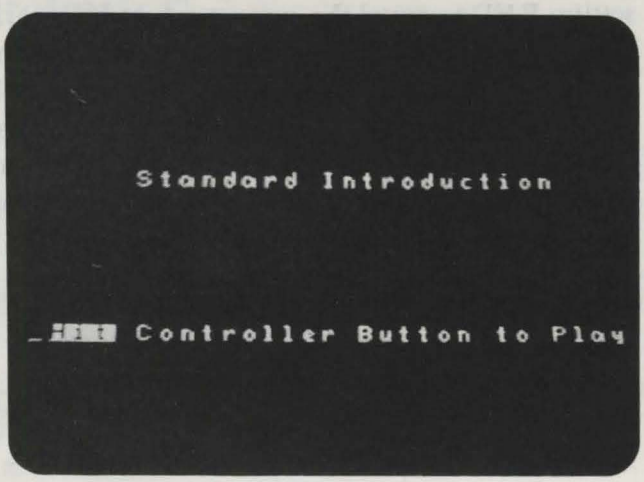
Variable	Function
cl	The random color chosen for the shape that is currently being drawn.
n	The degree that the shape is currently being rotated.
s	The size to which the shape is currently being enlarged.

Program Description

Lines	Function
0	Shape table pointer.
1-95	Standard introduction.
100-160	Main program.
3000-3060	Subroutine that calls other subroutines which draw the pig.
3100-3160	Subroutine to enlarge and rotate the current shape.
4000-4020	Subroutine to pick a random color for the shape.
8000-8130	Subroutine to load the shapes from data into memory.
9615-9648	Data statements containing the shapes.

```
  0 HIMEM :41000
  1 REM (c) 1984 by Brian Sawyer
  5 TEXT
 10 pm$ = "Rotator"
 15 FOR n = 1 TO 4
 20 PRINT CHR$(162)
 25 NEXT n
 30 tb = 16-INT(LEN(pm$)/2)
 35 PRINT TAB(tb); pm$
 40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
 45 PRINT " Hit Controller Button to Play"; CHR$(160)
 50 iv = 0: r = 0
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
    AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 REM   main program
110 GOSUB 8000
115 n = INT(RND(1)*5+1)
120 HGR: GOSUB 4000
125 SCALE  = 1: ROT  = 0
130 GOSUB 3000
135 HOME: PRINT " Press 'Q' to quit. Press any"
136 PRINT "other key to continue."
140 GET a$
150 IF a$ <> "q" AND a$ <> "Q" THEN 115
155 TEXT
160 END
3000 FOR s = 1 TO 9 STEP 2
3005 GOSUB 4000
3040 GOSUB 3100
3050 NEXT s
3060 RETURN
3100 FOR r = 1 TO 64 STEP 4
3120 SCALE  = s: ROT  = r
3130 DRAW n AT 118, 80
3150 NEXT r
3160 RETURN
4000 cl = INT(RND(1)*15+1): IF cl = 4 THEN 4000
4010 HCOLOR  = cl
4020 RETURN
8000 qs = 5: pt = 16766: tl = 41000
8005 of = qs*2+3
8010 by = tl: GOSUB 8100
```

```
8015 POKE pt, lo: POKE pt+1, hi
8020 POKE tl, qs
8030 FOR n = 1 TO qs
8040 by = of: GOSUB 8100
8050 POKE tl+n*2, lo: POKE tl+n*2+1, hi
8060 READ a: POKE tl+of, a: of = of+1
8065 IF a <> 0 THEN 8060
8070 NEXT n
8080 RETURN
8100 hi = INT(by/256): lo = by-hi*256
8130 RETURN
9615 DATA 36,45,54,63,0
9621 DATA 44,44,44,44,44,44,54,54,54,62,63
9624 DATA 63,63,44,0
9630 DATA 36,45,36,63,63,55,46,53,54,63,55
9633 DATA 54,45,45,45,37,36,63,39,0
9639 DATA 45,45,45,5,0
9645 DATA 36,36,44,53,54 54 54,54,45,37,36
9648 DATA 36,0
```



Standard Introduction

Program 23

This small segment of code appears at the start of all the programs in this book. It prints the title you have chosen for your program on the screen and prepares the Adam to play a game.

Enter the Standard Introduction into the computer and then save it on tape. Before you enter a program from this book into the computer, load the Introduction, and then change the title on line 10 to the title of the program that you will be entering. This process will eliminate retyping the Introduction for each program.

Technical Description

This routine provides a pleasing way to begin a game. It also initializes important variables and functions that are used in many of the programs. One such function is the random number function, RND, which is used to ensure that

each game will be unique. This function must be initialized with a *seed value* to operate properly. Initializing is accomplished in the Standard Introduction by setting RND to equal the current value of the variable r on line 90. The value of r when it is assigned to the RND function is determined by the amount of time that has passed before the user presses the controller button to begin the game.

Notice that the flashing message on the screen telling the player to hit the controller button is done by printing the word "hit" in INVERSE mode and then in NORMAL mode. This is repeated until the player presses the controller button.

On line 10, you replace the title "Standard Introduction" with the title that you have chosen for your program.

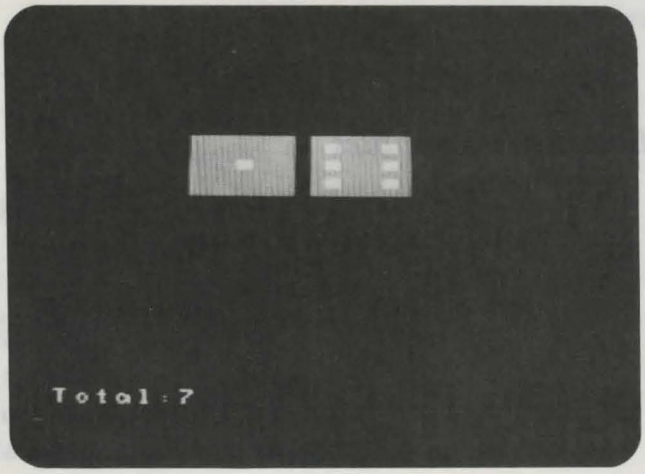
Important Variables

Variable	Function
tb	Tab location to center the program name.
c	Period of time the flashing message has been on the screen.
iv	Indicates whether the flashing message is in normal or inverse text.
r	Seed value for the random number function.
pm$	The program name.

Program Description

Lines	Function
15-25	Move the cursor to the center of the screen.
30-35	Print the program title in the center of the screen.
40	Moves the cursor to the bottom of the screen.
45	Prints the message that instructs the user to press the controller button.
55-80	Flash the message.
90	Initializes the random number function with a seed value.
92-95	Clear the variables and return to the normal printing mode.

```
 1 REM (c) 1984 By Brian Sawyer
 5 TEXT
10 pm$ = "Standard Introduction"
15 FOR n = 1 TO 4
20 PRINT CHR$(162)
25 NEXT n
30 tb = 16-INT(LEN(pm$)/2)
35 PRINT TAB(tb); pm$
40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
45 PRINT " Hit Controller Button to Play"; CHR$(160)
50 iv = 0: r = 0
55 NORMAL: IF iv = 1 THEN  INVERSE
60 PRINT " Hit"; CHR$(160)
65 c = 0
70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
   AND (c < 40) THEN 70
80 IF c = 40 THEN  iv = 1-iv: GOTO 55
90 r = RND(-r)
92 NORMAL
95 CLEAR
```



Dice

Program 24

This tool rolls two dice and prints them along with their total point value on the screen. You can use this tool in games that require an element of chance. Each time you press a key, the dice will be rolled. To quit the program, press Q.

Technical Description

To simulate the roll of a die, a random number is generated by the statement r=INT(RND(1)*6+1) on line 1300. The RND(1) function generates a random number from 0 to 1, but not including 1. This number is multiplied by 6 to give a number from 0 to 5, and then 1 is added to give a number from 1 to 6. These are all the numbers on a die. Because a number produced this way probably has a decimal part, the INT function is used to make it an integer. The resulting number is then stored in the variable t1 before it is used to print the die.

