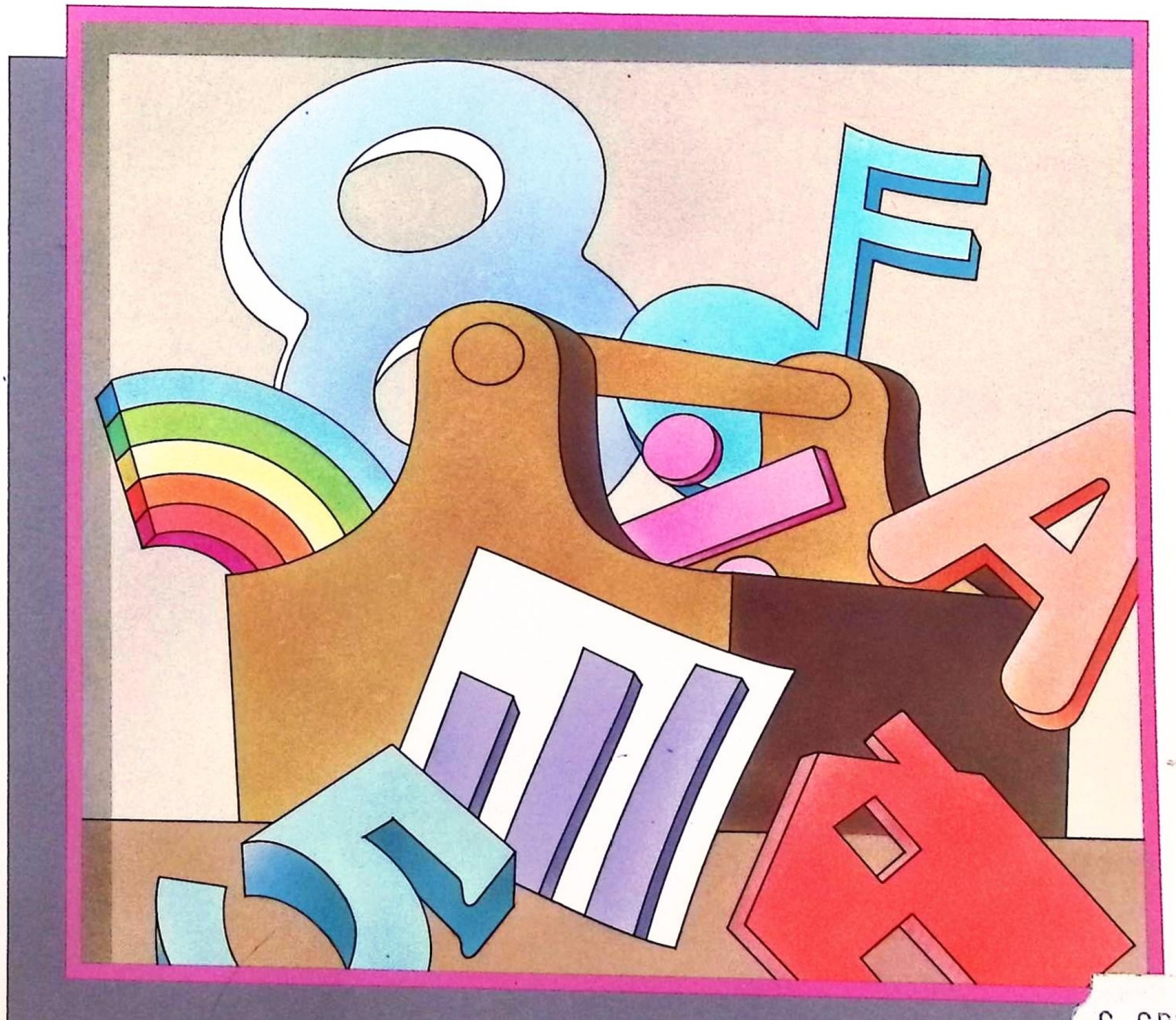
SAMS 22312

The Tool Kit Series: Coleco Adam™ Edition

Ted Buchholz and Dave Dusthimer



THE TOOL KIT SERIES

COLECO ADAM™ EDITION

Notice

Howard W. Sams & Co., Inc. wishes to advise potential readers of some of the circumstances affecting the Coleco Adam™.

Between the time that this book was written and the time that it was published, Coleco Industries, Inc. decided to quit manufacturing the Adam™. The planned distribution of the remaining Adam™ computers and the software that is available for the machines is not known by this publisher.

We understand that the 90-day warranty for the Adam™ will continue to be in effect and that Honeywell will provide repairs for the machine. Whether any other companies will support the machines presently in use is not known.

The Tool Kit Series Coleco Adam™ Edition

by
Dave Dusthimer and Ted Buchholz

Howard W. Sams & Co., Inc.
A Publishing Subsidiary of ITT
4300 West 62nd Street, Indianapolis, Indiana 46268 U.S.A.

FIRST EDITION
FIRST PRINTING—1985

International Standard Book Number: 0-672-22312-0
Library of Congress Catalog Card Number: 84-71059

Edited by: *Frank N. Speights*
Illustrated by: *David K. Cripe*

Printed in the United States of America.

Preface

Thousands of home computers are collecting dust, because their owners don't have the tools to make them work properly. All home appliances require tools to keep them humming along and your Coleco Adam™ is no exception. In this book, you will find all the tools you need to make the Adam™ perform the way it should.

This book grew out of a need we noted after buying our first computer. Most manuals and books approach BASIC programming by teaching statements, commands, and program data structures. This approach can be successful, but the average person is not motivated to learn program construction this way. After spending many frustrating evenings with User's Manuals and BASIC programming books, we still could not write the simple programs we had in mind when we bought the computers. But, finally, we discovered a way of learning to program that really works. By looking at programs in terms of their working parts—their subroutines—we were able to understand how to write simple programs and soon were designing our own games and quizzes. With the help of this book, you can too.

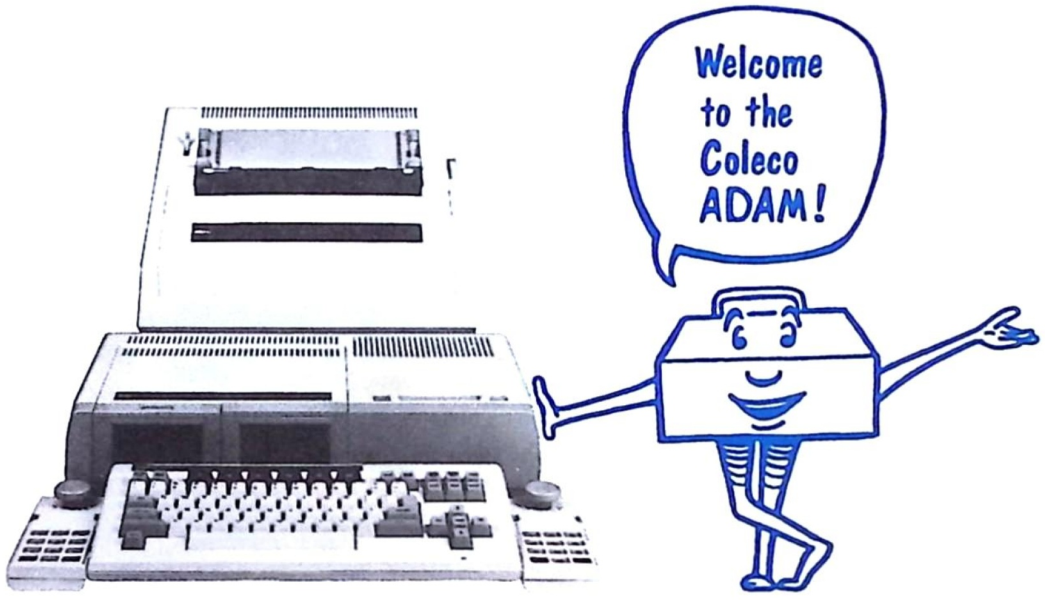
The "tools" in this Tool Kit book are brief 5- to 30-line subroutines. Color, sound, graphics, animation, and computational subroutines are the "tools," and, when combined, form the basis for a variety of educational programs and computer games. Each line of each subroutine is carefully described and explained. We will tell you what each subroutine will do, how it can be changed, what the variables control, and will give hints for further experimentation. This approach makes programming easier, less frustrating, and a lot more fun. The modular form of programming is a sound method of structured programming (and acknowledged so in computer science) and it works well for the average microcomputer owner.

This book will also help parents teach their children to program. Both of us have small children and the techniques used in this book have helped our children become computer literate. Building programs out of subroutines gives you, the user, quicker and more satisfying results than the traditional approaches. Since the subroutines are mostly educational and recreational, they will appeal to both parents and children.

In the Tool Kit books, we share a valuable approach to building simple programs with the beginning computer user. Now, let's get to it and have some fun.

DAVE DUSTHIMER
AND TED BUCHHOLZ

Contents



The Coleco Adam™ 1

All computers are the result of genius-level engineering. Fortunately for us "normal" people, you needn't be a genius to use one. Computers exist to serve their users. But since the Adam can't think like we do, we must learn to think like it does. Don't worry, it isn't hard once you understand how the Adam works.

You have an Adam, right! Well, now you have the ADAM TOOL KIT.

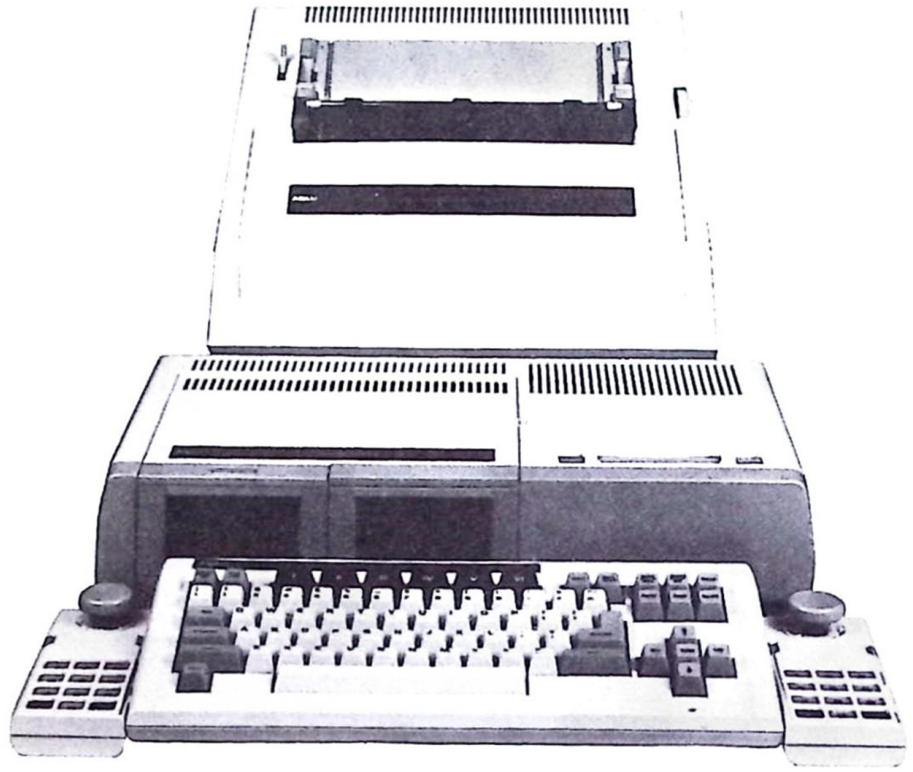


Fig. 1-1. The Adam system.

WHAT MAKES THE ADAM WORK?

At the core of every computer is its Central Processing Unit, or CPU, which controls, interprets, and executes the computer functions. The Adam's CPU is a Z-80® microprocessor. This microprocessor "chip" is very tiny, but it stores a great deal of information and is very powerful.

In a binary system, computers can only receive information through their CPU. That is, computers only know two things—whether data is "on" or "off," "0" or "1," or "yes" or "no." The digits (0 and 1) readable by the computer are known as *bits*. The Adam is called an 8-bit machine because each of its characters is composed of 8 bits.

We usually consider those bits put together to form a character as being a *byte*. Knowing the number of bytes available in the machine is another way of knowing just how much memory or space is available for use.

HOW MUCH MEMORY DOES THE ADAM HAVE?

There are two types of memory in the Adam computer—read-only memory (ROM) and random-access memory (RAM). The ROM is programmed into the Adam hardware by the manufacturer. The CPU controls computer operations by addressing programs in ROM (read-only memory). The ROM is permanent in the computer and we, as users, cannot use ROM for storage.

The random-access memory (RAM) is a temporary storage memory that serves as space for the user's programs. The unexpanded Adam (the standard Adam without additional memory) has 80 kilobytes of memory storage available. You can expand the Adam's RAM up to 144 kilobytes by use of the 64K memory expander. The longer your programs, the more memory or RAM you will need.

Unless you save (on the digital data pack) the data that is in RAM, the data disappears when the computer is turned off. The standard 80K of memory is more than enough to handle all of the programs contained in this book.

THE BINARY SYSTEM

If the computer can only understand "0" and "1" in the binary system, how can you communicate with it?

Fortunately, there is an operating system that interprets this binary machine code into something that is more easily understood by humans. The operating system for the Adam is a BASIC Interpreter, which allows us to communicate using SmartBASIC™ as a programming language. We input data principally by using the keyboard, but we can also use the keypad controller or any other similar device. Data is inputted using SmartBASIC™ as a language so the machine can understand it.

HOW DO YOU WRITE PROGRAMS IN BASIC?

Programs are written with commands, statements, and a structure that is governed by rules not so different from the grammar and syntax rules used in English. We're lucky because SmartBASIC™ is easier to learn than a foreign language. We will introduce you to the major commands and statements in the next chapter.

WHAT ACCESSORIES ARE NEEDED TO GET THE MOST OUT OF THE ADAM?

The Adam is a complete computer system. It comes complete with keyboard, two keypad controllers, a high-speed Digital Data Pack drive (the Memory Console), and a letter-quality SmartWRITER™ printer.

You can add a second data pack drive if you like. There are also many game expansion modules that are available from Coleco. You can add extras to the Adam as long as your money holds out, but you don't have to spend a fortune to enjoy your computer. We used the Adam just as it came from Coleco to do all of the programs in this book.

WHAT IS THE BEST WAY TO START LEARNING HOW TO USE THE ADAM?

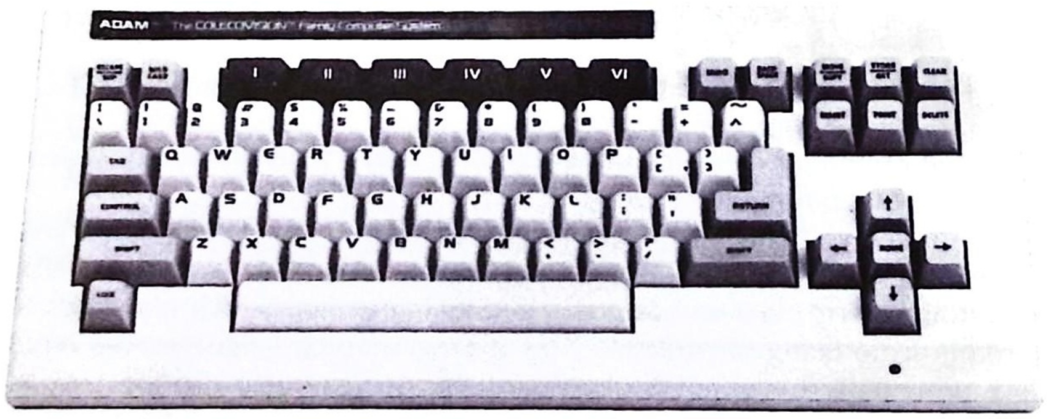


Fig. 1-2. Close-up of the Adam keyboard.

Familiarize yourself with the chief input device for the Adam—the keyboard.

If you have done any typing in the past, you will be comfortable with the Adam keyboard. All of the *light tan keys* control the standard alphanumeric characters. These light tan keys are organized exactly like a standard typewriter. But, let's look at the "special" keys on the Adam's keyboard.

Smart Keys

The large dark brown keys at the top of the keyboard are called Smart Keys. The Smart Keys are numbered from I to VI. These keys allow you to interact with the menu options when you are using the SmartWRITER word-processing system. Adam tells you which key to press for specific word-processing options. The prompts appear in boxes at the very bottom of the word-processing screen display.

The *Adam Word Processing* manual explains these options in detail.

Other Special Keys

There are several other "special" keys. These keys are a darker tan than the alphanumeric keys. Let's look at these keys in detail.

TAB—This key will move the cursor to the preset tab or column location.

SHIFT—There are two of these keys, but they both perform the same function. When the shift key is pressed with a letter key, the Adam prints a capital or uppercase letter. When the shift key is pressed along with a number or punctuation key, the character at the top of the key is displayed. For example, if you press SHIFT and the number 1 key, the Adam will display an exclamation point.

LOCK—This key will lock the SHIFT function. After LOCK is pressed, all characters will be displayed in the uppercase mode. All uppercase punctuation characters are also affected. To disengage the LOCK function, simply press the LOCK key again and the lowercase key format will be displayed.

CONTROL—This key activates certain command features. This key will also be used to control future expansion devices.

> **CONTROL** and the **C** key—This combination will break or stop a running program.
>
> **CONTROL** and the **H** key—These two keys will backspace over previously displayed characters without deleting the characters.
>
> **CONTROL** and the **L** key—This combination will clear the screen. The buffer or program in memory is not affected by pressing these keys.
>
> **CONTROL** and the **N** key—This combination allows you to insert letters or characters into the program wherever you like.
>
> **CONTROL** and the **O** key—When these keys are pressed, you are able to delete or erase information.
>
> **CONTROL** and the **P** key—These keys will print out onto the printer whatever is on the screen.
>
> **CONTROL** and the **S** key—When you list a program, pressing these keys will start and stop the listing procedure so that you can examine the listing.
>
> **CONTROL** and the **X** key—These keys will perform the same function as the UNDO key. It tells the computer to disregard a program line.

WILD CARD—We didn't use this key in this book. It will be used to control future expansion devices.

ESCAPE/WP—This key will activate Adam's word-processing system. Pressing this key while in the typewriter mode will turn on the SmartWRITER printer. Pressing this key while in the SmartWRITER mode clears the work space and returns you to the typewriter mode.

UNDO—This is a control key for SmartWRITER. If you press UNDO after DELETE, CLEAR, or BACKSPACE, it returns the text erased by those keys. It allows you to get back text removed by DELETE, CLEAR, or BACKSPACE.

BACKSPACE—This key moves the cursor one space to the left. This key will be helpful when you want to correct a mistake.

SmartWRITER Keys

In the upper right-hand corner of the keyboard, you will find a bank of six word-processing keys. These keys will only be used in the word-processing mode, and will usually be used with one of the Smart Keys.

MOVE/COPY—This key moves characters from one location to another in the text. This key can also be used to copy information. See the *Adam Word Processing* manual for details.

STORE/GET—Use this key to store or retrieve text from a Digital Data Pack. See the manual for details.