The HLIN=AT command on line 1210 is used successively by a FOR-NEXT loop to draw the die. This command is contained within a subroutine that is executed twice to draw each die.

Important Variables

Variable	Function
bk	Color of the dots on the dice.
dx,dy	Position at which to draw the dots on the die. The variable dx contains the horizontal coordinate and the variable dy contains the vertical coordinate.
r	Random number from 1 to 6. Variable r is later assigned to variable tl.
tl	Number to be printed on the die.
xc(*number1, number2*), yc(*number1, number2*)	Array containing the patterns for the dots to be drawn on the dice. The array xc contains the pattern for each of the dice on the horizontal coordinate, and the array yc contains the pattern for each of the dice on the vertical coordinate. Since there are six different numbers on a die, the size of *number1* is 6. The second dimension of both xc and yc, *number 2,* contains the numbers that will be used to draw the die. The size of *number2* equals 6.

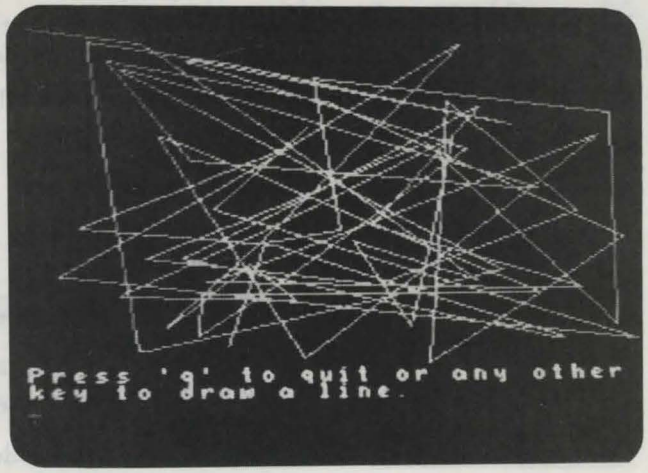
Program Description

Lines	Function
1-95	Standard introduction.
100-140	Main program.
1000-1040	Subroutine to set the color of the dice and to call the routine that draws them.

1100-1170	Subroutine to initialize arrays xc and yc of dot patterns for the different die numbers.
1200-1230	Subroutine to draw the die.
1300-1310	Subroutine to generate the random number for each die.
1400-1430	Subroutine to draw the dots on the dice corresponding to the random number.
1500-1590	Main subroutine that calls other subroutines to roll and print the dice.
9000-9050	Data statements for a die.

```
  1 REM (c) 1984 by Brian Sawyer
  5 TEXT
 10 pm$ = "Dice"
 15 FOR n = 1 TO 4
 20 PRINT CHR$(162)
 25 NEXT n
 30 tb = 16-INT(LEN(pm$)/2)
 35 PRINT TAB(tb); pm$
 40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
 45 PRINT " Hit Controller Button to Play"; CHR$(160)
 50 iv = 0: r = 0
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
    AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 GOSUB 1100
110 GOSUB 1500
120 GET a$
130 IF a$ = "q" OR a$ = "Q" THEN  TEXT: END
140 GOTO 110
1000 REM   set color;draw dice
1010 COLOR  = bk
1015 s = 6
1020 x = 10: y = 10: GOSUB 1200
1030 x = 18: y = 10: GOSUB 1200
1040 RETURN
1100 GR
```

```
1105 DIM xc(6, 6), yc(6, 6)
1110 FOR n = 1 TO 6
1120 FOR m = 1 TO 6
1125 READ xc(n, m)
1130 IF xc(n, m) = 999 THEN 1160
1140 READ yc(n, m)
1150 NEXT m
1160 NEXT n
1165 dc = 15
1166 bk = 7
1170 RETURN
1200 FOR by = y TO y+s
1210 HLIN x, x+s AT by
1220 NEXT by
1230 RETURN
1300 r = INT(RND(1)*6+1)
1310 RETURN
1400 COLOR  = dc
1405 FOR n = 1 TO 6
1406 IF xc(n, n) = 999 THEN 1430
1410 dx = xc(n, n)*2+x+1
1415 dy = yc(n, n)*2+y+1
1420 PLOT dx, dy
1425 NEXT n
1430 RETURN
1500 GOSUB 1000
1510 GOSUB 1000
1514 x = 10: y = 10
1520 GOSUB 1300
1530 GOSUB 1400
1535 tl = r
1540 x = 18: y = 10
1550 GOSUB 1300
1560 GOSUB 1400
1565 tl = tl+r
1570 HOME
1575 PRINT " Total:"; tl
1590 RETURN
9000 DATA 1,1,999
9010 DATA 0,0, 2,2, 999
9020 DATA 0,2, 1,1, 2,0, 999
9030 DATA 0,0, 2,0, 0,2, 2,2, 999
9040 DATA 0,0, 2,0, 0,2, 2,2, 1,1, 999
9050 DATA 0,0, 0,1, 0,2, 2,0, 2,1, 2,2, 999
```



Web

Program 25

This tool draws lines randomly on the screen. These lines can be of any length and drawn in any possible direction. Each time you press a key, a line will be drawn. If you hold a key down, the lines will be drawn continuously. Each successive line will begin at the endpoint of the last line. If you continue to press the keys, a weblike pattern will form. This tool can be used to create unusual backgrounds, explosions, spider webs, and many other things in your programs. Press Q to quit the program.

Technical Description

Lines 1100-1130 will draw a line from the end of the last line drawn to a random point on the Adam's screen. The endpoint of any line drawn on the screen is determined by using the random number function RND. The horizontal coordinate of the endpoint is produced by the statement x=INT(RND(1)*255+1) in

line 1100 and has a maximum value of 256. The vertical coordinate is produced in line 1110 by the statement y=INT(RND(1)*159+1) and has a maximum value of 160. The largest numbers produced by these statements are the maximum coordinates that the drawing area in the high-resolution mode will allow. You can reduce this area by changing 255 in line 1100 or 159 in line 1110 to a smaller number.

Important Variables

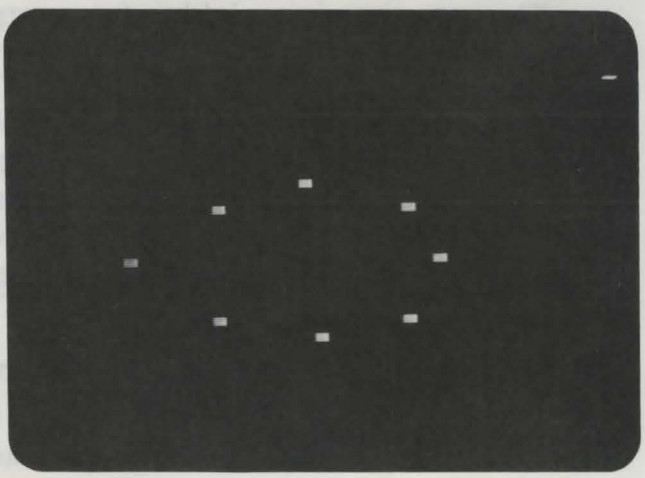
Variable	Function
a$	Contains the value of the last key pressed by the user.
x,y	Random screen coordinates of the current line. The variable x contains the horizontal coordinates and the variable y contains the vertical coordinates.

Program Description

Lines	Function
1-95	Standard introduction.
100-140	Main program.
1000-1040	Subroutine to enter the high-resolution mode, set the color, and print the message informing the reader to press another key.
1100-1130	Subroutine to draw a line at a random place on the screen.

```
 1 REM (c)1984 by Brian Sawyer
 5 TEXT
10 pm$ = "Web"
15 FOR n = 1 TO 4
20 PRINT CHR$(162)
25 NEXT n
30 tb = 17-INT(LEN(pm$)/2)
35 PRINT TAB(tb); pm$
40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
45 PRINT " Hit Controller Button to Play"; CHR$(160)
50 iv = 0: r = 0
55 NORMAL: IF iv = 1 THEN  INVERSE
```

```
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
    AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 GOSUB 1000
110 GOSUB 1100
120 GET a$
130 IF a$ <> "q" AND a$ <> "Q" THEN 110
135 TEXT
140 END
1000 HGR
1010 HCOLOR  = 1
1020 HOME
1025 PRINT "Press 'q' to quit or any other"
1030 PRINT "key to draw a line."
1040 RETURN
1100 x = INT(RND(1)*255+1)
1110 y = INT(RND(1)*159+1)
1120 HPLOT  TO x, y
1130 RETURN
```



Boom

Program 26

This tool makes a colorful explosion on your screen, hurling fireballs in all directions. Use this tool in your programs for fireworks, supernovas, volcanos, or any other kinds of explosions you can imagine.