INSERT—If you want to insert additional text into your document, you will use this key.

PRINT—The PRINT key and the appropriate Smart Key will print your text on the SmartWRITER printer.

DELETE—This key will erase text from the screen or from a work space.

Cursor Directional Control

In the lower right-hand corner of the keyboard, you will find the cursor control keys. The HOME key will send the cursor directly to the upper left-hand corner of the screen. The four arrow keys control the direction of the cursor—either up, down, right, or left. These keys will be helpful during editing.

Inputting Procedures

If you read your *User's Manual*, you already know that entering data is just a matter of typing, except that it is easier to make corrections on the Adam computer. Here are some simple inputting rules and suggestions.

1. Always number your program lines. When you number your program lines, we suggest numbering in increments of at least 10. This gives you space in which to alter or change a program.
2. Get in the habit of using REM or Remark statements as notes to yourself.
3. Be aware of punctuation marks. If you fail to use them properly, the programs won't run.
4. Use the LIST command. As your programs get longer, you will need to know how to list certain segments of the program.

Editing Before the Line Is Entered

Using the BACKSPACE key, move the cursor to the error you found and simply type over the error with the correct character. You must also retype any character that you backspace over. If you don't, the line will not enter properly. Try this:

```
1Ø PRINT "MY ADAMM IS VERY SMART"
```

Don't press RETURN yet.

1. Using the BACKSPACE key, move the cursor to the second M in ADAMM.
2. Hit the SPACEBAR and the M will disappear.
3. Now, retype the rest of the line.
4. Any unwanted character can be moved by hitting the SPACEBAR.

Editing After the Line Is Entered

Once a line is entered you must edit in a different way.

If you want to remove the line entirely, simply enter the line number and hit RETURN. When you list the program, the line will have disappeared.

If you want to correct the line, use the same line number and type in the line correctly. When you press RETURN, the corrected line will replace the incorrect line.

THE ADAM AS A CALCULATOR

Your Adam computer is, by its very nature, a number cruncher. It can do addition, subtraction, multiplication, and division, as well as a variety of algebraic and trigonometric functions.

To use the Adam as a calculator, all you need to do is enter Adam's computer mode and then use the PRINT statement and a number sentence. For example, to do addition, type in:

```
 PRINT 2 + 2  (press return)
4
```

For subtraction, type:

```
 PRINT 6 - 4  (press return)
2
```

Easy isn't it? Now, for multiplication, enter

```
 PRINT 5 * 2  (press return)
1Ø
```

The Adam knows that it is to multiply when it sees the asterisk, so you always use the asterisk as the multiplication sign. For division, type in:

```
 PRINT 1Ø / 2  (press return)
5
```

The Adam divides when it sees the slash.

You can experiment with the math capability of the Adam as much as you like, but before you perform problems that include different mathematical operations, you need to know how the Adam orders problems. Here is the Mathematical Hierarchy of the Adam:

1. All expressions within parentheses are evaluated according to hierarchical rules.
2. Exponents are acted upon in order from left to right.
3. Prefix plus and minus actions are performed.

4. Multiplication and division are done.
5. Addition and subtraction are completed.

All of these functions are performed from left to right in a line. We will spend more time on mathematical operations in Chapter 7.

VARIABLES

A variable is something that is assigned a value. We will assign values to variables in most of the programs in this book, so you will need to understand the concept.

If we say that X = 1, we are defining the variable *X*. If we put X = 1 in a program, the computer will interpret X as 1 every time it reads X. Variables can be used to help us perform mathematical operations. Our X = 1 is a *numeric variable*. We could use this variable in the following way:

```
10 x = 1
20 y = 10 * x
30 PRINT x
40 PRINT y
```

We have stated two variables, X and Y, in terms of each other; 10 X's equal 1 Y.

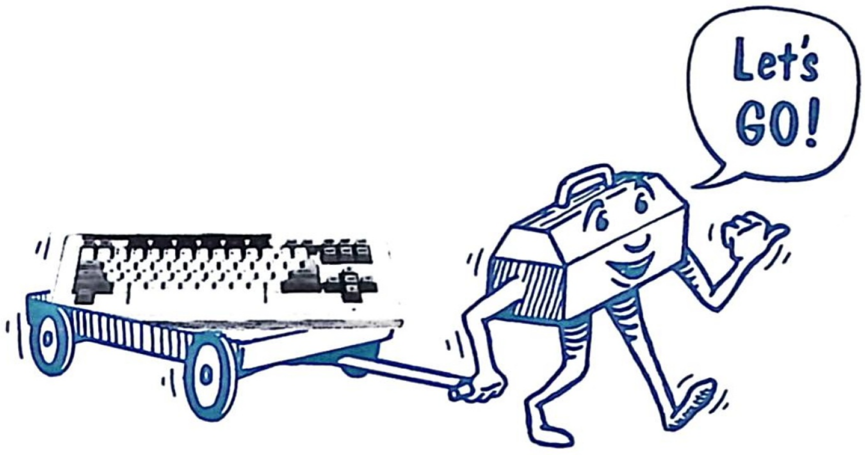
We can also state a variable in terms of a string. This type of variable is called, oddly enough, a *string variable*. If we want to represent a word with the letter *X*, we tell the computer to expect a string by adding a $. Let's say we are designing a "States and Capitals" game and we want a message to indicate an incorrect answer. We can say:

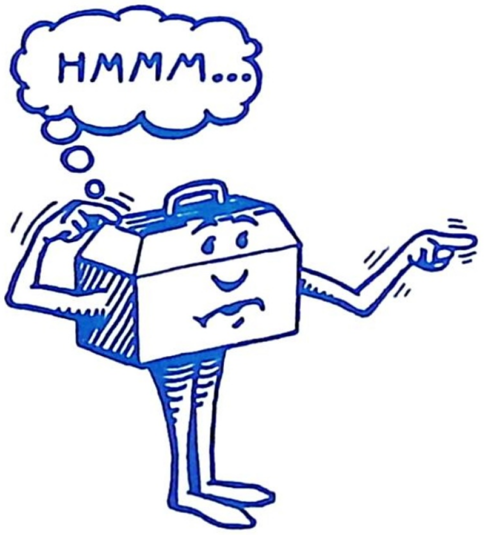
```
10 x$ = "WRONG"
20 PRINT x$
```

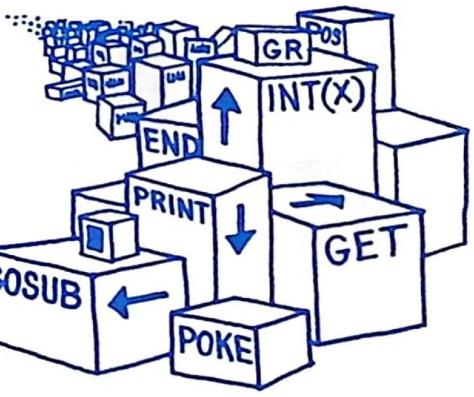
We are telling the computer that X$ means the same thing as the letters WRONG.

As we begin working with programs in Chapter 3, you will see how valuable variables can be. They can state the value of nearly anything. We will identify the variables for you as we move on. But remember that there are two types of variables—NUMBER and STRING.

Now that we understand how the computer works and how to input data, let's learn how to speak its language—SmartBASIC.







The Building Blocks of Adam BASIC

2

Once you learn how the Adam operates and how to speak its language, learning to program will be easy. In Chapter 1, we explained how the Adam works; now we are going to teach you to speak SmartBASIC. Remember that the Adam is a "dumb" machine. It can't do anything unless you, the user, command it to act. Before you get carried away with feelings of great power, you must also remember that the Adam can't speak English (remember this from Chapter 1?) so you must give your commands in a language that the computer understands.

SmartBASIC

Your Adam computer speaks SmartBASIC. However, your machine has a very limited vocabulary. The Adam only understands about 100 commands or functions. In the next few pages, we will explain each command and give you an example of how the command works. You will refer to this part of the book often, so you may want to put some notes in the margin. Now, let's learn how to talk to the Adam.

COMMANDS AND STATEMENTS

The following commands are fairly standard BASIC commands. Variations on these commands are used to communicate with most computers that understand BASIC.

CATALOG—This command will print out a listing of the file names stored on a digital data pack.

CLEAR—This command erases all variable values. All numeric variables are set to 0 and all string variables are erased.

CLOSE—CLOSE is used to close a file when you are finished working with it. This command keeps you from accidentally writing over stored information.

COLOR=—Defines the display color in the low-resolution mode. There are 16 colors available for use in the low-resolution mode.

CONT—This command will restart a program that has been halted by either an END statement or a CONTROL-C input. The program will begin where it left off.

DATA—This command allows you to store information in either string or number form. You must tell the computer to read the DATA with a READ command. Data can be stored in a program by simply typing:

```
DATA  INFO,INFO,INFO
```

and so on.

DEF FN—This statement allows you the opportunity to *define* your own functions for the needs of a specific program.

DEL—This command deletes or removes program lines from your program. The command must be followed by the numbers of the lines you want to delete.

DEL 10,60 will delete lines 10 through 60.

DEL 10 will delete line 10.

DEL 60 will delete line 60.

DELETE—This command will delete or erase an unlocked file.

DIM—Allows you to name and set up an array of numbers with X dimensions, number 1 through number Y. The statement must look like this:

$\texttt{DIM A(NUMBER}_x\texttt{, NUMBER}_Y\texttt{)}$

A string can be used to fill the array if the string symbol ($) is included behind the A.

DRAW—This is a high-resolution statement that will draw a user-defined character or shape at the desired screen location (see Chapter 5).

END—Stops the program.

FOR...TO...STEP—Loops are easily created with this statement. The NEXT statement must always be used with FOR...TO...STEP or you will receive an error message. FOR and TO are used to state a variable. For example:

```
5 FOR a = 1 TO 5
```

STEP can be used if you want the computer to "step" or cycle in increments greater than 1. If you want the computer to move in steps of 1, then you can leave the step portion of the command out. By using STEP and a negative number you can create a negative step. You must include the NEXT statement so that the computer will go back and act on the next variable in the FOR...TO...STEP command. Try this:

```
5 FOR i = 1 TO 5
1Ø PRINT "YOUR NAME"
15 NEXT i
```

FRE—This function will tell you how many free bytes of memory you have to work with at any given time.

GET—This statement gets input information from the user. The program waits for the user to press a key, but no question mark prompt appears and the user does not have to press RETURN.

GOSUB—This is one of a couple of branching commands. You must use a RETURN statement to tell the computer to return to the main body of the program. The RETURN statement will send the computer to the line right after the GOSUB statement. GOSUB must be followed by the program line to which you want to branch. For example:

`GOSUB 1000` (Will send the computer to line 1000.)

GOTO—This is another branching statement. Following the GOTO statement, you must input the line number that you want the computer to read next. Once you use a GOTO statement, a continuous loop is made. To break the loop, another GOTO statement can be used to send the computer to another line in the program outside of the loop. A

FOR...NEXT loop can also be used to test for a specific condition and then break the loop when the defined condition exists.

GR—When you want to enter the low-resolution graphics mode, use this command.

HCOLOR=—Defines the display color for high-resolution plotting.

HGR—This command activates the high-resolution text and graphics screen. This screen has 256 columns and 159 rows of graphic space, and a 4-row text screen at the bottom.

HGR2—Use HGR2 to activate the high-resolution graphics screen. This screen contains 256 columns and 192 rows.

HLIN—This command draws a horizontal line when the user provides column and row coordinates. The formula for HLIN looks like the following entry. This command will only work in the low-resolution mode.

```
HLIN Beginning Column Number, Ending Column
Number AT Row Number
```

HOME—This command clears the screen and moves the cursor to the upper left-hand corner of the screen.

HPLOT—Points and lines can be plotted in the high-resolution mode using this command. The HPLOT command requires a "column,row" coordinate to function. The formula for HPLOT is:

```
HPLOT Column#,Row#
```

HTAB—Sets the horizontal tabs much like on a typewriter. The command must be followed by a column number.

IF–GOTO—This is a very useful branching command. If a certain condition exists, the program will branch to a specific line number. For example:

```
140 IF x = 7 GOTO 240
```

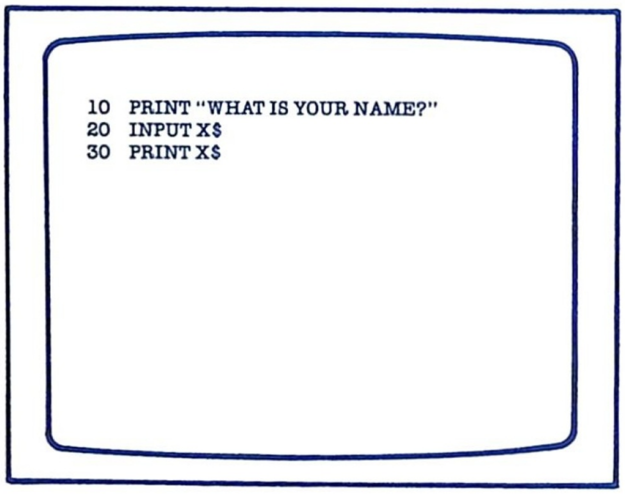
When the variable x equals 7, the program branches to line 240.

IF–THEN—These are also conditioned branching statements. If a certain condition exists in the program, you can tell the computer to perform a specific function. For example:

```
140 IF x = y THEN PRINT "YOU ARE CORRECT"
```

When variable x equals variable y, the program prints YOU ARE CORRECT.

INPUT—This statement halts the program and allows you to input a string or character using the keyboard. You can also use INPUT to print a prompt message. You use quotation marks just as you would in PRINT statements to accomplish this.



```
10  PRINT "WHAT IS YOUR NAME?"
20  INPUT X$
30  PRINT X$
```

An input program.

Try this:

```
10 PRINT "WHAT IS YOUR NAME ?"
20 INPUT x$
30 PRINT x$
```

INVERSE—When this command appears in a program, all of the characters that follow will be printed black-on-white instead of white-on-black.

**LEFT$**—This function will extract a specified number of characters from a string. LEFT$ begins counting from the first character in a string and counts to the right. For example:

```
1ØØ a$ = "MISSISSIPPI"
11Ø PRINT LEFT$(a$,4)
```

The computer will print MISS.

LEN—Stands for length. This function counts the number of numeric characters, letters, or spaces in a string.

LET—LET allows you to assign values to variables in a program. The LET command should look like this:

```
LET x = 1Ø
```

You are telling the computer that x will be equal to 10 until you tell it otherwise. The use of the LET statement is optional. x = 10 and LET x = 10 are interpreted exactly the same.

LIST—This command tells the computer to list the program statements in order. You may also list specific lines of the program by entering the line number after the command.

`LIST`	(This lists the entire program.)
`LIST 400,`	(Line 400 and all subsequent lines will be listed.)
`LIST 100`	(Only line 100 will be listed.)

LOAD—This command takes a program or data from a storage device (the Digital Data Pack) and "loads" it into the computer's memory. The LOAD command does not execute the program, it only loads it into memory.

LOCK—This command protects files that you have stored on a Digital Data Pack.

**MID$**—This command is used to extract part of a string. MID$ allows you to extract a substring from a larger string.

MON—Allows you to watch, on your monitor, data entering and leaving the Digital Data Pack.

NEW—This command clears any program in memory. It acts just like an eraser. Be sure to save and store any programs you want to keep before entering NEW.

NEXT—This statement can only be used with a FOR...TO command. The NEXT statement tells the computer to evaluate the next value in the FOR...TO statement. As long as the value does not exceed the limits of the FOR...TO statement, the computer will act on the value. The NEXT statement controls the loop. If the values are within limits, the loop will continue. Once the values exceed the limits set in the FOR...TO statement, the NEXT statement will tell the computer to leave the loop and continue with the next line of the program.

NOMON—Cancels the MON command.

NORMAL—This command reverses or cancels the INVERSE command.

NOTRACE—Cancels the TRACE command.

ONERR GOTO—When debugging a program, this command can save you a lot of time. When this command is activated, it will override the termination signal produced by an error condition.

ON GOSUB—This branching command will branch to several subroutines when a specified condition exists. This is the way the command looks.

```
120 ON x + 6 GOSUB 200
```

ON GOTO—This statement works just like ON GOSUB except that a GOTO loop is continuous.

OPEN—The OPEN command opens files so that you can work with the data in a file.

PDL—This command activates and reports on the condition of the paddles.

PLOT—This is used for displaying color or low-resolution graphics at a specific column,row coordinate. For example:

```
10 GR
20 COLOR = 4
30 PLOT 10,15
```

Will display a dark green block at Column 10, Row 15.

POKE—This command changes the contents of a memory location. You can store a value of your choosing in a specific location. The POKE statement must look like this:

```
    POKE (memory location), (a value from 0-255)
```

POS—This function will return the current horizontal location of the cursor from left to right.

PRINT—When you want to display information on the screen, the PRINT statement will do it. Words to be printed must appear in quotation marks. When words, numbers, or strings follow the PRINT statement with the variable's name, you don't need to use quotation marks.

PR#—The PR# command controls the printer. To activate the printer, enter PR#1, and then type LIST and the program on the monitor will print out. Use PR#0 to turn the printer off after you have finished printing.

READ—This statement tells the computer to read information from DATA statements in order, from left to right. You must use READ and DATA statements together. A comma must separate each piece of DATA.

REM—The computer does nothing with this statement. It is a notekeeping device for you. REM or Remark statements are given line numbers and identify parts of the program for user reference.

RENAME—This command changes the name of a file. To change the name of a file, simply enter:

```
RENAME THE OLD NAME , THE NEW NAME
```

RESTORE—This statement tells the computer to go back to the beginning of a DATA statement and read the information again.

RESUME—RESUME is used to start the program running after an ONERR routine has been executed.

RETURN—This statement tells the computer to go back and read the line immediately following a GOSUB command. Each subroutine must end with a RETURN statement.

**RIGHT$**—This command works just like MID$ and LEFT$. RIGHT$ will give you a specific number of characters at the right or end of the string.

ROT=—This statement allows you to rotate or turn a user-defined character in the high-res mode.

ROT=0 will display the character as defined.

ROT=16 turns the character 90 degrees clockwise.

ROT=32 turns the character 180 degrees clockwise.

ROT=64 turns the character 360 degrees clockwise.

RUN—This is the computer's ignition key. When you command the computer to RUN, it will begin at the first line of the program and carry out each program line in sequence. If you wish, you can enter a line number following the RUN command and the computer will begin running at that line number.

SAVE—This command is used to load data or a program from the computer RAM to a Digital Data Pack. This command has the same features and works the same way as the LOAD function does.

SCALE=—With the Adam, you can display a graphic larger than life if you like. By specifying a number from 1 to 255, you can make a character grow to 255 times its size.

SCRN—When you plot a lot of different colors, this command will help you keep track of where each color is plotted. The location of the block in question must be specified by a column,row coordinate.

SPEED—The SPEED statement lets the user control the display speed of the computer. The range of speed is from 255 (normal) to 0 (the slowest speed). To change the speed, simply put a line like this in your program.

```
SPEED = 1ØØ
```

STOP—This command is the same as END. It stops the program. To continue the program, you can type in CONT.