Technical Description

Lines 1000-1530 draw the explosion on the low-resolution graphics screen. You can use this code in your games to create an explosion. Set the values of ex and ey to the screen coordinates where you want the explosion to be centered (ex will contain the horizontal coordinate, and ey will contain the vertical coordinate). To make the explosion happen, execute the subroutine GOSUB 1000. This routine will not alter the background of the screen while the explosion occurs because it saves the color of the screen position that a piece of debris will be occupying in array s. The debris is then printed in that position. Finally, as the

debris leaves the position, the original color is restored. Lines 100-111 are necessary to initialize the variables for the explosion and must appear in your program.

There are nine pieces of debris from every explosion, and their positions are stored in arrays px and py; px contains the horizontal coordinate, and py contains the vertical coordinate. To make the explosion appear convincing, each piece of debris is moved outward from the center of the explosion in a different direction and speed. This is achieved by having two arrays (cx for the horizontal direction and cy for the vertical direction) contain fractional values that indicate not only direction, but speed for each piece of debris. The values of the arrays cx and cy are added to the arrays px and py, respectively, to give a new position for each piece of debris.

Important Variables

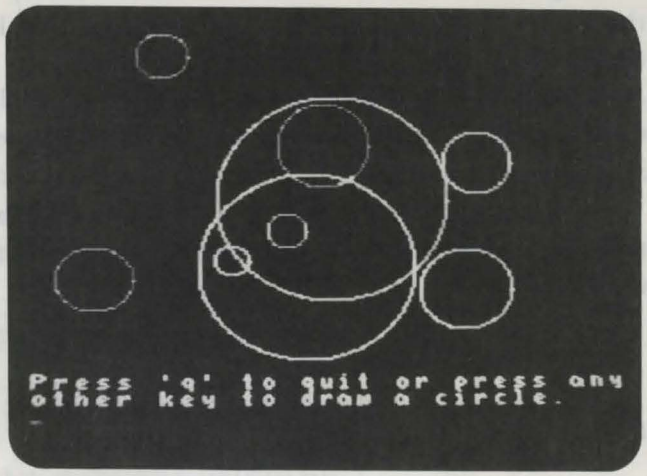
Variable	Function
ex,ey	Screen position where the explosion is centered. The variable ex contains the horizontal coordinate and the variable ey contains the vertical coordinate.
cl(*number*)	Array containing the colors of each piece of debris. Since there are nine pieces of debris, the size of *number* is 9.
cx(*number*), cy(*number*)	Arrays containing the speed at which each piece of debris is moving from the center of the explosion. Array cx( ) contains the speed for the horizontal direction, and array cy( ) contains the speed for the vertical direction.
px(*number 1, number2*), py(*number 1, number 2*)	Arrays containing the screen coordinates of each piece of debris. Array px( ) contains the horizontal coordinate, and array py( ) contains the vertical coordinate. The size of *number 1* and *number 2* is 9.
s*number1, number2*(9)	Contains the original color of the space that a piece of debris is currently occupying. The size of *number1* and *number2* is 9.
sz	The number of pieces of debris in the explosion.

Program Description

Lines	Function
1-95	Standard introduction.
100-140	Main program.
1000-1050	Subroutine that calls other subroutines necessary to make the explosion.
1100-1190	Subroutine to draw all the debris at their new positions.
1200-1290	Subroutine to move the debris away from the center of the explosion.
1300-1350	Subroutine to initialize all debris to start at the center of the explosion.
1400-1460	Subroutine to set up the array for the speed of the debris.
1500-1530	Subroutine to redraw the original color of the position that was occupied by the debris.

```
 1 REM  (c) 1984 by Brian Sawyer
 5 TEXT
10 pm$ = "Boom"
15 FOR n = 1 TO 4
20 PRINT CHR$(162)
25 NEXT n
30 tb = 16-INT(LEN(pm$)/2)
35 PRINT TAB(tb); pm$
40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
45 PRINT " Hit Controller Button to Play"; CHR$(160)
50 iv = 0: r = 0
55 NORMAL: IF iv = 1 THEN  INVERSE
60 PRINT " Hit"; CHR$(160)
65 c = 0
70 c = c+1: r = r+1: IF (PDL(7) = 0)
   AND (PDL(9) = 0) AND (c < 40) THEN 70
80 IF c = 40 THEN  iv = 1-iv: GOTO 55
90 r = RND(-r)
92 NORMAL
95 CLEAR
100 GR: COLOR  = 2
105 sz = 9
110 DIM n(sz), cl(sz)
111 DIM cx(sz), cy(sz), px(sz, 9), py(sz, 9), s(sz, 9)
120 ex = 20: ey = 20
```

```
 130 GOSUB 1000
 135 HOME
 136 PRINT "Press 'Q' to quit or press"
 137 PRINT "or press any other key to draw another explosion."
 138 GET a$: IF a$ <> "q" AND a$ <> "Q" THEN 95
 140 TEXT: END
1000 GOSUB 1400
1010 GOSUB 1300
1020 GOSUB 1200
1030 GOSUB 1100
1040 GOSUB 1500
1050 RETURN
1100 FOR q = 1 TO 6
1105 FOR n = 1 TO sz-1
1110 COLOR  = s(n, q): PLOT px(n, q), py(n, q)
1120 COLOR  = cl(q): PLOT px(n, q+1), py(n, q+1)
1180 NEXT n
1185 NEXT q
1190 RETURN
1200 FOR q = 1 TO 6
1205 FOR n = 1 TO sz-1
1206 cl(q) = q
1210 IF n(n) = 1 THEN  cx(n) = 0: cy(n) = 0
1230 px(n, q+1) = px(n, q)+cx(n)
1240 py(n, q+1) = py(n, q)+cy(n)
1244 s(n, q+1) = SCRN(px(n, q+1), py(n, q+1))
1245 IF px(n, q+1) > 39 OR px(n, q+1) < 0 THEN
     n(n) = 1: px(n, q+1) = px(n, q)
1246 IF py(n, q+1) > 39 OR py(n, q+1) < 0 THEN
     n(n) = 1: py(n, q+1) = py(n, q)
1280 NEXT n
1285 NEXT q
1290 RETURN
1300 s = SCRN(x, y)
1305 FOR n = 1 TO sz
1310 py(n, 1) = ey
1320 px(n, 1) = ex
1330 s(n, 1) = s
1340 NEXT n
1350 RETURN
1400 q = 0
1420 FOR n = 0 TO (2*3.14159) STEP (2*3.14159/(sz-1))
1430 q = q+1
1440 cx(q) = COS(n)*3
1445 cy(q) = SIN(n)*3
1450 NEXT n
1460 RETURN
1500 FOR n = 1 TO sz
1510 COLOR  = s(n, 7): PLOT px(n, 7), py(n, 7)
1520 NEXT n
1530 RETURN
```



Circles

Program 27

This tool draws circles of a random size and color on the high-resolution screen. (Black circles are not drawn since they would not show against the black background.) Each time you press a key, a circle is drawn. As the circle is being drawn, the coordinates of the center of the circle and its radius are displayed at the bottom of the screen. To quit, press Q.

Technical Description

The subroutine from lines 1000 to 1090 will draw a circle of radius r at center x,y on the high-resolution screen. The color of the circle is specified by the variable cl. The variables x and y, which together form the center of the circle, are used along with the variable r in lines 1010 and 1020 to calculate a point on the circle. This point is stored in variables px and py, where the variable px contains the horizontal coordinate of the point, and the variable py contains the vertical

coordinate. Both values of px and py are tested to determine if either coordinate is in range on the screen. If both coordinates satisfy the test, the point is plotted on the screen by using the HPLOT command on line 1070. This process is continued within a FOR-NEXT loop until all the points of the circle have been calculated. Isolate this section of the program and include it in your programs to draw a circle. To use this subroutine, set the values of the variables x,y, r, and cl, and then execute GOSUB 1000.