STR$—This function will change numbers or numeric variables into a string. This allows you to input numbers and have them displayed in a string print statement.

TAB—This command allows you to display characters at different locations. When used with the PRINT statement, this function acts like the tab key on a typewriter. For example:

```
1Ø PRINT TAB(1Ø) "HELLO"
```

will display the word HELLO, beginning at Column 10.

TEXT—This command will return the normal text screen to your display devices. You must use this command to get out of the low- and high-resolution graphics mode.

TRACE—When this command is activated, the line numbers of the program will display as the program is run. The NOTRACE command deactivates the TRACE command.

UNLOCK—This deactivates the LOCK command so that you can delete a file.

VLIN—This command will draw a vertical line between two or more

```
RENAME THE OLD NAME, THE NEW NAME
```

RESTORE—This statement tells the computer to go back to the beginning of a DATA statement and read the information again.

RESUME—RESUME is used to start the program running after an ONERR routine has been executed.

RETURN—This statement tells the computer to go back and read the line immediately following a GOSUB command. Each subroutine must end with a RETURN statement.

**RIGHT$**—This command works just like MID$ and LEFT$. RIGHT$ will give you a specific number of characters at the right or end of the string.

ROT=—This statement allows you to rotate or turn a user-defined character in the high-res mode.

ROT=0 will display the character as defined.

ROT=16 turns the character 90 degrees clockwise.

ROT=32 turns the character 180 degrees clockwise.

ROT=64 turns the character 360 degrees clockwise.

RUN—This is the computer's ignition key. When you command the computer to RUN, it will begin at the first line of the program and carry out each program line in sequence. If you wish, you can enter a line number following the RUN command and the computer will begin running at that line number.

SAVE—This command is used to load data or a program from the computer RAM to a Digital Data Pack. This command has the same features and works the same way as the LOAD function does.

SCALE=—With the Adam, you can display a graphic larger than life if you like. By specifying a number from 1 to 255, you can make a character grow to 255 times its size.

SCRN—When you plot a lot of different colors, this command will help you keep track of where each color is plotted. The location of the block in question must be specified by a column,row coordinate.

SPEED—The SPEED statement lets the user control the display speed of the computer. The range of speed is from 255 (normal) to 0 (the slowest speed). To change the speed, simply put a line like this in your program.

```
SPEED = 1ØØ
```

STOP—This command is the same as END. It stops the program. To continue the program, you can type in CONT.

STR$—This function will change numbers or numeric variables into a string. This allows you to input numbers and have them displayed in a string print statement.

TAB—This command allows you to display characters at different locations. When used with the PRINT statement, this function acts like the tab key on a typewriter. For example:

```
1Ø PRINT TAB(1Ø) "HELLO"
```

will display the word HELLO, beginning at Column 10.

TEXT—This command will return the normal text screen to your display devices. You must use this command to get out of the low- and high-resolution graphics mode.

TRACE—When this command is activated, the line numbers of the program will display as the program is run. The NOTRACE command deactivates the TRACE command.

UNLOCK—This deactivates the LOCK command so that you can delete a file.

VLIN—This command will draw a vertical line between two or more points. You will only be able to use this command in the low-resolution mode. The formula for VLIN looks like this:

```
VLIN  Beginning Row #,  Ending Row #  AT  Column #
```

VTAB—Allows you to place text on the screen at a precise location. VTAB works just like TAB, except that you indicate row numbers with VTAB. For example:

```
VTAB(10)
```

will display the text following at Row 10.

WRITE—This must be preceded with a PRINT statement and a CONTROL-D. This command will cause all subsequent PRINT values to be sent to a file.

XDRAW—This is a high-resolution graphics statement that displays a character in its complementary color. This can be used to erase a figure and simulate animation. This statement can only be used in the high-resolution graphics mode.

FUNCTIONS

Many of the functions will work only if you give the Adam a number to work with. The functions that require such a number are followed by the notation (X). The X is the *argument*.

ABS(X)—This function tells the computer to give you the absolute value of an expression. An Expression is sometimes referred to as the Argument. The ABS command will always give you a positive number even if the argument is negative. This command can be used to find out the absolute value of any complicated series of mathematical computations. For example, input:

```
10 PRINT ABS(47)
```

(The absolute value is 47, but we know that.)

Now try

```
    20 PRINT ABS (21*6.1)-20
```

(Now do you see why this command is useful?)

ASC(X)—This function will return the ASCII code value for any character.

```
    PRINT ASC("D")
```

(Will return the ASCII code for an uppercase D, or 68.)

ATN(X)—This function is rather mathematical and we won't use it in this book. This function will give you the arctangent of any number that appears in parentheses after the ATN command.

**CHR$(X)**—This function is the exact opposite of the ASC function. The CHR$ function will change an ASCII code number into the appropriate character.

```
5 a$ = CHR$(68)
1Ø PRINT a$
```

This will print the letter D.

COS(X)—This function will compute the cosign of any angle that you put into parentheses following the function. For example, enter:

```
PRINT COS(1Ø)
```

This will print the number −.839071532.

EXP(X)—This returns the exponential function or the opposite of the LOG function. This function raises the number 2.718281828 to the power that you input in parentheses.

```
PRINT EXP(1Ø)
```

This will raise 2.718281828 to the tenth power.

INT(X)—This function is used when you want a whole number as an output rather than a fraction or a decimal. The INT function returns the largest number possible that does not exceed the number in parentheses (ARGUMENT). For negative numbers, the next integer value will be returned. Try this:

```
5 a = INT(99.887)
10 PRINT a
```

The Adam will print 99. Try the same program only let **a = INT(−99.887).**

LEN(X)—This function will return the number of characters, including spaces in a string. Try this:

```
5 a$ = "THE ADAM IS A NEAT MACHINE"
10 PRINT LEN(a$)
```

The Adam will print 26. Note that all the spaces count as a character.

LOG(X)—This function will give you the natural logarithms of any number larger than 0. To figure the log of a number, simply input:

```
    PRINT  LOG(any number)
```

PEEK—This command allows us to look into a memory location and see what is stored there. You must include the numbers of the memory location you want to examine before PEEK will work. For example:

```
    PEEK(212)
```

will allow you to see what is stored in memory location 212.

RND—The random function tells the Adam to generate a real number that is less than 1 and greater than 0.

SIN(X)—The SIN function will return the sine of an angle. The angle should be the argument. For example:

```
PRINT  SIN(10)
```

will return the sine of a 10° angle.

SQR(X)—This handy function will give you the square root of any number you choose. Use the following program to extract the square root.

```
PRINT  SQR(any number)
```

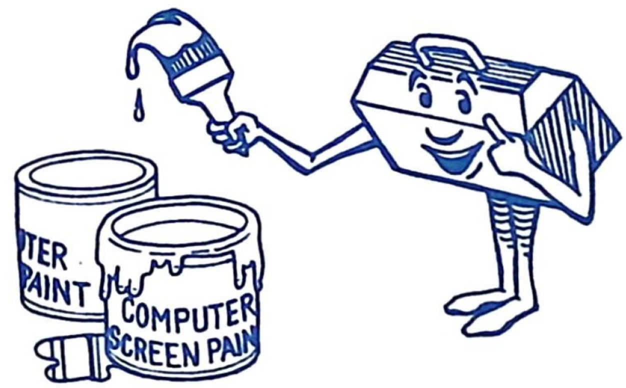
TAN(X)—This function will return the tangent of the angle number in the parentheses.

USR(X)—This function is a machine-language function. It initiates a subroutine with a starting point of X. A USR function must be stored in memory using POKE statements.

VAL(X)—This function will give you the numeric value for a string. This function ignores letters in a string and assigns numeric values of number characters only.

Use this chapter as a reference. It will be useful as you begin to develop your own programs.

Now that we know how the Adam works and how to "speak" Smart-BASIC, let's begin doing some real programming.



Color Subroutines 3

Now that you have learned how the Adam works and know the BASIC commands and statements, it is time to begin working with some color subroutines. The Adam is a very colorful computer, and with a little practice, you can dazzle your friends with color computing ability.

COLOR AND THE ADAM COMPUTER

The Adam is capable of displaying sixteen different colors. Each color has a number. The following is a list of all the colors and their number codes:

Code	Color	Code	Color
0	BLACK	8	DARK YELLOW
1	MAGENTA	9	MEDIUM RED
2	DARK BLUE	10	GREY
3	DARK RED	11	LIGHT RED
4	DARK GREEN	12	LIGHT GREEN
5	GREY	13	LIGHT YELLOW
6	MEDIUM GREEN	14	CYAN (AQUA)
7	LIGHT BLUE	15	WHITE

The color codes just listed are for low-resolution colors. The high-resolution mode has its own color codes; we will list them later. But, first, let's look at Adam's low-resolution color capability.

LOW-RESOLUTION COLOR

Before we do any color programming, we must learn about the commands that control the color capability of the Adam. Let's look at these commands one at a time.

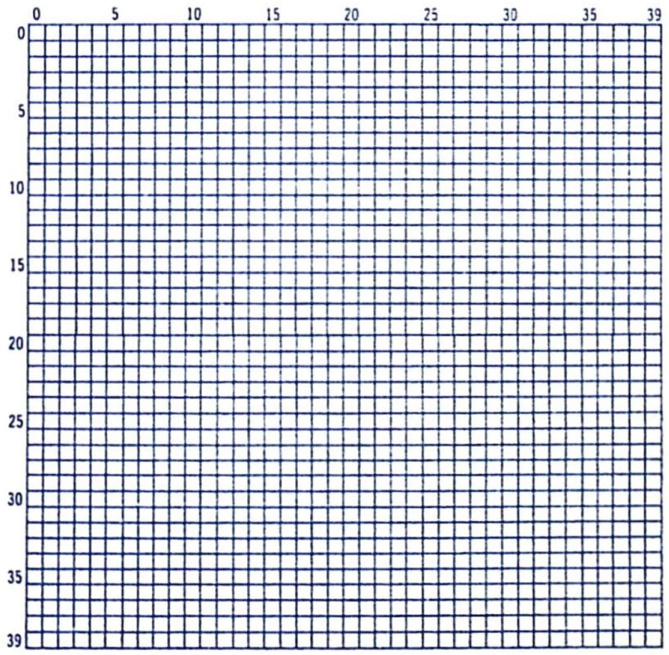
GR

The command, GR, tells the Adam that you wish to enter the low-resolution mode. You cannot create color in the text mode. All of our programs that require low-resolution mode color will begin with the GR command.

Screen Locations for the Low-Resolution Mode

The low-resolution screen looks just like the text mode screen to us. However, the Adam computer sees the screen in a much different way. You will need to become familiar with the screen locations before you will be able to generate color.

The low-res screen display is 40 columns across and 40 rows down. There is also space for four rows of text at the bottom of the screen. Alphanumeric characters and punctuation marks can only be printed in these last four rows. They will not display anywhere else on the screen. It will help if you can get some graph paper and number the boxes. Number from 0 to 39



Low-res screen location grid.

across the top, and from 0 to 39 down the side. Make your graph paper look like the drawing on page 39.

We will place our color on the screen using row and column coordinates. For example Row 0, Column 0 is in the upper left-hand corner of the screen. Row 39, Column 39 is in the lower right-hand corner of the screen. It is important to remember that the rows and columns are numbered from 0 to 39, not 1 to 40. Now that we know how the Adam sees the screen, let's display some color on the screen.

The HLIN Command

HLIN is a signal for the Adam computer to draw a horizontal line. This command must be followed by a series of numbers. The formula for the HLIN command looks like this.

> HLIN Beginning column location, Ending column location AT Row location.

For example:

```
HLIN 1Ø,2Ø AT 15
```

will draw a horizontal line from Column 10 to Column 20 at Row 15. Let's try these coordinates in our first color subroutine. Enter this:

```
1ØØ REM  First Color
11Ø GR
12Ø COLOR = 1
13Ø HLIN 1Ø,2Ø AT 15
```

Type RUN and press ENTER and see what happens. You should see a magenta line displayed at Row 15 that begins at Column 10 and ends at Column 20. Now, let's look at each line of the program.

—Line 100 names the program.
—Line 110 sends the Adam computer into low-resolution graphics.
—Line 120 names the color to be used as Magenta.
—Line 130 displays a magenta horizontal line.

COLOR=

In the last program, we used COLOR= to tell the Adam that we wanted to display in magenta. COLOR= tells the computer which of the sixteen colors we want to use for our display. This command does not display the color, it only identifies which color the Adam is to use. To display color, we must use one of the display commands. We have discussed HLIN already. Let's look at the other two.

VLIN

This command will display vertical lines. We still use row and column coordinates to display the color, but the coordinates are entered differently. Here is the formula for the VLIN command:

VLIN Beginning Row, Ending Row AT Column

Do you see the difference?

Let's try another color program using VLIN.

```
100 REM  Color 2
110 GR
120 COLOR = 2
130 VLIN 10,20 AT 15
```

Now type RUN and press RETURN.

As you can see, the Adam computer now displays a dark blue line at Column 15, beginning at Row 10 and ending at Row 20. Here is how the program works.

- —Line 100 names the program.
- —Line 110 sends Adam into the low-resolution mode.
- —Line 120 tells Adam to display in dark blue.
- —Line 130 displays a vertical dark blue line from Row 10 to Row 20 at Column 15.

Now let's look at the last display command.

PLOT

PLOT can be used for coloring specific blocks or screen locations. Again, we use row and column coordinates to place our color. The formula for PLOT looks like this:

PLOT Column Number, Row Number

Here is a sample program for you to try.

```
100 REM  Plot
110 GR
120 COLOR = 4
130 PLOT 20,20
```

When you type RUN and press ENTER, you should see a green square right in the middle of the screen.

Before we go on, here are some important things to remember about color in the low-resolution mode.

1. The GR command must always be present in the program before any color can be displayed.
2. The low-resolution screen is 40 columns across and 40 rows down.
3. COLOR= only defines the color, it cannot display color.
4. HLIN displays horizontal lines.
5. VLIN displays vertical lines.
6. PLOT displays a solid block or screen location.
7. After running a low-res program, you must enter the text command to return the screen to the text mode.

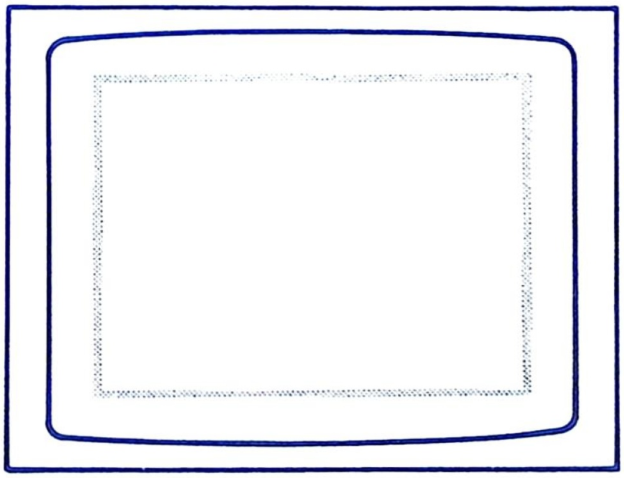
VARIABLES

In Chapter 1, we mentioned that you can assign values to variables. All of the display commands discussed so far can be used with variables. This allows a great deal of versatility. Let's do some programming and you will see what we mean.

Border

Our first color subroutine displays an attractive border. Type this:

```
100 REM  Border
110 GR
120 COLOR = 5
130 HLIN 0,39 AT 0
140 VLIN 0,39 AT 39
150 HLIN 0,39 AT 39
160 VLIN 0,39 AT 0
```



Border screen dump.

- —Line 100 names the program.
- —Line 110 activates the low-resolution screen.
- —Line 120 sets the display color at 5 (Grey).
- —Line 130 displays a horizontal line from Column 0 to Column 39 at Row 0.
- —Line 140 displays a vertical line from Row 0 to Row 39 at Column 39.

—Line 150 displays a horizontal line from Column 0 to Column 39 at Row 39.
—Line 160 displays a vertical line from Row 0 to Row 39 at Column 0.

This program displays a grey screen border. However, we can use a variable or two and add a little excitement to our border.

Border Flash

When you enter and run the following program, Adam will display a flashing multicolored border. Notice that we have used a variable in line 130, and a FOR/NEXT loop in lines 120 and 180.

```
100 REM  Border Flash
110 GR
120 FOR c = 1 TO 15
130 COLOR = c
140 HLIN 0,39 AT 0
150 VLIN 0,39 AT 39
160 HLIN 0,39 AT 39
170 VLIN 0,39 AT 0
180 NEXT c
190 GOTO 120
```

Let's look at the program.

—Line 100 names the program.
—Line 110 activates the low-resolution mode.
—Line 120 defines the variable c as every number from 1 to 15. These are the code numbers for every color except Black.
—Line 130 activates color c or each color in turn.
—Lines 140 to 170 display the border.
—Line 180 goes back and gets the next value of c.
—Line 190 loops to line 120 so that the border will continue to flash.

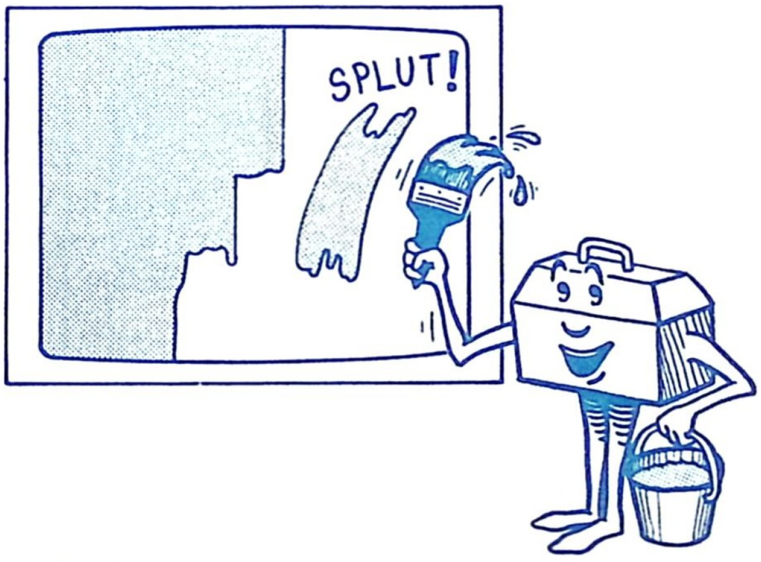
This program uses a very useful looping tool, the FOR/NEXT loop.

FOR/NEXT COMMANDS

When these commands are used together, a very useful loop is created. The FOR portion of the command sets the limits for the loop. In *Border Flash*, the limits are 1 to 15. In line 120, we tell the computer that the color (or C) will equal 1 to 15. The NEXT command forms the other end of the loop. The NEXT command tells the computer to get the next value of **c**, as long as **c** is within the limits set in the FOR statement. We will use this command a great deal in the rest of the book.

SCREEN COLOR

Next, we have a program that will color the entire screen.