To demonstrate the program's ability to draw circles, the RND function is used in lines 105-130 to generate random values for the position, color, and size of the circle.

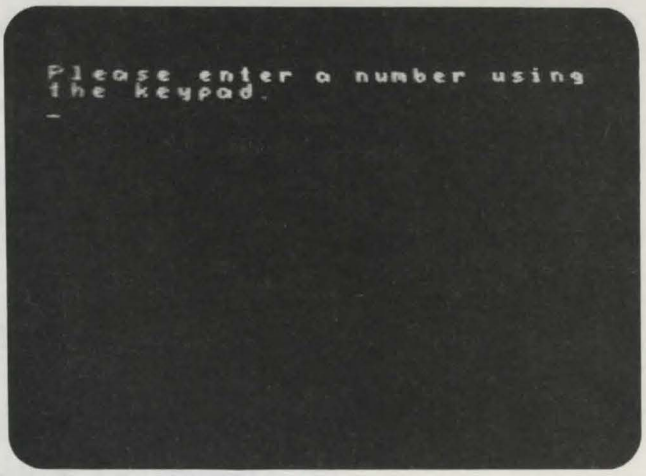
Important Variables

Variable	Function
cl	The color of the circle being drawn.
px,py	The point being plotted on the circle. The variable px contains the horizontal coordinate and the variable py contains the vertical coordinate.
r	The radius of the circle.
x,y	The screen position of the center of the circle. The variable x contains the horizontal coordinate and the variable y contains the vertical coordinate.

Program Description

Lines	Function
1-95	Standard introduction.
100-170	Main program; generate the random color, radius, and screen position of the circle.
1000-1090	Draws a circle of radius r, in color cl, centered at point x,y.

```
  1 REM (c) 1984 by Brian Sawyer
  5 TEXT
 10 pm$ = "Circles"
 15 FOR n = 1 TO 4
 20 PRINT CHR$(162)
 25 NEXT n
 30 tb = 16-INT(LEN(pm$)/2)
 35 PRINT TAB(tb); pm$
 40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
 45 PRINT " Hit Controller Button to Play"; CHR$(160)
 50 iv = 0: r = 0
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0)
    AND (PDL(9) = 0) AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 HGR
105 cl = INT(RND(1)*15+1)
107 IF cl = 0 OR cl = 4 THEN 105
110 x = INT(RND(1)*255+1)
120 y = INT(RND(1)*160+1)
130 r = INT(RND(1)*95+5)
132 IF x+r > 255 OR x-r < 1 OR y+r > 156
    OR y-r < 1 THEN 110
135 HOME: PRINT " Center at "; x; ","; y; "  Radius="; r
140 GOSUB 1000
145 HOME
146 PRINT "Press 'q' to quit or press any"
147 PRINT "other key to draw a circle."
150 GET a$
155 HOME
160 IF a$ = "q" OR a$ = "Q" THEN  TEXT: END
170 GOTO 105
1000 HCOLOR  = cl
1005 FOR n = 0 TO (2*3.14159) STEP 1/r
1010 px = x+COS(n)*r
1020 py = y+SIN(n)*r
1030 IF px < 0 THEN  RETURN
1040 IF py < 0 THEN  RETURN
1050 IF px > 255 THEN  RETURN
1060 IF py > 156 THEN  RETURN
1070 HPLOT px, py
1080 NEXT n
1090 RETURN
```



Keypad Input

Program 28

This tool allows you to enter a number into the computer by using the keypad on game controller 1. Press the left controller button to delete the last numeral pressed on the keypad. Press the right controller button to enter the number into the computer.

Technical Description

Lines 1400 to 1470 of this routine are used in many of the programs in this book. Each number that is pressed on the keypad is stored in the function PDL(13). The number that is pressed on the keypad is not the same number that is produced by the controller, however. When 0 is pressed, PDL(13) contains 0; when 1 is pressed, PDL(13) contains 1, and so on. Since PDL(13) can only represent a single digit from the keypad, array i is used to accumulate each keypad entry until the right controller button is pressed to indicate that the input is

complete. A formula then fuses each entry to form a single number, which is stored in the variable no. In order for this routine to work, your program must execute the statement DIM i(*size*) before calling the routine. The value of *size* is set to the maximum number of digits in a keypad input. In this example, *size* equals 9. By using array i to store an entry from the keypad, you can correct a mistake before entering the input. The variable k contains the array entry for the last keypad input, so decrementing k will simulate a backspace since the next keypad input will be inserted at the kth entry of array i. To begin the keypad routine, execute the statement GOSUB 1400.

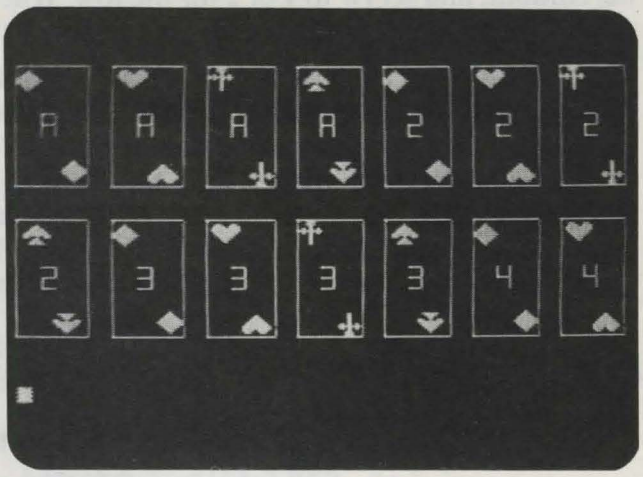
Important Variables

Variable	Function
k	Number of digits entered.
i(*number*)	Array containing the digits for the input number. The size of *number* is determined by the maximum number of digits allowed for a number.
n0	The value of the number input.
p1	Contains the latest value of PDL(7), which is the position of the left button on game controller 1.
p2	Contains the latest value of PDL(9), which is the right controller button.
pd	Contains the latest value of PDL(13), which is the number that has been pressed on the controller keypad.

Program Description

Lines	Function
1-95	Standard introduction.
100-140	Main program.
1400-1470	Accept a number from the keypad on game controller 1.

```
  1 REM (c) 1984 by Brian Sawyer
  5 TEXT
 10 pm$ = "Keypad Input"
 15 FOR n = 1 TO 4
 20 PRINT CHR$(162)
 25 NEXT n
 30 tb = 17 INT(LEN(pm$)/2)
 35 PRINT TAB(tb); pm$
 40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
 45 PRINT " Hit controller button to Play"; CHR$(160)
 50 iv = 0: r = 0
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0)
    AND (PDL(9) = 0) AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 DIM i(20)
105 HOME: PRINT
110 PRINT " Please enter a number using"
120 PRINT " the keypad."
125 PRINT " ";
130 GOSUB 1400
135 PRINT: PRINT " The number is "; no
140 END
1400 REM    keypad input
1410 k = 1
1415 pd = PDL(13)
1420 p1 = PDL(7)
1422 p2 = PDL(9)
1423 IF p2 = 1 AND k > 1 THEN 1450
1425 IF p1 = 1 AND k > 1 THEN  k = k-1:
     PRINT CHR$(163); " "; CHR$(163);
1426 IF p1 = 1 AND PDL(7) = 1 THEN 1426
1430 IF pd > 9 OR k > 9 THEN 1415
1435 PRINT pd;
1440 i(k) = pd
1442 k = k+1
1443 IF PDL(13) <> 15 THEN 1443
1445 GOTO 1415
1450 no = 0: q = 0: FOR k1 = k-1 TO 1 STEP -1
1455 no = no+i(k1)*10^q: q = q+1
1460 NEXT: PRINT
1470 RETURN
```



Card

Program 29

This program demonstrates a tool that will draw playing cards on the screen. The entire deck of cards is drawn, but since the screen cannot accommodate 52 cards, only 14 are drawn at a time. To continue with the display, press any key. Write your own card game using this tool.