ToolKit Man painting the screen.

```
100 REM  Screen Color
110 GR
120 FOR x = 0 TO 39
130 COLOR = 3
140 HLIN 0,39 AT x
150 NEXT x
```

—Line 100 names the program.
—Line 110 activates the low-resolution mode.
—Line 120 states the value of x as Rows 0 to 39.
—Line 130 defines the color as dark red.

—Line 140 displays the lines on the screen from Columns 0 to 30 at Row X.
—Line 150 gets the next value of x.

Notice that we have again used a FOR/NEXT loop in this program.

You can color the screen any color you like. If you make a few changes to the program, you can input a color directly into the program.

```
100 REM  Input Color
110 GR
115 INPUT c
120 FOR x = 0 TO 39
130 COLOR = c
140 HLIN 0,39 AT x
150 NEXT x
160 GOTO 115
```

In this program, line 115 stops the program and waits for you to input a color code number. Once you choose the number and press RETURN, the screen is colored. Line 160 loops the program back to line 115 so that you can pick another number.

Random Color

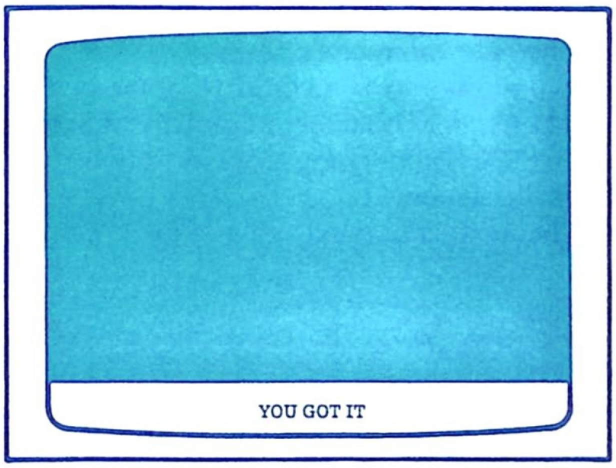
The "INPUT COLOR" program allows you to pick a color. Now let's let the Adam computer pick a color.

```
100 REM  Random Color
110 GR
120 n = RND(-n)
130 c = 1 + INT(15 * RND(1))
140 COLOR = c
150 FOR x = 0 TO 39
160 HLIN 0,39 AT x
170 NEXT x
180 GOTO 120
```

—Line 100 names the program.
—Line 110 activates the low-resolution mode.
—Lines 120 and 130 tell the Adam computer to pick a number at random.
—Line 140 defines c as a random number between 1 and 15.

—Lines 150 to 170 color and display the colors on the screen.
—Line 180 loops back to line 120 so that the Adam computer can select another number.

We will use the random-number generator quite a bit in later chapters. Remember that lines 120 and 130 generate the random number and line 140 tells the computer to be sure that the number is between 0 and 15.



REWARD dump.

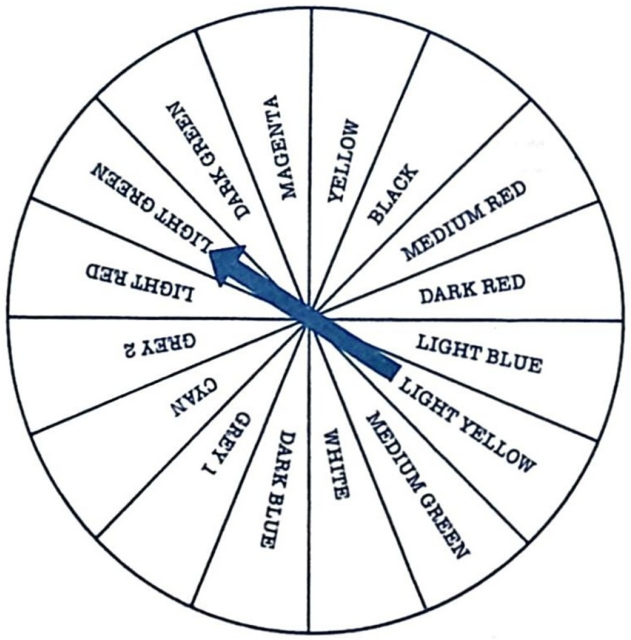
Reward

Now let's look at a useful Reward subroutine.

```
100 REM  Reward
110 GR
120 FOR c = 1 TO 15
130 COLOR = c
140 FOR x = 0 TO 39
150 HLIN 0,39 AT x
160 NEXT x
170 PRINT "       YOU GOT IT"
180 FOR i = 1 TO 100: NEXT I
190 HOME
200 NEXT c
```

This program combines color and some text. You can use this subroutine to produce a reward for a correct answer in a game or a quiz. Here is how it works.

—Line 100 names the program.
—Line 110 activates the low-resolution mode.
—Line 120 states the value of c as 1 to 15.
—Line 130 sets the color.
—Line 140 states the value of x, or row, as 0 to 39.
—Line 150 displays the colors.
—Line 160 gets the next value of x.
—Line 170 prints the prompt, YOU GOT IT, at the bottom of the screen.
—Line 180 is a little timer to hold the prompt on the screen. The computer counts from 1 to 100 before going to the next line.
—Line 190 clears the screen.
—Line 200 gets the next color (next value of c).



Color wheel.

Color Choice

Here is a fun program that allows you to choose a color.

```
100 REM  Color Choice
110 HOME
120 PRINT "PRESS THE FOLLOWING KEYS AND I WILL DO
    MY  STUFF"
125 PRINT
130 PRINT "RED","1"
135 PRINT
140 PRINT "BLUE","2"
145 PRINT
150 PRINT "GREEN","3"
155 PRINT
160 PRINT "YELLOW","4"
165 FOR i = 1 TO 250: NEXT i
170 INPUT c
180 IF c = 1 GOTO 1000
190 IF c = 2 GOTO 2000
200 IF c = 3 GOTO 3000
210 IF c = 4 GOTO 4000
220 GOTO 170
1000 GR
1010 COLOR = 3
1020 FOR x = 0 TO 39
1030 HLIN 0,39 AT x
1035 NEXT x
1040 PRINT "THIS IS A RED SCREEN"
1050 FOR i = 1 TO 300: NEXT i
1060 HOME
1070 GOTO 170
2000 GR
2010 COLOR = 2
2020 FOR x = 0 TO 39
2030 HLIN 0,39 AT x
2035 NEXT x
2040 PRINT "THIS IS A BLUE SCREEN"
2050 FOR i = 1 TO 300: NEXT i
2060 HOME
2070 GOTO 170
3000 GR
3010 COLOR = 4
3020 FOR x = 0 TO 39
3030 HLIN 0,39 AT x
3035 NEXT x
3040 PRINT "THIS IS A GREEN SCREEN"
3050 FOR i = 1 TO 300: NEXT i
3060 HOME
3070 GOTO 170
```

```
4000 GR
4010 COLOR = 13
4020 FOR x = 0 TO 39
4030 HLIN 0,39 AT x
4035 NEXT x
4040 PRINT "THIS IS A YELLOW SCREEN"
4050 FOR i = 1 TO 300: NEXT i
4060 HOME
4070 GOTO 170
```

—Line 100 names the program.
—Line 110 clears the screen.
—Lines 120 to 160 print the words in quotation marks onto the screen.
—Line 170 holds the program until you input a number.
—Lines 180 to 210 branch the program according to the numbers that are input.
—Lines 1000 to 1060 color, display, and print the red screen condition.
—Line 1070 loops back to line 170, so you can select another number.
—Lines 2000 to 2060 color, display, and print the blue screen condition.
—Line 2070 loops back to line 170.
—Lines 3000 to 3060 color, display, and print the green screen conditions.
—Line 3070 loops back to line 170.
—Line 4000 to 4060 color, display, and print the yellow screen condition.
—Line 4070 loops back to line 170.

When you run this program, you will see the beginning screen prompts. Once you input a number from 1 to 4 and the program branches, the screen goes into the low-resolution mode (lines 1000, 2000, 3000, and 4000). You can watch as the screen is colored and the identifying prompt appears at the bottom of the screen.

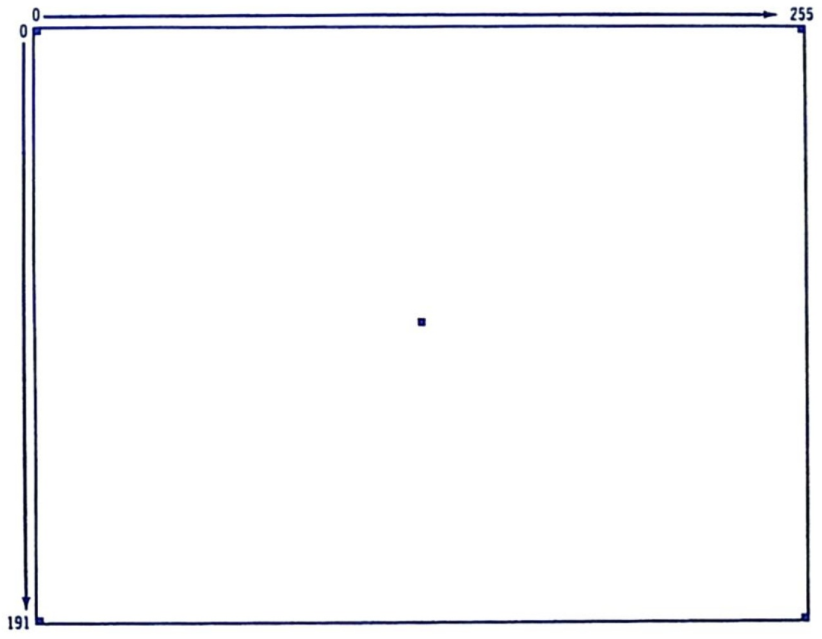
You can add as many colors as you like to the *Color Choice* program. You can also plot a border of a different color if you like. Experiment and have some fun. When you are ready, we can move on to high-resolution color.

HIGH-RESOLUTION COLOR

Using high-resolution color is no more difficult than using low-resolution color. In fact, once you know how to use the high-res commands, you will be off and running. Before we talk about these commands, however, let's take a look at the high-resolution screen.

High-Resolution Screen Locations

The high-resolution screen has 256 columns and 159 rows in the mixed text and graphics mode, and 256 columns and 192 rows in the full screen graphics mode. Since there are more columns and rows in the high-resolution mode, we can place color with greater precision and greater density.



High-res screen location grid.

Column 0 is in the top left corner of the screen, and Row 0 is at the top of the screen. Columns are numbered from 0 to 255. Rows are numbered from 0 to 159 (mixed screen) or 0 to 191 (full graphics). Use our illustration to help you place your color. Graph paper will also be helpful. Now, let's look at the high-resolution commands.

HGR

The HGR command sends the Adam computer into the high-resolution text and graphics mode. This command activates the 256-column by 159-row graphics screen with 4 rows of text screen at the bottom. This command must appear in the program before any color can be displayed on the high-res screen.

HGR2

HGR2 activates the full graphics high-res screen. When this command is used, you will get a screen that is 256 columns across and 192 rows down. There is no text space at the bottom of this screen.

HCOLOR=

The high-resolution mode has its very own set of colors. To activate these colors, you will use the HCOLOR= command. This command works just like COLOR= except that you must use the high-res color codes in the following chart.

HIGH-RESOLUTION COLOR CODES

Code	Color	Code	Color
0	BLACK	8	DARK YELLOW (ORANGE)
1	GREEN	9	DARK BLUE
2	DARK RED	10	GREY
3	WHITE	11	LIGHT RED
4	BLACK	12	DARK GREEN
5	MEDIUM RED	13	LIGHT YELLOW
6	MEDIUM BLUE	14	CYAN (AQUA)
7	WHITE	15	MAGENTA (PURPLE)

To select a color, simply add one of the color codes to the HCOLOR= command, like this:

```
HCOLOR=13
```

HPLOT

To draw lines or points in the high-resolution mode, you must use the HPLOT command. As you will see, this command is very versatile.

To plot one point, simply add a column,row coordinate like this:

```
HPLOT 2ØØ,8Ø
```

This will plot a point at Column 200, Row 80.

You can also plot a line using HPLOT. All you need to do is tell Adam where to begin the line and where to end it. For example:

```
HPLOT 1Ø,5Ø TO 1Ø,1ØØ
```

will plot a line from Column 10, Row 50 to Column 10, Row 100.

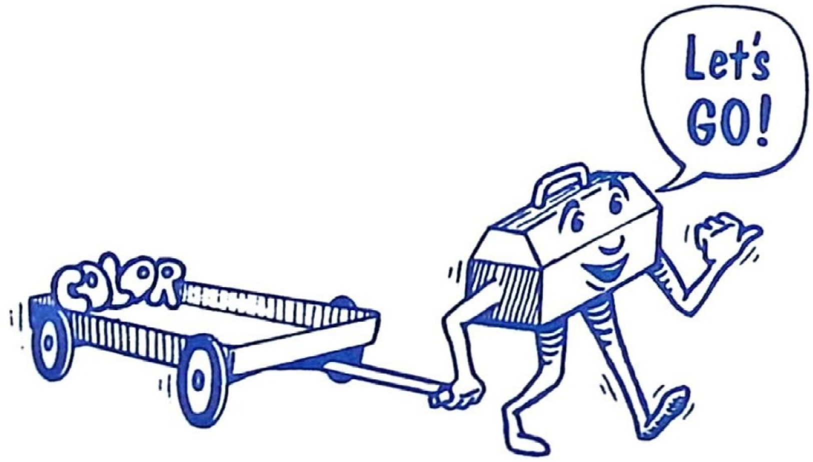
Let's put all of these commands together in a program that demonstrates the Adam's high-resolution color capabilities.

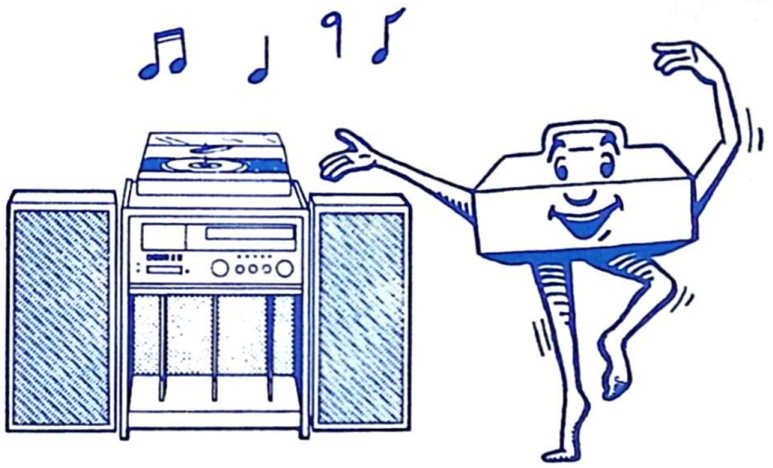
High-Resolution Demo

```
1ØØ REM  High Res
11Ø HGR
12Ø HCOLOR = 1
13Ø FOR y = 1Ø8 TO 148: FOR x = 68 TO 71
14Ø HPLOT y,x: NEXT x
15Ø HCOLOR = 2
16Ø FOR x = 72 TO 75
17Ø HPLOT y,x: NEXT x
18Ø HCOLOR = 6
19Ø FOR x = 76 TO 79
2ØØ HPLOT y,x: NEXT x
21Ø HCOLOR = 5
22Ø FOR x = 8Ø TO 83
23Ø HPLOT y,x: NEXT x
24Ø HCOLOR = 14
25Ø FOR x = 84 TO 87
26Ø HPLOT y,x: NEXT x
27Ø HCOLOR = 1Ø
28Ø FOR x = 88 TO 91
29Ø HPLOT y,x: NEXT x
3ØØ NEXT y
31Ø PRINT "THIS IS HIGH RES COLOR"
```

—Line 100 names the program.
—Line 110 activates the high-resolution mode.
—Line 120 sets the color at 1.
—Line 130 defines Y as Column 108 to 148 and X as Row 68 to 71.
—Line 140 plots a bar from Column 108 to 148 and from Row 68 to 71; it gets the next value of x.
—Line 150 sets the color at dark red.
—Line 160 sets the x, or row, coordinates from Row 72 to Row 75.
—Line 170 plots the dark red bar.
—Line 180 sets the color at medium blue.
—Line 190 sets the x, or row, coordinates from Row 76 to Row 79.
—Line 200 plots the medium blue bar.
—Line 210 sets the color at medium red.
—Line 220 sets the x, or row, coordinates from Row 80 to Row 83.
—Line 230 plots the medium red bar.
—Line 240 sets the color at cyan.
—Line 250 sets the x, or row, coordinates from Row 84 to Row 87.
—Line 260 plots the cyan bar.
—Line 270 sets the color at grey.
—Line 280 sets the x, or row, coordinates from Row 88 to Row 91.
—Line 290 plots the grey bar.
—Line 300 gets the next value of y.
—Line 310 prints the prompt in quotation marks.

We will work with the high-res mode more in the graphics chapter.
Now let's look at how to make the Adam computer beep.



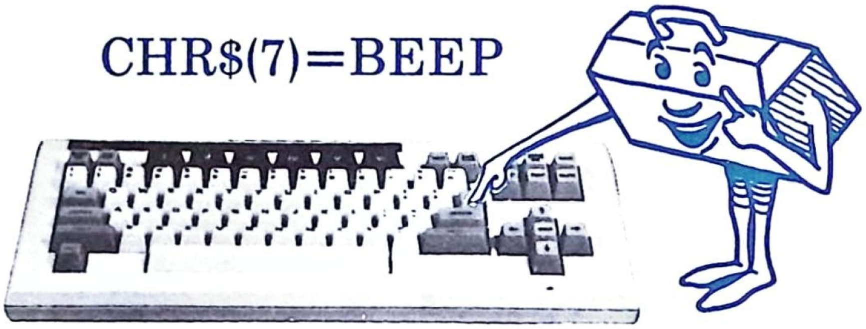


Sound Subroutines 4

The Adam computer has a separate sound chip on board. But this chip can only be accessed by machine-language programs and that kind of programming is well beyond the scope of this book. But do not despair. We can still make the Adam beep at us.

By experimenting with FOR/NEXT loops and the SPEED control of the Adam, we can create some interesting sounds. Let's make some noise.

TURNING ON THE BELL



When you use the Adam computer as a typewriter, you will hear a bell or beep sound every time you press a key. To turn on the bell in a program, you must access the ASCII number for the bell.

The code number for the bell is 7. Type in the following:

```
10 PRINT CHR$(7)
```

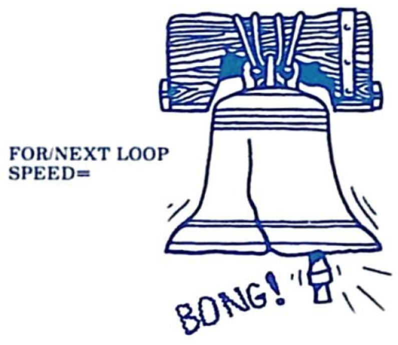
Now, press RETURN. Hear the bell?

The CHR$ function matches an ASCII code with its character; in this case, the bell. Since the computer cannot print a sound, it plays a beeping noise. You can enter the line you just typed anywhere that you want the Adam to beep at you. When the computer reaches the line, it will sound the bell.

If you add a FOR/NEXT loop, you can make some really interesting sounds.

CHR$(7) AND FOR/NEXT LOOPS

By using FOR/NEXT loops, you can make the bell sound repeatedly.



Try this:

```
100 FOR i = 1 TO 20
110 PRINT CHR$(7)
120 NEXT i
```

—Line 100 defines i as the numbers 1 through 20.
—Line 110 activates the bell.
—Line 120 gets the next value of i.

This little program will activate and sound the bell twenty times. By changing line 100, you can control the number of times the bell is activated. Try changing the numbers in the FOR statement.

The Adam computer also provides us with a way to control the speed of the beeps.

SPEED

The SPEED command has many uses, but it is great for controlling the timing of the beeps.

The SPEED= command must be followed by a number from 0 to 255. The 255 is Adam's normal execution speed. But, by entering a lower number, we can slow down the speed at which the computer works. Note, however, that after you run a program with a slower specified speed, you will need to type in SPEED=255 to bring the computer back up to its normal operating speed. You can do this in the immediate mode before you EDIT or RUN programs again.

Let's look at some programs that use the SPEED= command.

```
100 FOR s = 0 TO 255 STEP 5
105 SPEED = s
110 PRINT CHR$(7)
120 NEXT s
```

—Line 100 defines s, or the speed, as 0 to 255. The step portion of the line tells the computer to cycle in increments of 5.
—Line 105 controls the speed in terms of the variable s.
—Line 110 activates the bell.
—Line 120 gets the next value of s.

When you run this program, you will hear the bell sounding very slowly at first, but as you listen, the beeps get faster and faster until they almost run together.

You can change line 100 and the effect will be reversed. Give this a try:

```
100 FOR s = 255 TO 0 STEP -5
105 SPEED = s
110 PRINT CHR$(7)
120 NEXT s
```

This next program allows you to vary the speed of the beeps another way. Input the following:

```
100 FOR i = 1 TO 25
105 IF i = 5 THEN SPEED = 200
110 IF i = 10 THEN SPEED = 100
120 IF i = 20 THEN SPEED = 50
130 PRINT CHR$(7)
140 NEXT i
```

—Line 100 sets the timing clock; the bell will beep 25 times.
—Lines 105 to 120 change the speed of the beeps according to the number of beeps sounded. On the fifth beep, the speed goes to 200. On the tenth beep, the speed goes to 100. Then, on the twentieth beep, the sound goes to 50.
—Line 140 gets the next value of i.

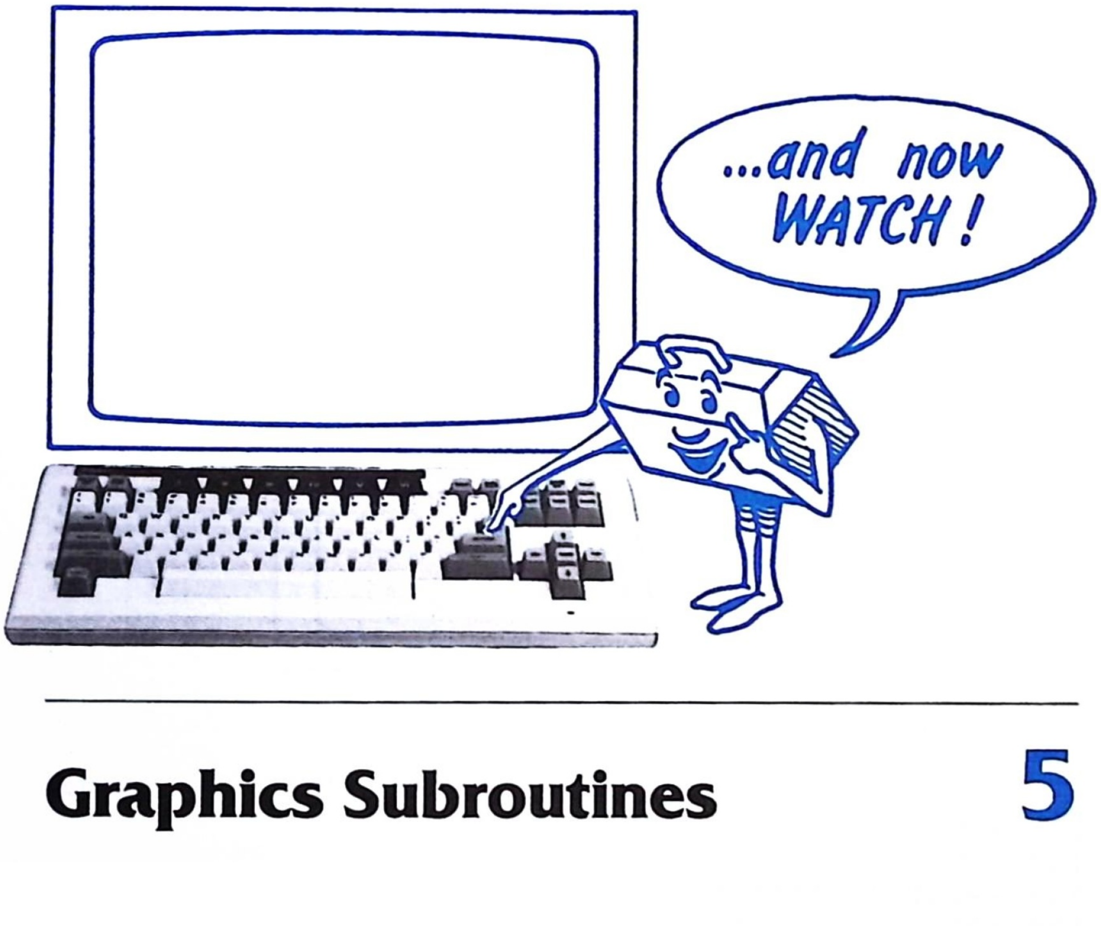
Now we will let you select your own speed. This program will create some neat sounds and it will also give you a good feel for the different execution speeds.

```
90 HOME
100 INPUT "INPUT THE SPEED OF YOUR CHOICE."; x
110 SPEED = x
120 FOR i= 1 TO 20
130 PRINT CHR$(7)
140 NEXT i
150 SPEED = 255
160 GOTO 90
```

—Line 90 clears the screen.
—Line 100 asks you to input the speed of your choice and names your desired speed as x.
—Line 110 sets the speed according to your input in terms of the variable x.
—Line 120 sets the loop. The Adam computer will beep the bell 20 times.
—Line 130 activates the bell.
—Line 140 gets the next value of i.
—Line 150 resets the speed at 255.
—Line 160 loops back to line 90 so that you can select another speed.

We encourage you to experiment with the programs in this chapter. By varying the number of beeps and the speed at which they are played, you can create your own sound effects for games and quizzes.

Now, if you are ready, we will give graphics a try.



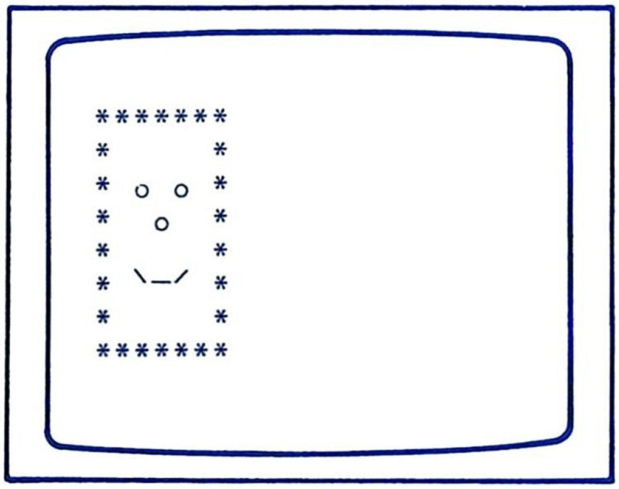
Graphics Subroutines 5

One of the reasons that you probably bought your Adam was so that you could create computer graphics. Right? Well, in this chapter, we will show you how to create graphics in the text mode, the low-resolution mode, and the high-resolution mode. Also, in this chapter, we will create some interesting borders, graphs, and playing boards for games as well as some really neat computer pictures.

Let's begin with the easiest way to create graphics—using the PRINT command.

CREATING GRAPHICS WITH THE PRINT COMMAND

We normally think of the alphanumeric keys and the punctuation keys as a means for displaying text. But with some imagination, they can also be used for displaying graphics. As you will soon see, we can create some interesting graphic displays using letters, numbers, and punctuation marks.



```
*******
*     *
* o o *
*  o  *
*     *
* \_/ *
*     *
*******
```

Input the following program:

```
1ØØ REM  Smiles
1Ø5 HOME
11Ø PRINT "*******"
12Ø PRINT "*     *"
13Ø PRINT "* o o *"
14Ø PRINT "*  o  *"
15Ø PRINT "*     *"
```

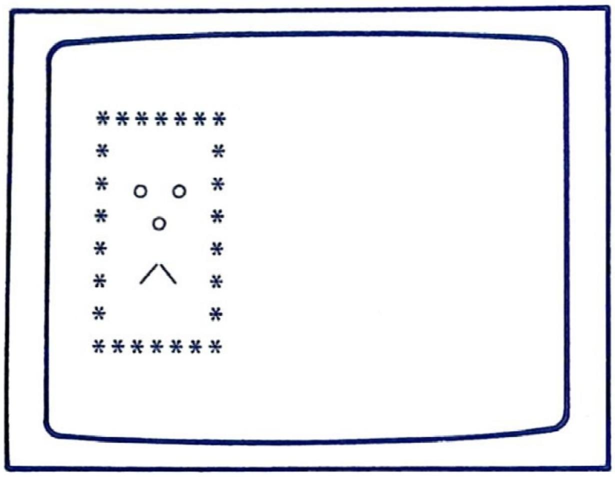
```
160 PRINT "* \_/ *"
170 PRINT "*     *"
180 PRINT "*******"
```

—Line 100 names the program.
—Line 105 clears the screen and begins the display in the upper left-hand corner of the screen.
—Line 110 prints seven asterisks.
—Line 120 prints one asterisk, five spaces, and another asterisk.
—Line 130 prints one asterisk, one space, a lowercase letter 0, a space, another lowercase 0, a space, and another asterisk.
—Line 140 prints an asterisk, two spaces, a lowercase 0, two more spaces, and another asterisk.
—Line 150 prints an asterisk, five spaces, and another asterisk.
—Line 160 prints an asterisk, a space, a backslash, an underscore (a shifted #6), a virgule (a right slash), a space, and another asterisk.
—Line 170 prints an asterisk, five spaces, and another asterisk.
—Line 180 prints seven asterisks.

This program will display our smiling face in the upper left-hand corner of the screen. This little guy is useful as an indicator of a correct answer or a good score on a game or quiz. By changing line 160, you can change the smile into a frown. Try this:

```
100 REM  Frown
105 HOME
110 PRINT "*******"
120 PRINT "*     *"
130 PRINT "* o o *"
140 PRINT "*  o  *"
150 PRINT "*     *"
160 PRINT "*  ^  *"
170 PRINT "*     *"
180 PRINT "*******"
```

All of the lines are the same as in the *SMILE* program except that line 160 now displays the exponent sign instead of the slashes.



```
*******
*     *
* o o *
*  o  *
*     *
*  /\ *
*     *
*******
```

When you run this program, you will notice that it still displays in the upper left-hand corner of the screen. There is a way that we can display our little faces in other places on the screen. Let's take a look at the text mode screen and, then, two useful placement commands.

SCREEN LOCATIONS

In Chapter 3, we discussed the screen locations in the low- and high-resolution modes. But the text mode screen is a little different. The text mode screen has 768 blocks or screen locations. Just as in the low-res and high-res modes, the screen locations are identified by row numbers and column numbers. The Adam text mode screen is made up of 24 rows and 32 columns.

Rows

The row designation describes the up-and-down location of the blocks. Row 0 is at the top of the screen and Row 23 is at the bottom of the screen. The row number thus gives the Adam computer the vertical screen location.

Columns

The column designation describes the side-to-side location of the blocks. Column 0 is at the far left of the screen and Column 31 is at the far right of the screen. The column number gives the Adam computer the horizontal screen location. Using the row and column numbers, you can plot or identify any screen block location on the text screen. Find the screen location for the block at Row 0, Column 0. If you guessed the first block in the upper left-hand corner of the screen, you guessed right.

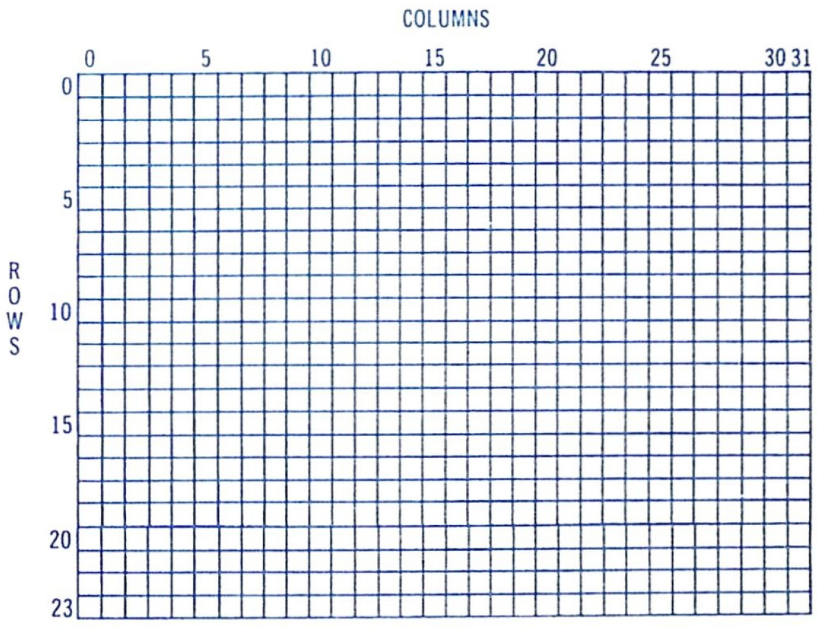


Fig. 5-1. Text mode screen locations.

Now that we understand the organization of the text mode screen, we should discuss the two placement commands.

VTAB and HTAB

If you have done very much typing, you know that the tab key can be used to easily type columns. The TAB function on the Adam computer can be set so that characters can be displayed at any row or column location that you desire. The HTAB command will control the horizontal location of a display. To use HTAB, simply type in the command and follow it with the column number where you want the display to begin. For example:

```
HTAB 15
```

tells the Adam computer to display the characters that follow, beginning at Column 15.

VTAB works in a similar way. The VTAB command will control the vertical location of the display. VTAB must be followed by the desired row number for the display. For example:

```
VTAB 12
```

will display the characters that follow starting at Row 12.

Let's use these commands and practice placing some graphic displays. Add these HTAB and VTAB commands to our *SMILE* program.

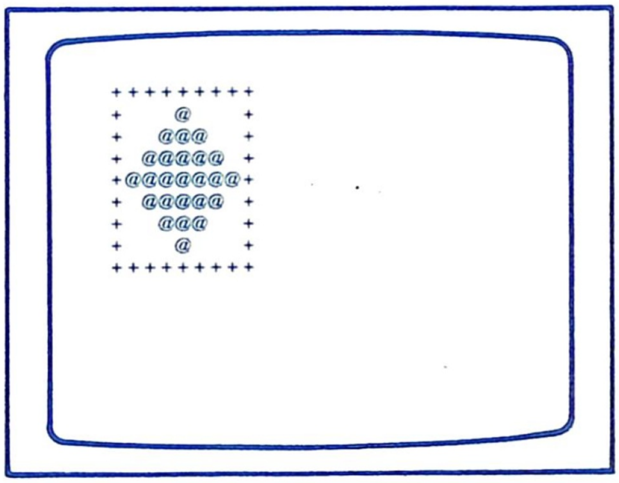
```
100 REM  Htab Vtab
110 HOME
120 HTAB 13: VTAB 9
130 PRINT "*******"
140 HTAB 13: VTAB 10
150 PRINT "*     *"
160 HTAB 13: VTAB 11
170 PRINT "* o o *"
180 HTAB 13: VTAB 12
190 PRINT "*  o  *"
200 HTAB 13: VTAB 13
210 PRINT "*     *"
220 HTAB 13: VTAB 14
230 PRINT "* \_/ *"
240 HTAB 13: VTAB 15
250 PRINT "*     *"
260 HTAB 13: VTAB 16
270 PRINT "*******"
```

—Line 100 names the program.
—Line 110 clears the screen.
—Line 120 sets the coordinates at Column 13, Row 9.
—Line 130 is the same as line 110 in the *SMILE* program.
—Line 140 sets the coordinates at Column 13, Row 10.
—Line 150 is the same as line 120 in the *SMILE* program.
—Line 160 sets the coordinates at Column 13, Row 11.
—Line 170 is the same as line 130 in the *SMILE* program.
—Line 180 sets the coordinates at Column 13, Row 12.
—Line 190 is the same as line 140 in the *SMILE* program.
—Line 200 sets the coordinates at Column 13, Row 13.

—Line 210 is the same as line 150 in the *SMILE* program.
—Line 220 sets the coordinates at Column 13, Row 14.
—Line 230 is the same as line 160 in the *SMILE* program.
—Line 240 sets the coordinates at Column 13, Row 15.
—Line 250 is the same as line 170 in the *SMILE* program.
—Line 260 sets the coordinates at Column 13, Row 16.
—Line 270 is the same as line 180 in the *SMILE* program.

When you run this program you will see our smiling friend in the middle of the screen. Note that the HTAB and VTAB commands only affect the display immediately following the commands.

Next, let's do another graphic using the PRINT statement.



```
+++++++++
+   @   +
+  @@@  +
+ @@@@@ +
+@@@@@@@+
+ @@@@@ +
+  @@@  +
+   @   +
+++++++++
```

```
1ØØ REM  Diamond
1Ø5 HOME
11Ø HTAB 12: VTAB 8
12Ø PRINT "+++++++++"
13Ø HTAB 12: VTAB 9
14Ø PRINT "+   @   +"
15Ø HTAB 12: VTAB 1Ø
16Ø PRINT "+  @@@  +"
17Ø HTAB 12: VTAB 11
18Ø PRINT "+ @@@@@ +"
19Ø HTAB 12: VTAB 12
2ØØ PRINT "+@@@@@@@+"
21Ø HTAB 12: VTAB 13
22Ø PRINT "+ @@@@@ +"
```

```
230 HTAB 12: VTAB 14
240 PRINT "+  @@@  +"
250 HTAB 12: VTAB 15
260 PRINT "+   @   +"
270 HTAB 12: VTAB 16
280 PRINT "+++++++++"
```

—Line 100 names the program.
—Line 105 clears the screen.
—Line 110 sets display coordinates for line 120 at Column 12, Row 8.
—Line 120 displays nine plus signs.
—Line 130 sets display coordinates for line 140 at Column 12, Row 9.
—Line 140 displays a plus sign, three spaces, @, three spaces, and another plus sign.
—Line 150 sets display coordinates for line 160 at Column 12, Row 10.
—Line 160 displays a plus, two spaces, three @'s, two spaces, and another plus sign.
—Line 170 sets display coordinates for line 180 at Column 12, Row 11.
—Line 180 displays a plus, one space, five @'s, another space, and another plus sign.
—Line 190 sets the display coordinates for line 200 at Column 12, Row 12.
—Line 200 displays a plus, seven @'s, and another plus sign.
—Line 210 sets the display coordinates for line 220 at Column 12, Row 13.
—Line 220 displays a plus, a space, five @'s, a space, and another plus sign.
—Line 230 sets the display coordinates for line 240 at Column 12, Row 14.
—Line 240 displays a plus, a space, three @'s, a space, and another space.
—Line 250 sets the display coordinates for line 260 at Column 12, Row 14.
—Line 260 displays a plus, three spaces, an @, three more spaces, and another plus sign.
—Line 270 sets the display coordinates for line 280 at Column 12, Row 16.
—Line 280 displays nine pluses.