Technical Description

You can use this tool to draw the cards in your card games. The cards are drawn in the high-resolution mode. To have a card printed, the program must set the value of the variable no to the number of the card that you want drawn. The permissible values of no range from 1 to 13, with 1 equaling an ace, 2 through 10 equaling the face value of the card, 11 for a jack, 12 for a queen, and 13 for a king. For the suit of the card, set the variable su to one of the following: 1 for a diamond, 2 for a heart, 3 for a spade, and 4 for a club. To specify where you want

your card to be drawn on the screen, set the variable cl to a number from 1 to 7 for the column, and set rw to 1 or 2 for the row. When all the variables have been set, draw the card by executing the statement GOSUB 2700.

You do not need to include lines 1 through 145 in your card games since they only demonstrate the tool. However, you must include the lines 1000 through 10053 since they are necessary to draw a card. Before you draw any cards, execute the statement GOSUB 1000 to initialize the tool.

Important Variables

Variable	Function
cl	The column at which to print the card.
cx,cy	The actual screen position at which to draw the card. The variables cy and cx are calculated from the variables cl and rw.
no	The rank of the card to be drawn.
rw	The row at which the card is to be drawn.
su	The suit of the card to be drawn.

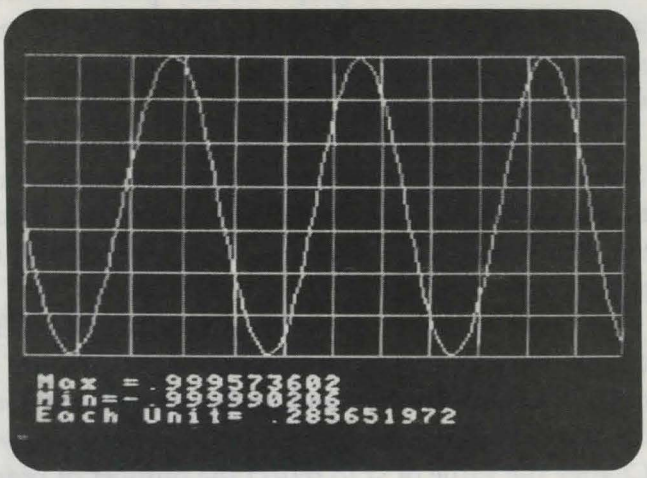
Program Description

Lines	Function
0	Shape table pointer.
1-95	Standard introduction.
100-150	Main program.
1000-1020	Subroutine to enter high-resolution mode and initialize the array.
2400-2450	Subroutine to read in the information on how to draw the numbers on the cards.
2700-2860	Subroutine to draw the card at a specified position on the screen.
8000-8130	Subroutine to load the shape data that contain the suit.
9103-10230	Data containing the suits.

```
  0 HIMEM :41000
  1 REM (c) 1984 by Brian Sawyer
  5 TEXT
 10 pm$ = "Card"
 15 FOR n = 1 TO 4
 20 PRINT CHR$(162)
 25 NEXT n
 30 tb = 17-INT(LEN(pm$)/2)
 35 PRINT TAB(tb); pm$
 40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
 45 PRINT " Hit Controller Button to Play"; CHR$(160)
 50 iv = 0: r = 0
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0)
    AND (PDL(9) = 0) AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 GOSUB 1000
105 cl = 0: rw = 1
110 FOR no = 1 TO 13
115 FOR su = 1 TO 4
120 cl = cl+1: IF cl > 7 THEN  cl = 1: rw = rw+1
125 IF rw = 3 THEN  rw = 1: HGR
130 GOSUB 2700
131 kl = kl+1: IF kl = 10 AND no = 13 THEN 133
132 IF kl <> 14 THEN 135
133 HOME: PRINT " Press any key to continue.":
    GET a$: kl = 0
135 NEXT su
140 NEXT no
150 TEXT: END
1000 HGR: SCALE  = 1: ROT  = 0
1010 DIM c(14, 20)
1015 GOSUB 8000: GOSUB 2400
1020 RETURN
2400 FOR n = 1 TO 13
2405 m = 0
2410 READ c(n, m)
2420 IF c(n, m) < 999 THEN  m = m+1: GOTO 2410
2430 NEXT n
2450 RETURN
2700 REM print card nu,su,cl,rw
2710 cx = (cl-1)*36+7: cy = (rw-1)*70+10
2715 HCOLOR  = 11: wd = 27
2720 HPLOT cx, cy TO cx+wd, cy
2722 HPLOT cx+wd, cy TO cx+wd, cy+55
2724 HPLOT cx+wd, cy+55 TO cx, cy+55
2727 HPLOT cx, cy+55 TO cx, cy
```

```
 2728 IF nf = 1 AND bt = 0 THEN  RETURN
 2729 f = 0: IF su > 2 THEN  f = 9
 2730 HCOLOR  = 15: IF su > 2 THEN  HCOLOR  = 6
 2732 ROT  = 0
 2733 DRAW su AT cx+5, cy+2+f
 2734 ROT  = 36
 2735 DRAW su AT cx+22, cy+53-f
 2800 REM passed no,cx,cy
 2801 HCOLOR  = 8
 2805 cx = cx+10: cy = cy+22
 2810 n = 0
 2815 HPLOT c(no, 0)+cx, c(no, 1)+cy
 2820 IF c(no, n) = 999 THEN 2860
 2830 x = c(no, n)+cx: y = c(no, n+1)+cy
 2840 HPLOT  TO x, y: HPLOT x, y
 2850 n = n+2: GOTO 2820
 2860 RETURN
 8000 qs = 4: pt = 16766: tl = 41000
 8005 of = qs*2+3
 8010 by = tl: GOSUB 8100
 8015 POKE pt, lo: POKE pt+1, hi
 8020 POKE tl, qs
 8030 FOR n = 1 TO qs
 8040 by = of: GOSUB 8100
 8050 POKE tl+n*2, lo: POKE tl+n*2+1, hi
 8060 READ a: POKE tl+of, a: of = of+1
 8065 IF a <> 0 THEN 8060
 8070 NEXT n
 8080 RETURN
 8100 hi = INT(by/256): lo = by-hi*256
 8130 RETURN
 9103 DATA 62,45,55,63,45,45,46,63,63,63,62
 9106 DATA 45,45,45,45,46,63,63,63,63,63,173
 9109 DATA 61,47,45,45,45,61,62,63,63,47,46
 9112 DATA 45,61,62,47,38,0
 9553 DATA 77,9,53,63,45,221,219,27,63,45,173
 9556 DATA 63,63,47,45,45,45,45,45,60,39,173
 9559 DATA 218,219,227,60,175,45,151,63,63,45,45
 9562 DATA 45,45,45,245,63,63,63,63,119,45,45
 9565 DATA 45,245,63,63,119,45,245,123,45,0
10003 DATA 36,36,36,36,36,53,54,54,54,54,52
10006 DATA 38,36,36,36,36,47,61,44,63,61,47
10009 DATA 62,53,54,61,45,47,47,61,37,53,62
10012 DATA 60,47,61,45,47,61,63,63,63,63,37
10015 DATA 54,52,52,37,52,0
10041 DATA 45,45,61,60,63,37,45,55,61,45,55
10044 DATA 45,63,60,36,38,60,63,62,39,45,45
10047 DATA 45,45,46,37,63,39,63,63,63,45,45
10050 DATA 45,45,39,63,63,63,37,45,45,39,63
10053 DATA 37,6,0
10090 DATA 0,0, 0,10, 0,0, 5,0, 5,10, 5,5, 0,5, 999
10095 DATA 0,0, 5,0, 5,5, 0,5, 0,10, 5,10, 999
```

```
10110 DATA 0,0, 5,0, 5,5, 0,5, 5,5, 5,10, 0,10 ,999
10120 DATA 0,0, 0,5, 5,5, 5,0, 5,10, 999
10130 DATA 5,0, 0,0, 0,5, 5,5, 5,10, 0,10 ,999
10140 DATA 5,0, 0,0, 0,5, 5,5, 5,10, 0,10, 0,5, 999
10150 DATA 0,0, 5,0, 5,10, 999
10160 DATA 0,0, 0,10, 5,10,5,0, 0,0, 0,5, 5,5, 999
10170 DATA 0,0, 5,0, 5,10, 5,5, 0,5, 0,0,999
10190 DATA 2,0, 2,10, 0,10, 10,10, 10,0, 6,0, 6,10, 999
10200 DATA 3,0, 9,0, 6,0, 6,10, 0,10, 0,5, 999
10210 DATA 0,0, 0,10, 3,10, 3,8, 3,12, 3,10, 5,10, 5,0,
          0,0,999
10220 DATA  0,0, 0,10, 0,6, 5,10, 0,6, 5,0, 999
10230 DATA  0,0, 0,10, 0,0, 5,0, 5,10, 5,5, 0,5, 999
```



Graph Maker

Program 30

This program will be useful to those who enjoy math. Graph Maker draws out the points from a mathematical function onto a grid on the screen of your Adam. In this example the function that is graphed is a sine function, which resembles a rounded wave. The minimum and maximum values of the function are displayed at the bottom of the screen. The amount that each unit—a square in the graph—represents is also shown. To clear the screen after the graph has been drawn, press any key.