Now, try your own hand at graphics in the text mode. It is easy and fun.

LOW-RESOLUTION GRAPHICS

In Chapter 3, we learned how to display colors on the screen. Now we are ready to do some real graphics. But first, we should review the low-res screen and display commands. The low-res screen is made up of forty columns by forty rows. Our graphics will be placed on the screen using column and row coordinates. Remember both the rows and columns are numbered from 0 to 39, not 1 to 40.

Low-Res Display Command Review

GR—Activates the low-res mode.

HLIN—Draws a horizontal line. You must include the beginning column location, the ending column location, and the row location. Like this:

```
HLIN 1Ø,15 AT 5
```

This statement in a program will display a line from Column 10 to 15, at Row 5.

VLIN—Draws a vertical line. Again you must include column and row coordinates. First state the beginning row location and the ending row location and, then, the column location. For example:

```
VLIN 15,25 AT 1Ø
```

This statement in a program will draw a line from Row 15 to Row 25 at Column 10.

COLOR=—This command defines the color the Adam computer is to use.

PLOT—This command will display specific screen locations. Enter the command followed by the column number, a comma, and then the row number.

Now let's have some fun.

Tic-Tac-Toe

Let's begin with something simple, like a Tic-Tac-Toe board.

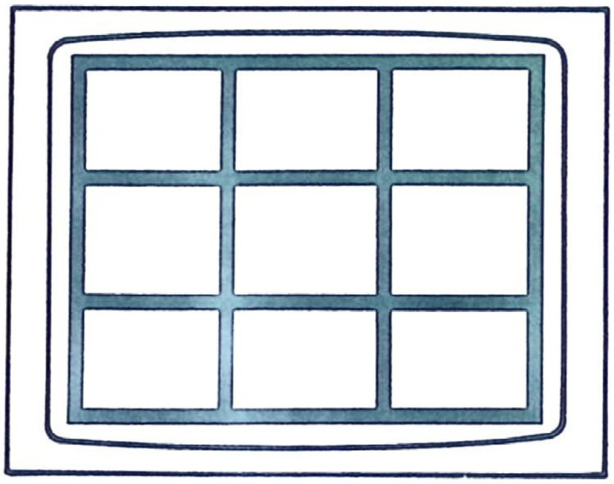


Fig. 5-2. Tic-Tac-Toe board.

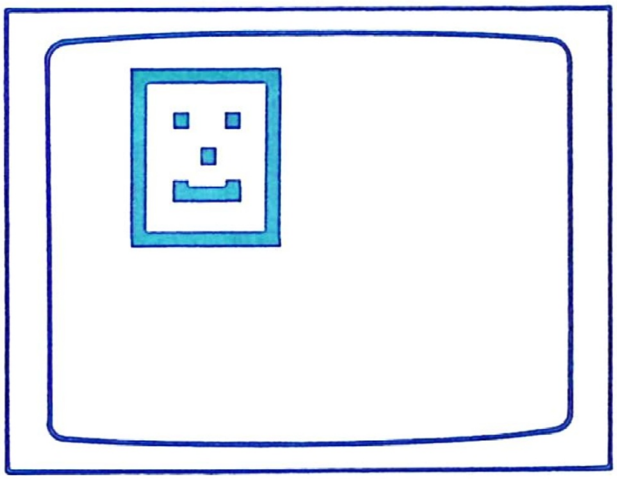
```
100 REM  Tictac
110 GR: COLOR = 2
120 HLIN 0,39 AT 0
130 HLIN 0,39 AT 13
140 HLIN 0,39 AT 26
150 HLIN 0,39 AT 39
160 VLIN 0,39 AT 0
170 VLIN 0,39 AT 13
180 VLIN 0,39 AT 26
190 VLIN 0,39 AT 39
```

—Line 100 names the program.
—Line 110 activates the low-res mode and defines the plotting color as dark blue.
—Lines 120 to 150 plot horizontal lines from Column 0 to Column 39 at Rows 0, 13, 26, and 39.
—Lines 160 to 190 plot vertical lines from Row 0 to Row 39 at Columns 0, 13, 26, and 39.

When you run this program, you will see a boxed 3 x 3 Tic-Tac-Toe playing area. We will use this playing board later in the traditional games chapter.

Happy Face

Let's try a smiley face with color.



```
100 REM  Happy
110 GR: COLOR = 15
120 HLIN 10,30 AT 5
130 HLIN 10,30 AT 25
140 VLIN 5,25 AT 10
150 VLIN 5,25 AT 30
155 COLOR = 7
160 FOR y = 13 TO 14: FOR x = 8 TO 9
170 PLOT y,x: NEXT x: NEXT y
180 FOR y = 26 TO 27: FOR x = 8 TO 9
190 PLOT y,x: NEXT x: NEXT y
200 FOR y = 19 TO 21: FOR x = 12 TO 13
210 PLOT y,x: NEXT x: NEXT y
220 FOR y = 13 TO 27: FOR x = 19 TO 20
225 PLOT y,x: NEXT x: NEXT y
230 FOR y = 13 TO 14: FOR x = 17 TO 18
240 PLOT y,x: NEXT x: NEXT y
250 FOR y = 26 TO 27: FOR x = 17 TO 18
260 PLOT y,x: NEXT x: NEXT y
```

—Line 100 names the program.
—Line 110 activates the low-res mode and defines the plotting color as white.
—Lines 120 to 150 plot the square around the face.

—Line 155 changes the plotting color to light blue.
—Line 160 to 190 plot the eyes.
—Lines 200 and 210 plot the nose.
—Lines 220 to 260 plot the smile.

This program could be used to reward a correct answer in a quiz or game. Try making the face frown instead of smiling. Or, how about adding a beard or some hair.

MORE FUN WITH LOW-RESOLUTION GRAPHICS

A game that helps you identify the different flags of the world might be useful. Let's use the low-res graphics capability of the Adam computer and display some flags.

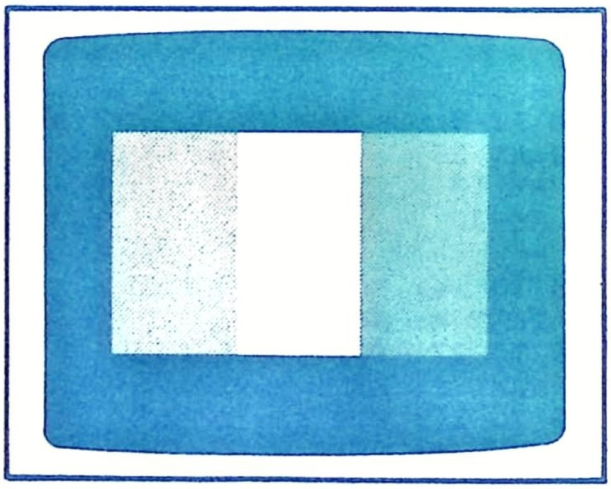


Fig. 5-3. Graphics and flags.

```
100 REM  Italy
110 GR: COLOR = 4
120 FOR x = 0 TO 39: FOR y = 1 TO 12
130 PLOT y,x: NEXT y: NEXT x
140 COLOR = 15
150 FOR x = 0 TO 39: FOR y = 13 TO 24
160 PLOT y,x: NEXT y: NEXT x
```

```
170 COLOR = 3
180 FOR x = Ø TO 39: FOR y = 25 TO 36
190 PLOT y,x: NEXT y: NEXT x
```

—Line 100 names the program.
—Line 110 activates the low-res mode and defines the plotting color as dark green.
—Lines 120 and 130 plot the green portion of the flag from Row 0 to 39 and from Column 1 to 12.
—Line 140 defines the plotting color as white.
—Lines 150 and 160 plot the white portion of the flag from Row 0 to 39 and from Column 13 to 24.
—Line 170 defines the plotting color as dark red.
—Lines 180 and 190 plot the red portion of the flag from Row 0 to 39 and Column 25 to 36.

When you run this program, your Adam computer will proudly display the colors of Italy. If you make a few changes in the program, you can display the flags of other countries. For example:

- Change the color in line 110 to 2 and you will see the flag of France.
- Change line 170 to color=11 (light red) and change line 110 back to 4 and you have the Irish flag.
- By changing the color in line 110 to 0, the color in line 140 to 13, and the color in line 170 to 3, you can display the colors of Belgium.

Now, let's try another flag with a different design. Type in the following:

```
100 REM  Sweden
110 GR: COLOR = 7
120 FOR y = 1 TO 38: FOR x = 1 TO 38
130 PLOT y,x: NEXT x: NEXT y
135 COLOR = 13
140 FOR x = 1 TO 38: FOR y = 12 TO 13
150 PLOT y,x: NEXT y: NEXT x
160 FOR y = 1 TO 38: FOR x = 18 TO 19
170 PLOT y,x: NEXT x: NEXT y
```

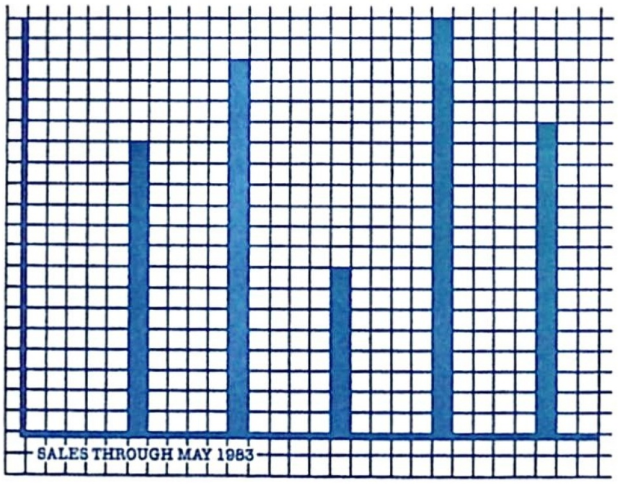
—Line 100 names the program.
—Line 110 activates the low-res mode and defines the plotting color as light blue.

—Lines 120 and 130 color the screen light blue.
—Line 135 defines the plotting color as yellow.
—Lines 140 to 170 plot the yellow cross on the flag.

Just as before, if you change the colors, you can display even more flags.
For example, if you change the color in line 110 to 3 and the color in line 135 to 15, you will see the flag of Denmark. To see the flag of Finland, change the color in line 110 to 15 and the color in line 135 to 7.
You might want to design your own flag or emblem or, perhaps, display the emblem for an organization that you belong to on Adam's screen.

GRAPHS

Let's try displaying a graph. You can use this program to display anything in graph form. Just change the numbers.



Input the following program:

```
100 REM  Graph
110 GR: COLOR = 15
120 FOR x = 5 TO 30
130 PLOT 5,x: NEXT x
140 FOR y = 6 TO 35
150 PLOT y,30: NEXT y
160 FOR x = 11 TO 29
```

```
170 COLOR = 2: PLOT 10,x; NEXT x
180 FOR x = 8 TO 29
190 COLOR = 1: PLOT 16,x: NEXT x
200 FOR x = 18 TO 29
210 COLOR = 4: PLOT 22,x: NEXT x
220 FOR x = 5 TO 29
230 COLOR = 13: PLOT 28,x: NEXT x
240 FOR x = 10 TO 29
250 COLOR = 7: PLOT 33,x: NEXT x
260 PRINT "SALES THROUGH MAY 1983"
```

—Line 100 names the program.
—Line 110 activates the low-res mode and defines the plotting color as white.
—Lines 120 to 150 plot the white axis of the graph.
—Line 160 sets the row numbers at 11 to 29.
—Line 170 sets the plotting color at dark blue and plots the blue bar in the graph.
—Lines 180 and 190 color and display the magenta bar in the graph.
—Lines 200 and 210 color and display the green bar in the graph.
—Lines 220 and 230 color and display the yellow bar in the graph.
—Lines 240 and 250 color and display the light blue bar in the graph.
—Line 260 displays the legend at the bottom of the screen.

You can change the legend and the plotting points to represent anything you like.

Here is another way to display numbers graphically. This display represents different components of your home budget. Think of the solid bar as your total home budget. The different colors represent the various percentages of your budget. Input the following program:

```
100 REM  Bar
110 GR: COLOR = 1
120 FOR x = 5 TO 10: FOR y = 15 TO 25
130 PLOT y,x; NEXT y: NEXT x
140 COLOR = 9
150 FOR x = 11 TO 20: FOR y = 15 TO 25
160 PLOT y,x: NEXT y: NEXT x
170 COLOR = 12
180 FOR x = 21 TO 35: FOR y = 15 TO 25
190 PLOT y,x: NEXT y: NEXT x
200 PRINT "HOME BUDGET BREAKDOWN"
210 PRINT "M=FOOD R=ENERGY G=RENT"
```

—Line 100 names the program.
—Line 110 activates the low-res mode and defines the plotting color as magenta.
—Lines 120 and 130 plot the magenta part of the graph from Row 5 to 10 and Column 15 to 25.
—Line 140 colors the next part of the display in red.
—Lines 150 and 160 plot the red portion of the graph from Row 11 to 20 and Column 15 to 25.
—Line 170 colors the next part of the display green.
—Lines 180 and 190 plot the green portion of the graph from Row 21 to 35 and Column 15 to 25.
—Lines 200 and 210 display the legend at the bottom of the screen.

Here is our last low-res program. We think it demonstrates Adam's artistic ability. We encourage you to create your own masterpieces. Be sure to draw your picture on graph paper first; if you don't, you will probably make some plotting mistakes. We also recommend that you run the program after each plot statement is entered. This makes it much easier to find and correct your plotting errors.

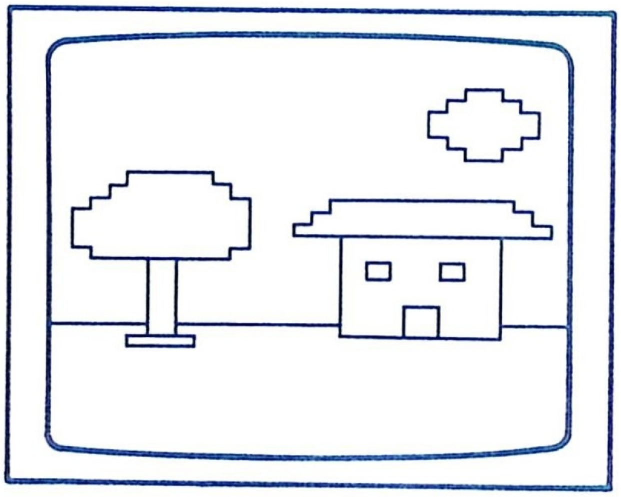


Fig. 5-4. Cabin and tree.

Enter the following program:

```
100 REM  Cabin
110 GR: COLOR = 12
120 FOR x = 31 TO 39: FOR y = 0 TO 39
```

```
130 PLOT y,x: NEXT y: NEXT x
140 FOR x = 20 TO 23: FOR y = 3 TO 14
150 PLOT y,x: NEXT y: NEXT x
160 FOR y = 6 TO 12: PLOT y,17: NEXT y
170 FOR y = 5 TO 13: PLOT y,18: NEXT y
180 FOR y = 4 TO 14: PLOT y, 19: NEXT y
190 FOR y = 4 TO 13: PLOT y, 24: NEXT y
200 COLOR = 5: FOR x = 25 TO 32: FOR y = 8 TO 9
210 PLOT y,x: NEXT y: NEXT x
220 FOR y = 7 TO 10: PLOT y,33: NEXT y
230 COLOR = 11: FOR x = 22 TO 32: FOR y = 21 TO
    35
240 PLOT y,x; NEXT y: NEXT x
250 FOR y = 18 TO 38: COLOR = 10: PLOT y,21: NEXT
    y
260 FOR y = 19 TO 37: PLOT y,20: NEXT y
270 FOR y = 20 TO 36: PLOT y,19 : NEXT y
280 COLOR = 7: FOR x = 30 TO 32: FOR y = 27 TO
    28:  PLOT y,x
285 NEXT y: NEXT x
290 COLOR = 13: FOR x = 24 TO 25: FOR y = 23 TO
    24:  PLOT y,x
295 NEXT y: NEXT x
300 FOR x = 24 TO 25: FOR y = 32 TO 33: PLOT y,x:
    NEXT  y: NEXT x
310 FOR x = 4 TO 6: FOR y = 33 TO 38: PLOT y,x:
    NEXT  y: NEXT x
320 FOR y = 34 TO 37: PLOT y,3: PLOT y,7: NEXT y
330 FOR y = 35 TO 36: PLOT y,2: PLOT y,8: NEXT y
```

—Line 100 names the program.
—Line 110 activates the low-res mode and sets the color to be green.
—Lines 120 and 130 plot the grass.
—Lines 140 to 190 plot the tree leaves.
—Lines 200 to 220 plot the tree trunk.
—Lines 230 and 240 plot the red portions of the house.
—Lines 250 to 270 plot the roof on the house.
—Lines 280 and 285 color the door blue.
—Lines 290 and 295 display the windows on the house.
—Lines 300 to 330 display the sun.

When you run this program, you will see a neat little cabin with a lovely tree in the front yard. You might want to put some apples on the tree, or maybe a picket fence around the house.

Maybe you prefer to sketch a cemetery with headstones. Or, an ocean scene might be nice. Test your imagination and have fun.

HIGH-RESOLUTION GRAPHICS

We talked about the high-res mode earlier in the color chapter. But, let's review the basic concepts again.

High-Res Screen

There are two different high-res screens. One has a 4-line text buffer at the bottom and one does not.

High-res screen No. 1 has 256 columns and 159 rows that we can use to plot graphics. This screen also has a space for 4 lines of text at the bottom of the screen. This screen is activated by the HGR command.

The second high-res screen has 256 columns and 192 rows. There is no text space on this screen. This full-graphics screen is activated by the HGR2 command.

Now that we understand the high-res screens, let's review the high-res display commands.

HPLOT

We will display all of our high-res graphics with the HPLOT command. Here is how it works.

HPLOT 200,50 will plot a point at Column 200, Row 50. We can also draw lines using HPLOT. The Adam computer will draw horizontal, vertical, or diagonal lines with HPLOT. All you need to do is tell the Adam where to begin drawing and where to stop. For example,

```
HPLOT 100,50 TO 150,50
```

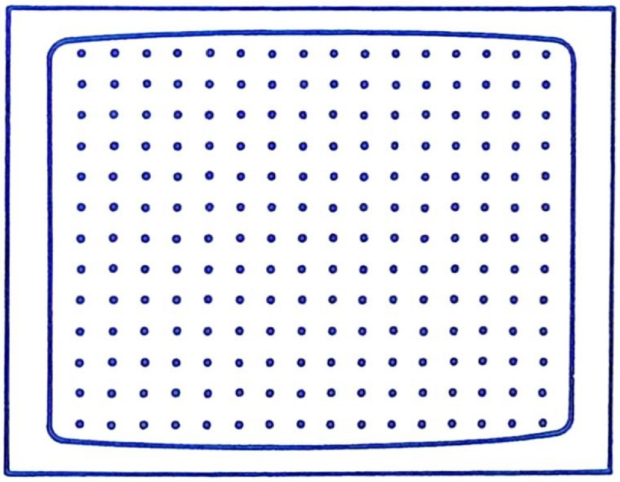
will draw a line at Row 50 from Column 100 to 150.

Try this:

```
100 HGR: HCOLOR = 1
110 HPLOT 100,50 TO 150,50
120 HPLOT TO 200,100
```

Note that line 120 uses the last coordinate in line 110 as its beginning coordinate. This is a very useful feature as you will soon see. Now, let's do some drawing with high-resolution graphics.

Let's begin with some very simple line graphics. Our first high-res program plots a dot grid. Try this:



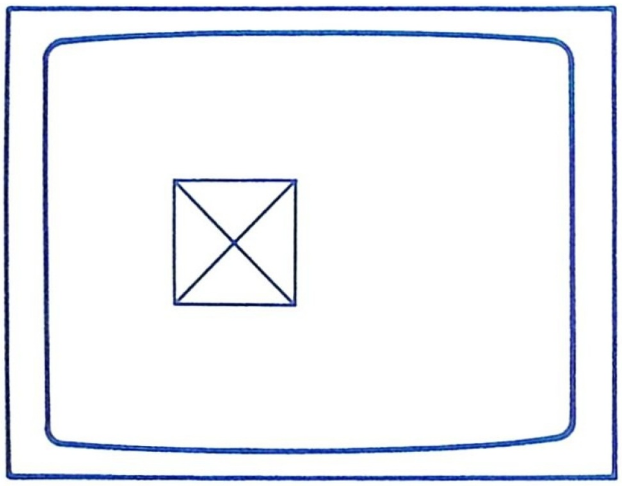
```
100 REM  Graph Grid
110 HGR: HCOLOR = 3
120 FOR x = 0 TO 160 STEP 10
130 FOR y = 0 TO 250 STEP 10
140 HPLOT y,x: NEXT y: NEXT x
```

—Line 100 names the program.
—Line 110 activates the high-res mode and sets the color to be white.
—Line 120 states the row-variable x as 0 to 160 in steps of 10.
—Line 130 states the column-variable y as 0 to 250 in steps of 10.
—Line 140 plots a point at each y,x coordinate.

When you run this program you will see a whole screen full of dots. The dots will be 10 pixels, or screen locations, apart. You can change the number following the STEP command and alter the spacing of the dots.

Now, let's try plotting some lines instead of dots. Type in the following program:

```
100 REM  Across
110 HGR: HCOLOR = 3
120 HPLOT 10,0 TO 240,0
130 HPLOT TO 240,150
140 HPLOT 10,0 TO 10,150
150 HPLOT TO 240,150
160 HPLOT 10,0 TO 240,150
170 HPLOT 240,0 TO 10,150
```



—Line 100 names the program.
—Line 110 activates the high-res mode and sets the color to be white.
—Line 120 plots a line from Column 10, Row 0 to Column 240, Row 0.
—Line 130 uses the last point in line 120 as the beginning point which, in this case, is Column 240, Row 0. Thus, the line will be plotted from Column 240, Row 0 to Column 240, Row 150.
—Line 140 plots a line from Column 10, Row 0 to Column 10, Row 150.
—Line 150 plots a line from Column 10, Row 150 to Column 240, Row 150. Again, the beginning point for this line is the ending point in line 140.
—Line 160 plots a line from Column 10, Row 0 to Column 240, Row 150.
—Line 170 plots a line from Column 240, Row 0 to Column 10, Row 150.

Lines 160 and 170 plot the diagonal points. Remember that when HPLOT TO is used, the Adam computer uses the last plotted point that is stated in the previous line at the beginning plot point for the current line. This feature reduces your inputting time.

When this program runs, you should see a square divided by two diagonal lines. Try your hand at adding more lines. Try dividing the square into two equal parts. Draw your new lines on graph paper. Figure out your column and row coordinates and see what you can do. If you don't have any graph paper, we suggest you get some. It is frustrating and time consuming to plot a picture or figure without drawing it on graph paper first.

Next, let's try a three-dimensional illusion. We call this program *3-D*.

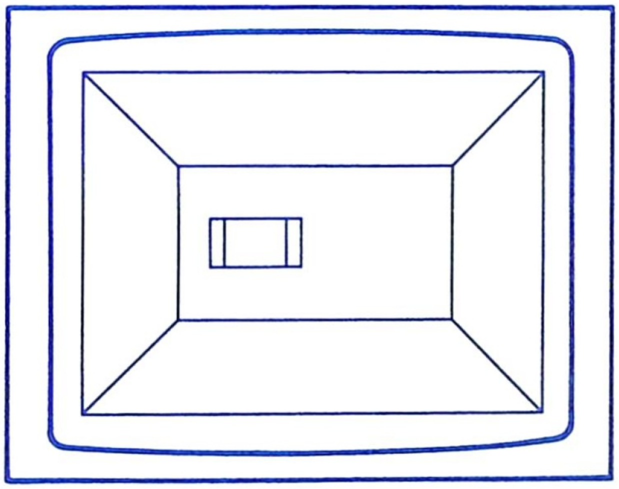


Fig. 5-5. 3-D drawing of a room.

```
100 REM  3-D
110 HGR: HCOLOR = 3
120 HPLOT 10,0 TO 240,0
130 HPLOT TO 240,150
140 HPLOT 10,0 TO 10,150
150 HPLOT TO 240, 150
170 HPLOT 85,50 TO 155,50
180 HPLOT TO 155,100
190 HPLOT TO 85,100
200 HPLOT TO 85,50
```

```
210 HPLOT 10,0 TO 85,50
220 HPLOT 240,0 TO 155,50
230 HPLOT 240,150 TO 155,100
240 HPLOT 10,150 TO 85,100
250 HPLOT 60,65 TO 60,85
260 HPLOT TO 25,110
270 HPLOT TO 25,40
280 HPLOT TO 60,65
```

—Line 100 names the program.
—Line 110 activates the high-resolution mode and sets the color to be white.
—Lines 120 to 280 plot the lines that make up our simulated room. All of the lines work exactly the same way. The only difficult part about this program is figuring out the correct plotting points.

Why don't you add a door and maybe another window. Don't worry if you make a mistake. Erase the incorrect line and try again. You will be amazed how quickly you can plot shapes with a little practice.

Now, using what we have learned, let's get a bit more complicated.

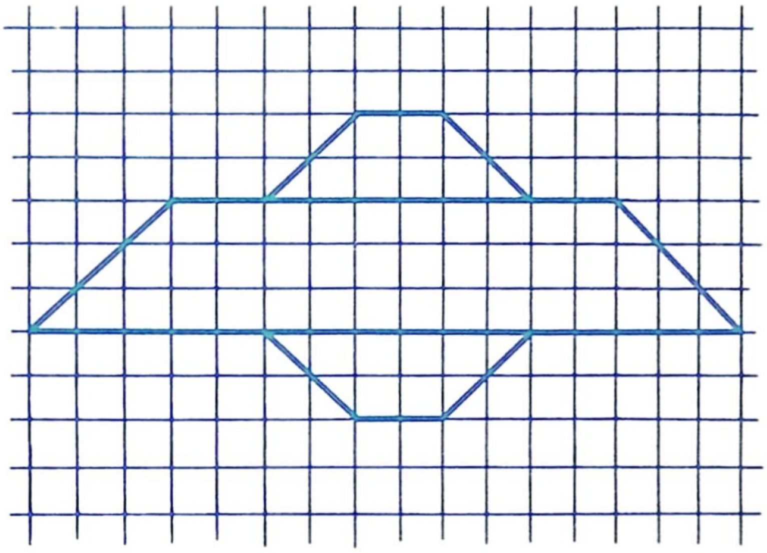
PLOTTING ARCADE FIGURES

The next several programs will be very useful to you in later chapters. In Chapter 6, we will show you how to animate, or move, these characters. And then, in Chapter 10, we will use them in some arcade games.

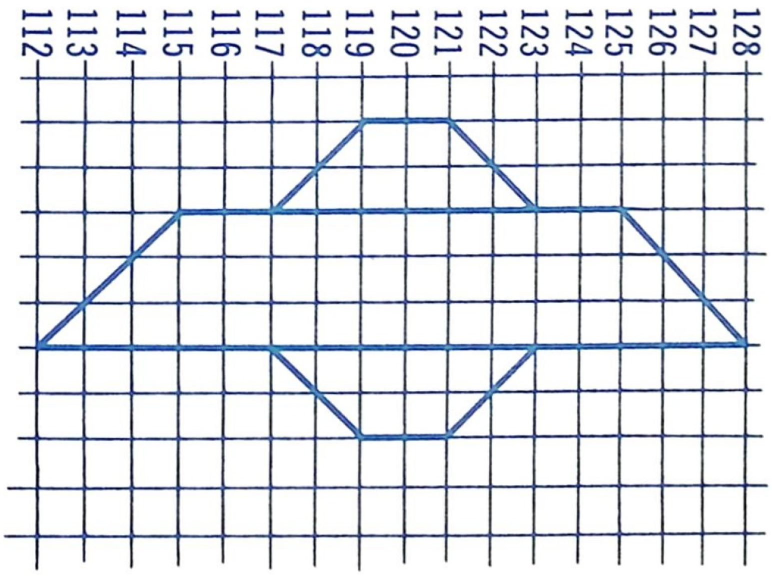
As you will see, we used variables to define our plotting points rather than the actual column and row coordinates. When we discuss animation in the next chapter, you will see why we use variables. At first glance, these next programs will look confusing. But with some graph paper and a little practice, they are not difficult.

Before we get to the programs, let's look at how we figured out the coordinates.

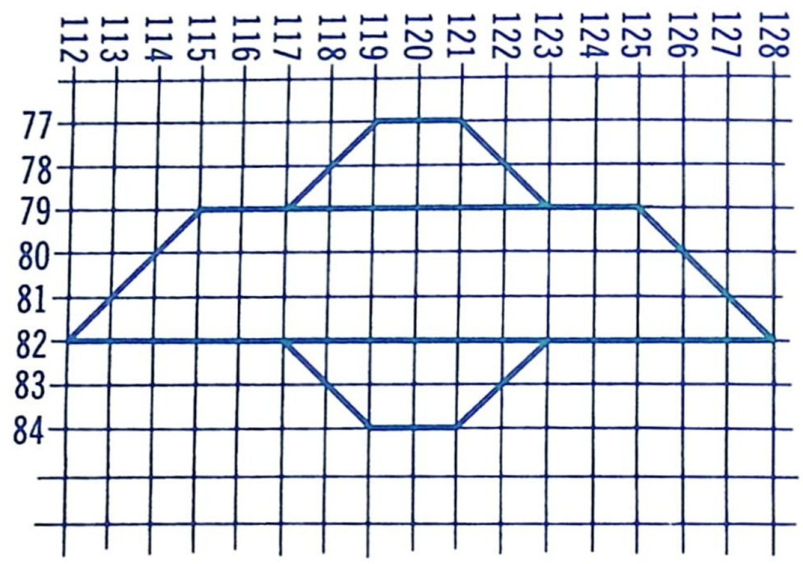
STEP 1—First, we drew our figures on graph paper. You don't have to be an artist as the general shape will do. Here is what our saucer figure looked like.



STEP 2—Once the figure was drawn, we found the middle of the figure and labeled it Column 120, because we wanted the display to be in the middle of the screen. Then we wrote in the column numbers for each graph line, all the way across the figure. Our drawing now looked like this.



STEP 3—We did the same thing for the row numbers—like this.



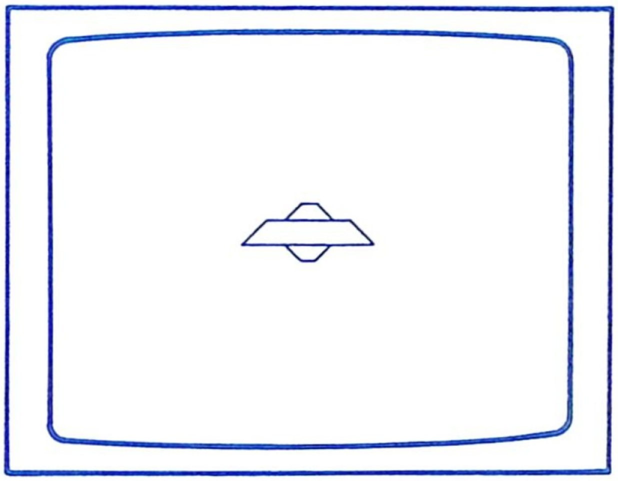
STEP 4—We arbitrarily set a value for X (column) and Y (row). We selected 112 as X and 84 as Y.

STEP 5—Finally, by counting the points, we were able to plot all of the coordinates in terms of our X and Y variables. For example, if we wanted to begin plotting at Column 117, we stated 117 as X + 5.

We suggest that you input the programs that follow and try to follow our plotting on a piece of graph paper. You should save all of these programs on your digital data pack because we will use them again in later chapters.

Saucer

Let's try a saucer program.



```
50 REM  Saucer
100 HGR: HCOLOR = 3
110 X = 112: y = 84
120 HPLOT x,y-2 TO x+16, y-2
130 HPLOT x+16,y-2 TO x+13, y-5
140 HPLOT x+13,y-5 TO x+3,y-5
150 HPLOT x+3,y-5 TO x,y-2
160 HPLOT x+5,y-5 TO x+7,y-7
170 HPLOT TO x+9,y-7
180 HPLOT TO x+11, y-5
190 HPLOT x+5,y-2 TO x+7,y
200 HPLOT TO x+9,y
210 HPLOT TO x+11,y-2
```

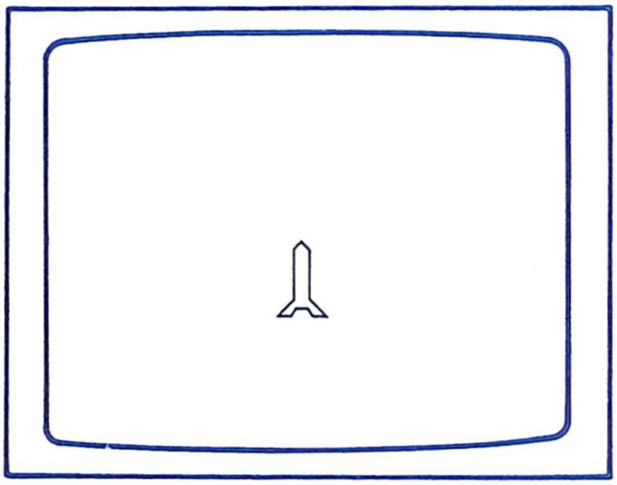
—Line 50 names the program.
—Line 100 activates the high-res mode and sets the color to be white.
—Line 110 states the column, or x variable, at 112 and the row, or y variable, at 84.
—Line 120 plots a line from x (or Column 112), Row y−2 (or Row 82) to x+16 (Column 128), Row y−2 (Row 82). See how it works?
—Line 130 plots a line from x+16 (Column 128), Row y−2 (Row 82) to x+13 (Column 125), Row y−5 (Row 79).

—Lines 140 to 210 work just like lines 120 and 130. If nothing else, plotting shapes like this will do wonders for your addition and subtraction skills.

When you run this program, you will be rewarded for your effort.

Rocket

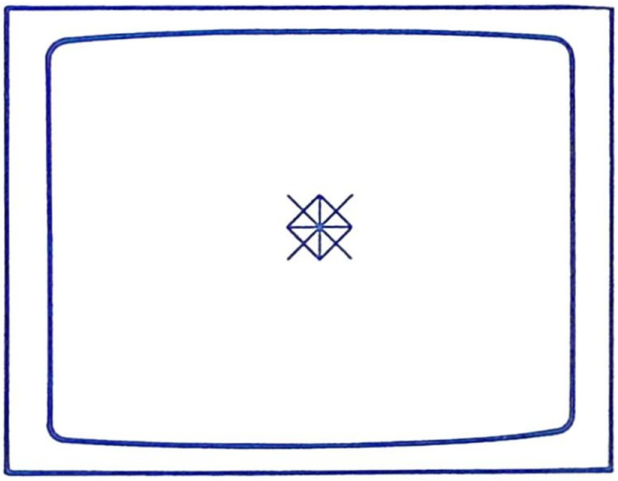
Here is another out-of-this-world program. This time we will display a rocket. Since this program works just like the *Saucer* program, we will not describe each line.



```
50 REM  Rocket
100 HGR: HCOLOR = 3
110 x = 112: y = 127
120 HPLOT x+8,y TO x+12,y+4
130 HPLOT x+8,y TO x+4,y+4
140 HPLOT TO x+4,y+19
150 HPLOT TO x,y+23
160 HPLOT TO x+4,y+23
170 HPLOT TO x+6,y+21
180 HPLOT TO x+10,y+21
190 HPLOT TO x+12,y+23
200 HPLOT TO x+16,y+23
210 HPLOT TO x+12,y+18
220 HPLOT TO x+12,y+3
```

Asteroid

Before we come back to earth, let's take a look at a program for designing an asteroid.

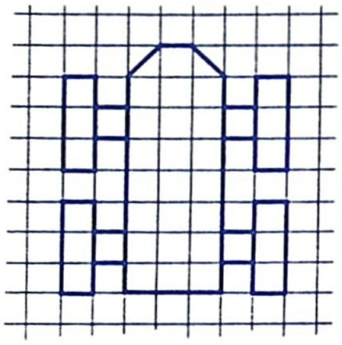


```
100 REM Asteroid
110 HGR: HCOLOR = 3
120 x = 16: y = 36
130 HPLOT x+4,y TO x+8,y+4
140 HPLOT TO x+4,y+8
150 HPLOT TO x,y+4
160 HPLOT TO x+4,y
170 HPLOT x,y+8 TO x+8,y
180 HPLOT x,y TO x+8,y+8
```

Again, this program works just like the last two.