Technical Description

In this program, the Adam's SIN function is used to generate the values for the sine wave. Although the SIN function is adequate for most applications, it is not perfectly accurate. A sine wave alternates between 1 and −1, but notice in this program that the wave never reaches these values, although it comes very close.

Most graphs are numbered in the horizontal and vertical directions so you can find the value of the graph at any point. However, it is impossible to print numbers directly onto the Adam's high-resolution screen. This is why the maximum and minimum values of the function are printed, as well as the value of a unit on the graph.

The array y contains the values of the function to be graphed on the screen. The size of this array is contained in the variable sz. This value can range from 2 to 255. In this example, 200 points are plotted, so sz is set to 200 in line 100. When the program is executed, it will plot the values in array y by connecting each point with a line.

In line 120, the array y is filled with 200 values from the sine function. You can modify the program to graph your own functions. Eliminate line 120, which contains the sine function, and replace it with the function you want graphed. You can use lines 111 through 129 to fill the array y with the values of the function. Change the value of sz to reflect the number of points to be plotted in your graph.

This program can also be used to create interesting backgrounds for your games. For instance, the sine wave could be used to show hills or waves in the ocean. Chances are you will not want the grid on your background. You can eliminate it by removing the command GOSUB 1300 in line 1115.

Important Variables

Variable	Function
min	Smallest value contained in function to be graphed.
max	Largest value contained in function to be graphed.
sz	The number of points produced by the function.
y(*number*)	Array containing the points of the function to be plotted. The size of the array is determined by the variable sz, so *number* equals sz.

Program Description

Lines	Function
1-95	Standard introduction.
100-170	Main program.
1000-1050	Subroutine to find the maximum and minimum values in the function.

1100-1120	Subroutine to enter high-resolution graphics mode.
1200-1230	Subroutine to plot the function on the graph.
1300-1370	Subroutine to draw the graph.

```
  1 REM  (c) 1984 by Brian Sawyer
  5 TEXT
 10 pm$ = "Graph Maker"
 15 FOR n = 1 TO 4
 20 PRINT CHR$(162)
 25 NEXT n
 30 tb = 16-INT(LEN(pm$)/2)
 35 PRINT TAB(tb); pm$
 40 FOR n = 1 TO 4: PRINT CHR$(162): NEXT n
 45 PRINT " Hit Controller Button to Play"; CHR$(160)
 50 iv = 0: r = 0
 55 NORMAL: IF iv = 1 THEN  INVERSE
 60 PRINT " Hit"; CHR$(160)
 65 c = 0
 70 c = c+1: r = r+1: IF (PDL(7) = 0) AND (PDL(9) = 0)
    AND (c < 40) THEN 70
 80 IF c = 40 THEN  iv = 1-iv: GOTO 55
 90 r = RND(-r)
 92 NORMAL
 95 CLEAR
100 sz = 200
110 DIM y(sz)
120 FOR n = 1 TO sz: y(n) = SIN(n/10): NEXT n
130 GOSUB 1000
140 GOSUB 1100
150 GOSUB 1200
160 GET a$
170 TEXT: END
1000 mi = 99999: max = -999999
1010 FOR n = 1 TO sz
1020 IF y(n) < mi THEN  mi = y(n)
1030 IF y(n) > ma THEN  ma = y(n)
1040 NEXT n
1050 RETURN
1100 HGR
1110 HCOLOR  = 1
1115 GOSUB 1300
1120 RETURN
1200 HCOLOR  = 1
1202 PRINT " Max ="; max
1203 PRINT " Min="; min
1204 PRINT " Each Unit= "; (max-min)/7
1205 FOR n = 0 TO sz-1
```

```
1210 x = 10+(240/(sz-1))*n: y = (140/(ma-mi))*(y(n+1)-mi)+10
1212 IF n = 0 THEN  HPLOT x, y
1215 HPLOT  TO x, y
1220 NEXT n
1230 RETURN
1300 HCOLOR  = 6
1305 FOR x = 10 TO 250 STEP 20
1320 HPLOT x, 10 TO x, 150
1330 NEXT x
1340 FOR y = 10 TO 150 STEP 20
1350 HPLOT 10, y TO 250, y
1360 NEXT y
1370 RETURN
```

Appendix A
Graphics Modes

The Coleco Adam Entertainer uses three principal graphics modes: text, low-resolution, and high-resolution. Each mode displays information in a different way, and each mode is suited to certain applications. This appendix contains a description of each of the modes and the graphics commands that are used in the programs.

Text Mode

In text mode only text that consists of letters, numbers, and symbols can be displayed on the screen. Text mode is used for typing in or editing programs, but it is also used for programs that display only text and do not require color or graphics. The introductory screen used by all the programs in this book operates in text mode.

The text screen is 31 columns by 23 rows, with each space containing one character. Text mode has a flashing line on the screen that is called the *cursor.* The cursor is the position where text will be displayed on the screen. The cursor can be positioned anywhere on the screen with the HTAB *column* and VTAB *row* commands. Use HTAB to move the cursor to a column and VTAB to a row. For example, if you wanted to position the cursor at row 3, column 10, you would include the statement HTAB 3:VTAB 10 in your program.

Use the PRINT command to display text on the screen. Any text printed on the screen is displayed in white on black. If the command INVERSE is set, any subsequently executed PRINT statements will appear as black on white. Use the NORMAL command to restore the display to white on black.

The Adam goes immediately into text mode after you load SmartBASIC. If the screen is in high-resolution or low-resolution mode, you can change to text mode by entering the command TEXT. When the TEXT command is executed, the screen is cleared and the cursor is moved to the home position.

Low-Resolution Mode

Low-resolution mode is used for games that require color but not a large amount of detail. It is also fast in displaying pictures, which is important for fast action games.

In low-resolution mode the screen is 40 columns by 40 rows. Unlike text mode, which displays letters, numbers, and symbols, a low-resolution screen can only display colors on the screen positions.

At the bottom of the screen is a *text window.* This is a 4-line area in which text messages can be displayed using the PRINT command. The area of the window can be erased and the cursor moved to the upper-left corner of the window by using the HOME command.

To enter the low-resolution mode, use the GR command. There are sixteen different colors that can be drawn in the low-resolution mode. The COLOR *N* command sets the color that a position will be drawn in. The value of *N* ranges from 0 to 15. Once the color is set, it will not change until another COLOR command is executed. Initially the color is 0, or black. Here is a table of the colors and the numbers they correspond to:

Number	Color
0	Black
1	Magenta
2	Dark Blue
3	Dark Red
4	Dark Green
5	Grey 1
6	Medium Green
7	Light Blue
8	Light Yellow

9	Medium Red
10	Grey 2
11	Light Red
12	Light Green
13	Light Yellow
14	Cyan
15	White

The PLOT *X,Y* command will draw the position at column *X* and row *Y*. If you wanted the position at row 20, column 38 to be drawn with the color dark blue, you would first execute the command COLOR 2 and then execute PLOT 20,38.

The low-resolution mode has two commands that will draw lines: The command VLIN *Y1,Y2* at *X* causes a line to be drawn vertically from row *Y1* to row *Y2* along column *X;* the command HLIN *X1,X2* at *Y* draws a horizontal line from column *X1* to column *X2* along row *Y*.

Many of the action games require the program to detect collisions of objects. This can be easily done by using the SCRN (*X,Y*) command to find the color of the position at column *X* and row *Y* and thus determine if it is occupied. For example, a program needs to know the color of the position at row 23, column 30 in order to determine whether an object, which has the color of violet, is occupying that position. The program can find out by executing the command SCRN (23,30), and if it returns a value of 2, which is the value of violet, the object occupies the position.

High-Resolution Mode

In high-resolution mode the screen contains 256 columns by 160 rows. Each position on the high-resolution screen is a tiny dot called a *pixel.* At the bottom of the screen there is a text window, similar to the one in the low-resolution mode, that can display four lines of text using the PRINT command. To clear the text window and move the cursor to the top left-hand corner of the window, execute the HOME command.

To enter the high-resolution mode, use the command HGR. Sixteen different colors can be displayed in the high-resolution mode. The command HCOLOR *N* sets the color that a pixel will be drawn in. The value of *N* can range from 0 to 15. Here is a table of the colors and the numbers that they correspond to:

Number	Color
0	Black 1
1	Green
2	Violet
3	White 1
4	Black 2

5	Orange
6	Blue
7	White 2
8	Brown
9	Dark Blue
10	Grey
11	Pink
12	Dark Green
13	Yellow
14	Aqua
15	Magenta

To light a pixel on the screen, use the command HPLOT *X,Y*. This lights a pixel in the current color at column *X*, row *Y*. The HPLOT TO command can be used to draw lines between two points on the screen. HPLOT *X1,Y1* TO *X2,Y2* draws a line from the point at column *X1*, row *Y1* to column *X2*, row *Y2*.

In the high-resolution mode it is not possible to check the color of a space on the screen, as can be done in the low-resolution mode. High-resolution mode does offer shape tables, a useful way of displaying images on the screen. Shape tables are discussed in detail in Appendix B.

Appendix B:
Shape Tables

Shape tables will draw detailed graphic displays on the screen. A shape table is a list of numbers that define shapes that can only be drawn on the Adam's high-resolution screen. With SmartBASIC commands, these shapes can be colored, rotated, expanded, and erased. Many of the programs in this book use shape tables; for example, a shape table is used in World Conquest to draw the continents and in Fruit Detective to draw the fruit. This appendix explains the design and implementation of shape tables.

Designing a shape table requires three stages: creating a shape, converting it into numbers, and storing it in the Adam's memory. There are two ways in which you can design a shape table: by hand, or with the aid of a program. This book contains a program called Shape Maker that allows you to design a shape on the screen by using the joystick. The shape will then be converted into DATA statements that you can use in your programs. Using Shape Maker does not require a knowledge of how to represent the shape numerically or how to load a shape into memory.

The other method of designing a shape table is by hand. This is a more difficult and complex approach, but it reveals exactly how shapes are represented in the Adam's memory. Designing a shape by hand requires knowledge of how the Adam's memory is structured, the ability to convert binary numbers to decimal numbers, and the notion of a high-order and a low-order byte. If you are not knowledgeable in these areas and you want to create shapes, you should use the Shape Maker program. A later section of this appendix entitled "Commands to Print a Shape" will teach you how to print a shape.

Designing a Shape

A shape table can be thought of as a series of instructions that direct an imaginary pen on the screen. There are four possible directions that the pen can be moved: up, down, left, or right. When the pen moves in one of these directions, it will either plot by lighting a pixel (one dot on the high-resolution screen), or *not* plot by doing nothing to the pixel. A total of eight instructions are possible by combining the movements of the pen and plotting. One instruction would be to plot and then move left, or another instruction would be not to plot and then move left.

The idea behind *not* plotting while moving is that the pen can be maneuvered without ruining what is already on the screen. By giving the pen a series of instructions to move and plot, a shape can be drawn. A simple example would be to draw a box with these four commands: plot and then move right, plot and then move down, plot and then move left, plot and then move up. These movements are illustrated in Figure B-1.

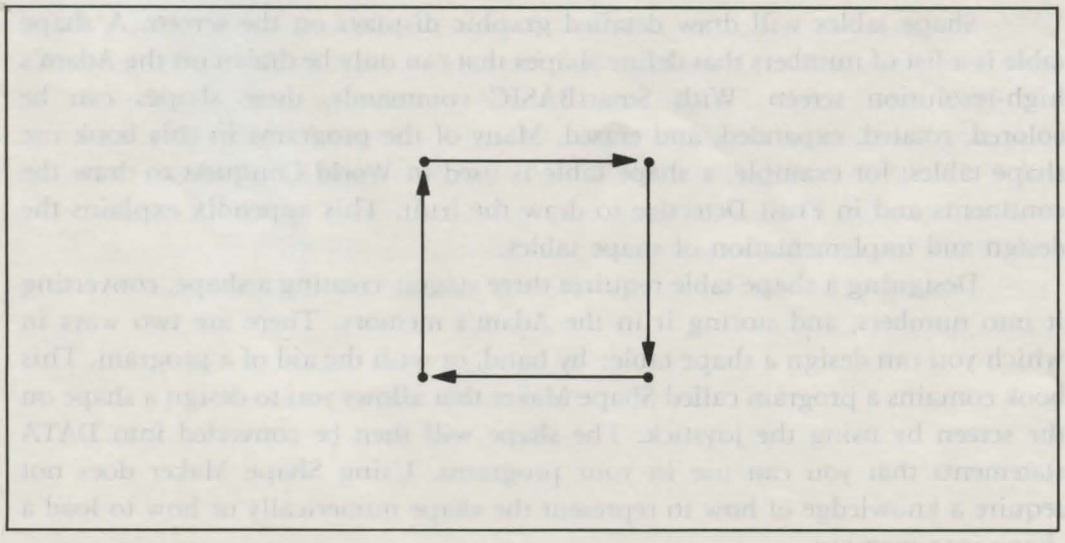


Figure B-1. Steps in drawing a square

Table B-1. Pen Instructions and Their Binary Codes

Symbol	Action	Binary Code	Decimal Code
↑	Move up without plotting	000	0
→	Move right without plotting	001	1
↓	Move down without plotting	010	2
←	Move left without plotting	011	3
↑	Move up with plotting	100	4
→	Move right with plotting	101	5
↓	Move down with plotting	110	6
←	Move left with plotting	111	7

Draw your shapes on graph paper by having each square on the paper represent a pixel on the screen. Plan the series of pen actions that will draw your shape the most efficiently. An inefficient sequence will require more pen actions than are necessary. Try to minimize the amount of backtracking that the pen has to do.

Your series of pen actions must be converted into numerical form so they can be stored in the Adam's memory. Each cell of the Adam's memory is made up of an *8-bit word,* or *byte.*

Each of the eight possible pen instructions corresponds to a 3-bit binary representation, which is called the *binary code.* For example, the binary code for the instruction to move down without plotting is 010. Consult Table B-1 to find the binary codes for each of the pen instructions.

Each byte can contain as many as three different pen instructions. Each byte is divided into three sections: section 1 contains bits 0 to 2, section 2 contains bits 3 to 5, and section 3 contains bits 6 and 7. Each section can store the binary code for a single pen action (see Table B-2). Fill up each section in the byte beginning with section 1. Notice that section 3 can only contain two bits, although the binary codes are three bits. Section 3 is considered a special case, since it can only hold one of four instructions. In section 3 the binary code indicates the following: 00 for no pen action, 11 for moving left without plotting, 10 for moving down without plotting, and 01 to move right without plotting.

Table B-2. Shape Table Byte

Bit	Section 3		Section 2			Section 1		
	7	6	5	4	3	2	1	0
M = Movement bit P = Plot/No Plot bit	M	M	P	M	M	P	M	M

Continue converting the binary codes in your instruction sequence into bytes until all your pen actions have been converted. To indicate the end of your shape, include a byte filled entirely with zeros at the end of the series. Now go through these bytes and convert them from binary to decimal. If the decimal contents of a byte have a value over 255, you have made a mistake since the value of a byte can only range from 0 to 255. After you have converted all of the bytes to decimal, you should have a series of numbers from 0 to 255 ending in 0. This is your shape definition. The best way to store these numbers in your program is with DATA statements.

When you execute your program, it needs to load the shape definitions into memory. The first address in memory where your shape table will reside is called the *base address.* All of the programs in this book that use shape tables have location 41000 as the base address since this area in memory is unoccupied except for very large programs. To reserve a base location for the shape table, use the command HIMEM:*memory-location*.

The shape table itself consists of two parts: the shape directory and the shape definitions. The bytes that immediately follow the base address are of the shape directory. In descriptions of the function of these memory locations, the term displacement will refer to the distance from the base address. The byte at displacement 0, which is the base address, contains the number of shape definitions in the shape table. The next byte, at displacement 1, which is the next address after the base address, is unused and should contain a 0. Displacements 2 and 3 contain the low-order and high-order bytes, respectively, that when added to the base address will give the locations of the first shape definition. The next two displacements, 4 and 5, contain the low-order and high-order bytes for the second shape, if there is a second shape. The pattern continues until all shapes have been accounted for. Consult Figure B-2 to see how the shape table is organized in memory.

After the shape directory comes the shape definitions. Load your shapes from the DATA statements into memory with the POKE command. Be sure that the program loads the shape definitions after the shape directory; and leave enough room between each shape so that if any modifications are necessary to a shape, they will not interfere with any other shape in memory. When all information has been loaded into the shape table, you must set the Adam's shape table *pointer* to point to the base address. This pointer resides at locations 16766 and 16767, which contain the low-order and high-order bytes of the shape table base address respectively. Use the POKE command to store the base address in these locations.

Commands to Print a Shape

Before any shape can be drawn, the program must be in the high-resolution mode. This can be done with the HGR command. The fundamental shape table command in SmartBASIC is DRAW. Executing DRAW *N* AT *X,Y* causes shape number *N* to be displayed at column *X*, row *Y*.

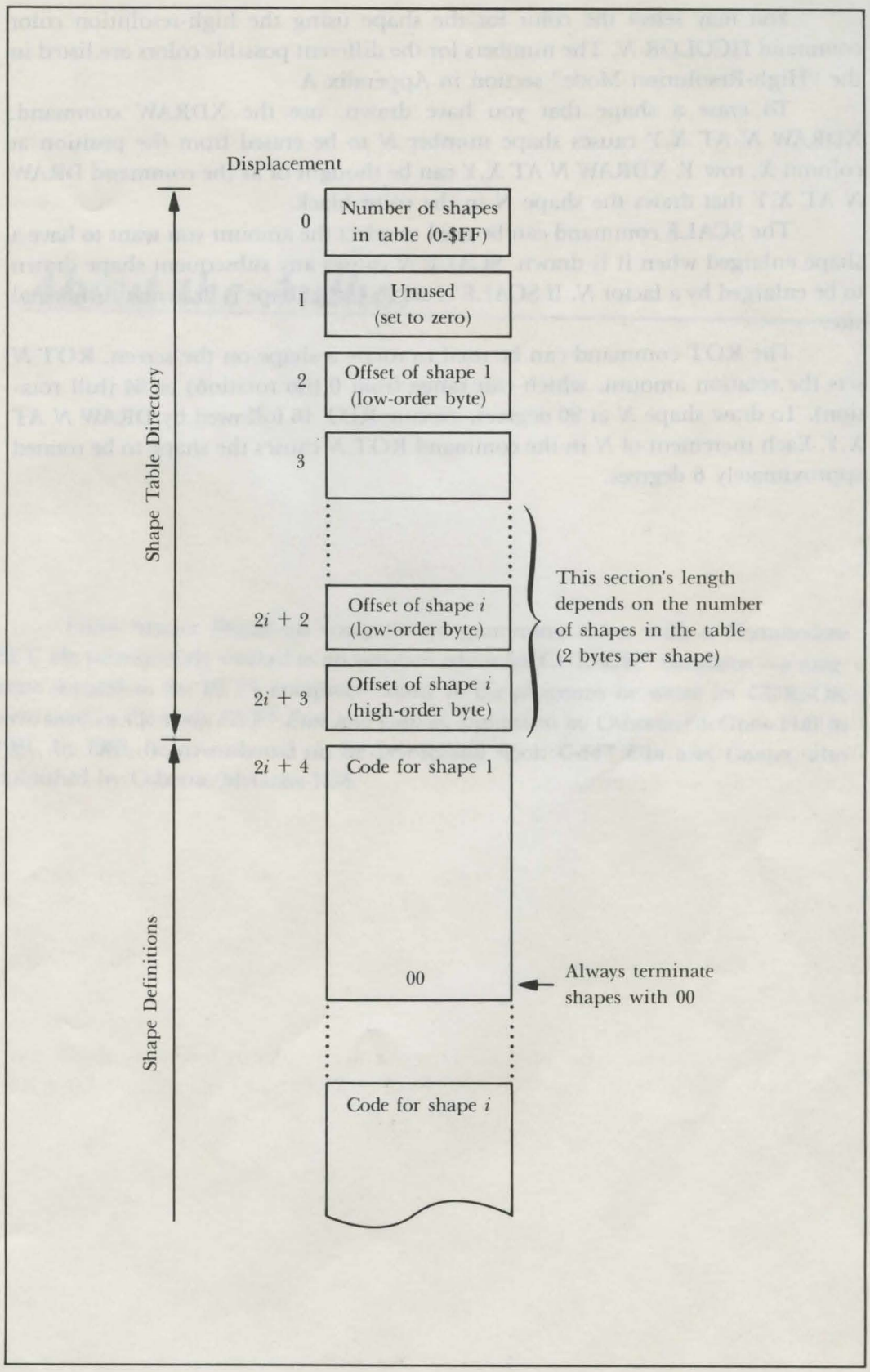


Figure B-2. Shape table organization

You may select the color for the shape using the high-resolution color command HCOLOR *N*. The numbers for the different possible colors are listed in the "High-Resolution Mode" section in Appendix A.

To erase a shape that you have drawn, use the XDRAW command. XDRAW *N* AT *X,Y* causes shape number *N* to be erased from the position at column *X*, row *Y*. XDRAW *N* AT *X,Y* can be thought of as the command DRAW *N* AT *X,Y* that draws the shape N in the color black.

The SCALE command can be used to select the amount you want to have a shape enlarged when it is drawn. SCALE *N* causes any subsequent shape drawn to be enlarged by a factor *N*. If SCALE is set to 1, the shape is drawn at its normal size.

The ROT command can be used to rotate a shape on the screen. ROT *N* sets the rotation amount, which can range from 0 (no rotation) to 64 (full rotation). To draw shape *N* at 90 degrees, execute ROT 16 followed by DRAW *N* AT *X,Y*. Each increment of *N* in the command ROT *N* causes the shape to be rotated approximately 6 degrees.

About the Author

Brian Sawyer began his computer programming career with a Commodore PET. He subsequently worked as an assistant editor for CURSOR™ magazine—a magazine devoted to the PET® computer. Many of the programs he wrote for CURSOR were used in the book *PET® Fun and Games,* published by Osborne/McGraw-Hill in 1981. In 1983, he co-authored of the best-selling book *C-64™ Fun and Games,* also published by Osborne/McGraw-Hill.

The Coleco™ Adam™ Entertainer

• Games • Graphics • Sound

Hours of fun-filled challenge and entertainment can be yours with this collection of 30 programs written especially for the Coleco™ Adam.™ Included are many games and informative programs that make the best use of your Coleco Adam's special color and graphics capabilities.

In addition to playing action-packed games, you can create greeting cards, banners, and designs with the programs listed in **The Coleco™ Adam™ Entertainer.** Photographs of each program's run are included, together with complete instructions and listings written in Coleco Adam's programming language SmartBASIC.™

Explanations for each program in this book teach you fundamental programming techniques. Whatever your level of experience, this delightful book is the ideal companion for you and your Coleco Adam!

ISBN 0-88134-134-7

McGraw Hill