Car

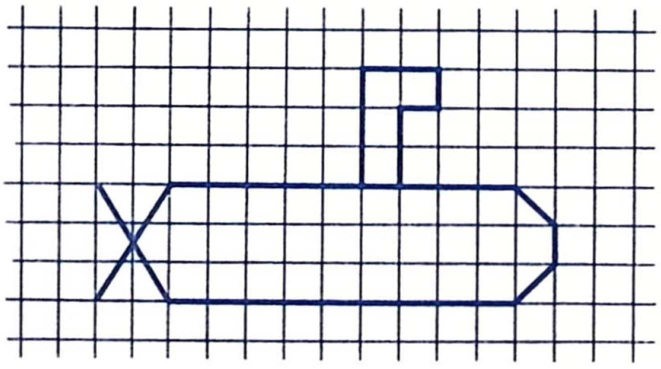
Enough about space and the great beyond. The next program pulls us back to earth and displays a sleek racing machine. Try the following short program.



```
100 REM Car
110 HGR: HCOLOR = 3
120 x = 114: y = 112
130 HPLOT x+5,y TO x+3,y+2
140 HPLOT TO x+3,y+19
150 HPLOT TO x+9,y+19
160 HPLOT TO x+9,y+2
170 HPLOT TO x+7,y
180 HPLOT TO x+5,y
190 HPLOT x,y+3 TO x,y+8
200 HPLOT x+1,y+3 TO x+1,y+8
210 HPLOT x+11,y+3 TO x+11,y+8
220 HPLOT x+12,y+3 TO x+12,y+8
230 HPLOT x,y+13 TO x,y+18
240 HPLOT x+1,y+13 TO x+1,y+18
250 HPLOT x+11,y+13 TO x+11,y+18
260 HPLOT x+12,y+13 TO x+12,y+18
```

Submarine

If you love the sea, maybe you will prefer our next type of transportation.

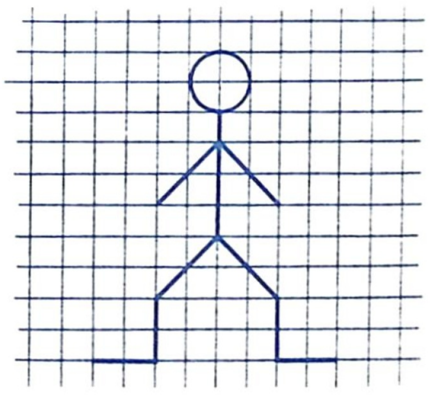


```
100 REM  Sub
110 HGR: HCOLOR = 3
120 x = 105: y = 30
130 HPLOT x+6,y+14 TO x+27,y+14
140 HPLOT TO x+29,y+13
150 HPLOT TO x+29,y+7
160 HPLOT TO x+27,y+6
170 HPLOT TO x+6,y+6
180 HPLOT TO x,y+12
190 HPLOT TO x,y+7
200 HPLOT TO x+6,y+14
210 REM  Periscope
220 HPLOT x+18,y+6 TO x+18,y
230 HPLOT TO x+24,y
240 HPLOT TO x+24,y+3
250 HPLOT TO x+22,y+3
260 HPLOT TO x+22,y+6
```

When you run this program, you should see our deadly U-Boat, periscope and all.

Before we move on, let's display a little stick figure. We call him:

JACK



```
100 REM  Jack
110 HGR: HCOLOR = 3
120 x = 74: y = 113
130 HPLOT x+1,y+11 TO x+4,y+11
140 HPLOT TO x+4,y+8
150 HPLOT TO x+7,y+6
```

```
160 HPLOT TO x+10,y+8
170 HPLOT TO x+10,y+10
180 HPLOT TO x+10,y+12
190 HPLOT x+7,y+6 TO x+7,y+2
200 HPLOT x+7,y+3 TO x+4,y+5
210 HPLOT x+7,y+3 TO x+10,y+5
220 HPLOT x+6,y TO x+7,y
230 HPLOT x+5,y+1 TO x+9,y+1
240 HPLOT x+6,y+2 TO x+7,y+2
```

DRAWING CIRCLES WITH THE ADAM COMPUTER

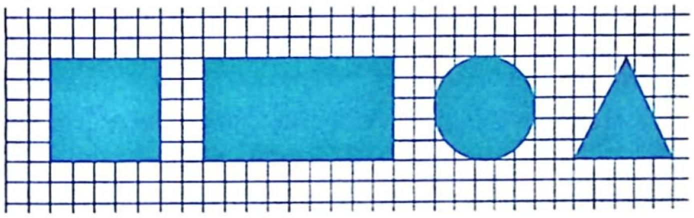
We have discussed Adam's ability to draw straight lines. Circles are even more fun. By using sine and cosine functions, we can draw any circle. Let's look at these functions in action. Type in the following program:

```
100 REM  Circle
110 HGR: HCOLOR = 3
120 INPUT "RADIUS OF THE CIRCLE ";r
130 px = 120: py = 80: br = 0: er = 7.7
135 j = .5
140 HPLOT px+r, py
150 FOR i = br TO er STEP j
160 x = r * COS(i) + px
170 y = -r * SIN(i) + py
180 HPLOT TO x,y
190 NEXT i
200 GOTO 120
```

—Line 100 names the program.
—Line 110 activates the high-res mode and sets the color to be white.
—Line 120 allows us to input the radius of the circle that we want to plot.
—Lines 130 to 135 state the variables.
  px is the column number at the center of the circle,
  py is the row number at the center of the circle,
  br is the radius to begin plotting,
  er is the radius to end the drawing,
  j is the step or increment at which the circle is drawn.
—Line 140 plots a point at the beginning column location plus the radius inputted in line 120 at the beginning row number.

—Line 150 states the value of i as the beginning radian to the ending radian in steps of j.
—Line 160 defines x, or the column, as the radius times the cosine of i + (the beginning column location).
—Line 170 defines y, or row, as −r (a negative radius) times the sine of i plus the beginning row number.
—Line 180 plots a line from px + r, py to x,y.
—Line 190 gets the next value of i.
—Line 200 loops back to line 120 so that we can select another radius.

Try several different circles. All of your circles will stay on the screen until you break the loop by pressing CONTROL and the C key.



Shapes

This program puts much of what we have covered in this chapter together. Our *Shapes* program asks you to input a letter. If you enter a c, the program will display a blue circle. If r is entered, an orange rectangle appears. When the letter t is entered, a green triangle appears. An s will display a yellow square. Enter the program and try it. The program tells you exactly what to do.

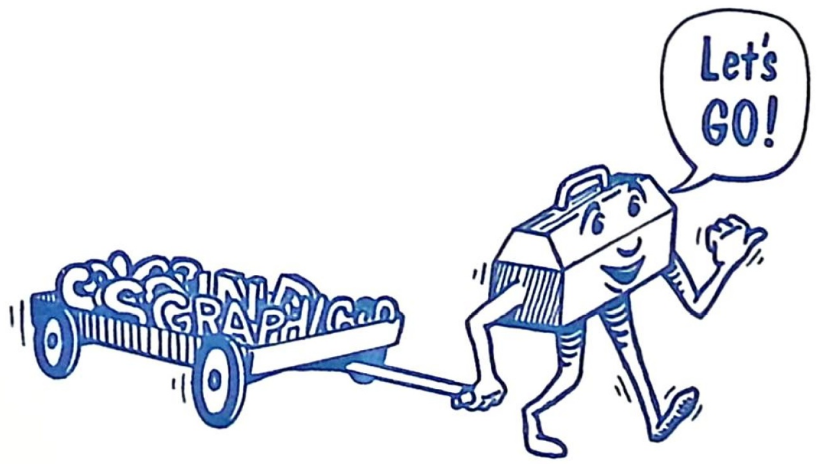
```
50 REM  Shapes
100 HOME
110 GOSUB 1000
120 INPUT a$
130 IF a$ = "s" OR a$ = "S" GOTO 200
140 IF a$ = "c" OR a$ = "C" GOTO 300
150 IF a$ = "r" OR a$ = "R" GOTO 400
160 IF a$ = "t" OR a$ = "T" GOTO 500
170 GOTO 120
200 HGR: HCOLOR = 13
210 FOR x = 110 TO 130: FOR y = 70 TO 90
220 HPLOT x,y: NEXT y: NEXT x
```

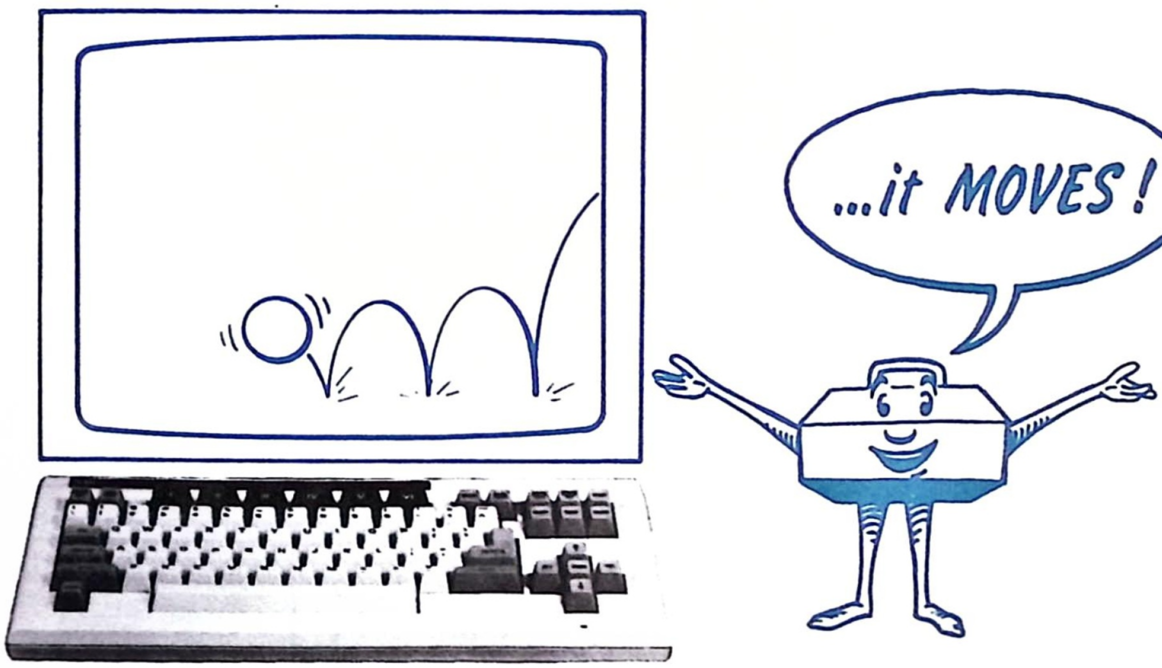
```
230 PRINT "THIS IS A YELLOW SQUARE."
240 FOR i = 1 TO 200: NEXT i
250 GOTO 120
300 HGR: HCOLOR = 6
310 cx = 120: cy = 80: r = 20
320 br = 0: er = 6.6: s = .3
330 HPLOT cx+r, cy
340 FOR i = br TO er STEP s
350 x = r * COS(i) + cx
360 y = -r * SIN(i) + cy
370 HPLOT TO x,y
380 NEXT i
390 PRINT "THIS IS A BLUE CIRCLE."
395 GOTO 120
400 HGR: HCOLOR = 5
410 FOR x = 110 TO 130: FOR y = 75 TO 85
420 HPLOT x,y
430 NEXT y: NEXT x
440 PRINT "THIS IS AN ORANGE RECTANGLE."
450 GOTO 120
500 HGR: HCOLOR = 12
510 HPLOT 110,80 TO 130,80
520 HPLOT TO 120,50
530 HPLOT TO 110,80
540 PRINT "THIS IS A GREEN TRIANGLE."
550 GOTO 120
1000 PRINT "THIS PROGRAM LETS YOU"
1010 PRINT
1020 PRINT "DISPLAY FOUR DIFFERENT"
1030 PRINT
1040 PRINT "GEOMETRIC SHAPES."
1050 PRINT
1060 PRINT "TO SELECT A SHAPE PRESS"
1070 PRINT
1080 PRINT "THE FIRST LETTER OF THE"
1090 PRINT "SHAPE YOU WANT TO SEE."
1100 PRINT
1110 FOR i = 1 TO 1000: NEXT i
1120 PRINT "THIS CHART WILL HELP"
1130 PRINT: PRINT: PRINT
1140 PRINT "CIRCLE","PRESS C"
1150 PRINT
1160 PRINT "SQUARE","PRESS S"
```

```
1170 PRINT
1180 PRINT "RECTANGLE","PRESS R"
1185 PRINT
1190 PRINT "TRIANGLE","PRESS T"
1195 PRINT
1200 PRINT "ALWAYS PRESS RETURN AFTER"
1210 PRINT
1220 PRINT "THE CORRECT LETTER."
1230 RETURN
```

—Line 50 names the program.
—Line 100 clears the screen.
—Line 110 branches to line 1000.
—Line 120 allows us to input our choice of shapes.
—Line 130 states that if s is pressed, the program branches to line 200.
—Line 140 says that if c is pressed, the program branches to line 300.
—Line 150 says that if r is pressed, the program branches to line 400.
—Line 160 states that if t is pressed, the program branches to line 500.
—Line 170 loops back to line 120.
—Lines 200 to 240 display the yellow square and print the prompt.
—Lines 300 to 390 plot the blue circle and print the prompt.
—Lines 400 to 440 plot the orange rectangle and print the prompt.
—Lines 500 to 540 plot the green triangle and print the prompt.
—Lines 1000 to 1220 print the prompts and directions that appear at the beginning of the program.
—Line 1230 returns to line 120.

We hope you enjoyed your journey through graphics. Next, we will look at how to animate your graphics. We are ready when you are.





Animation Subroutines 6

Animation, or moving graphics characters around the screen, is a lot of fun. And, although it may seem difficult, it really isn't. Once you have mastered the graphics subprograms that we discussed in Chapter 5, you are ready to animate.

It is helpful it you understand how the computer animates. Animation is created by plotting a character on the screen in one place, erasing it, and plotting the character again in another place. The computer does this three-step operation extremely quickly so all we see is the image, or character, blinking along.

ANIMATION IN ONE SCREEN LOCATION

We have used the FOR/NEXT loop quite a bit in earlier chapters. This loop is one way that the Adam draws and redraws a character. Let's try a very simple animation program. Input the following program.

```
5Ø REM  1 Spot
1ØØ HOME
11Ø FOR x = 1 TO 9
12Ø HTAB 1Ø: VTAB 1Ø
13Ø PRINT x
14Ø NEXT x
```

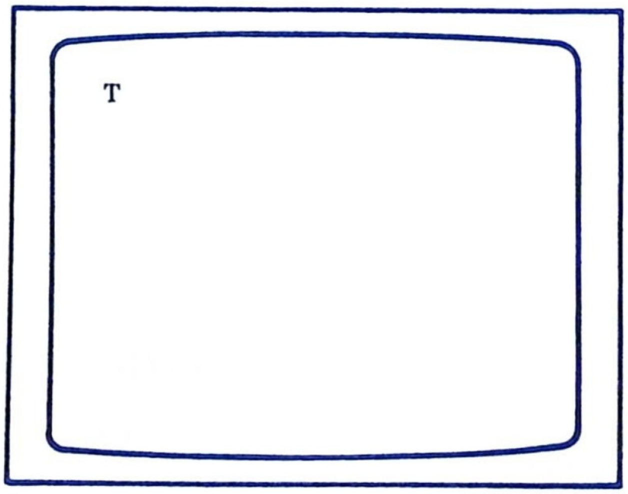


Fig. 6-1. T and I letters being displayed at one screen location.

On the first cycle of the loop, the program displays the number 1. The next cycle will display the number 2 and so on, through the number 9. Since only one character can normally occupy a screen location at a time, each new value of X "erases" the old one. By using this technique, you can give the illusion of movement.

Now, let's try something that is a little more fun than flashing numbers and, at the same time, is another way to do animation within one screen location. In this program, we will display the letters I and T alternately at the same screen location.

Type in the following program:

```
5Ø REM  I and T
1ØØ HOME
11Ø HTAB 1Ø: VTAB 1Ø
12Ø PRINT "I"
13Ø HTAB 1Ø: VTAB 1Ø
14Ø PRINT "T"
15Ø GOTO 11Ø
```

—Line 50 names the program.
—Line 100 clears the screen.
—Line 110 uses the HTAB and VTAB to identify the screen location at Row 10, Column 10.
—Line 120 prints the letter I at Row 10, Column 10.
—Line 130 sets the screen location at Row 10, Column 10.
—Line 140 prints the letter T at Row 10, Column 10.
—Line 150 loops back to line 110 and the T's and the I's will continue to flash.

When you run this program, you should see the I and T exchanging places at Row 10, Column 10. However, Adam works so fast that all you will really see is the top of the T flashing. You could add a timing loop and slow things down a little. Try adding:

```
125 FOR i = 1 TO 100:NEXT i
```

Now you should be able to see the letters change.

Whether we use FOR/NEXT loops or overprint two or more characters, the result is the same. If you are dealing with animating one screen location, you are simply drawing and redrawing over the character in that location.

ANIMATION IN THE LOW-RESOLUTION AND HIGH-RESOLUTION MODES

The rest of this chapter deals with animating in the low- and high-resolution modes. The principles of animation are the same as animating in the text mode, but the actual procedures are different.

In the graphics modes, the erasing process is done a little differently. We will erase a shape by plotting that shape in the background color. The Adam computer plots the shape even though we can't see it. So, to animate a shape, we first plot the shape in the desired display color, then plot the same shape in the background color, and then plot the shape in the display color again, only in a different screen location. Let's look at horizontal animation first.

HORIZONTAL ANIMATION

By using a FOR/NEXT loop, we can make objects move from one screen location to another. Let's look at a low-resolution example first.

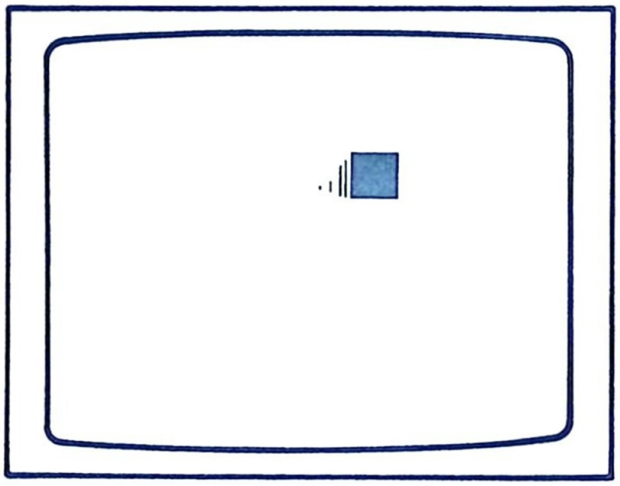


Fig. 6-2. Moving a block across the screen.

```
100 REM  Across
110 HOME: GR
120 FOR x = 1 TO 35
130 COLOR = 3
```

```
140 PLOT x,20
150 COLOR = 0
160 PLOT x, 20
170 NEXT x
```

—Line 100 names the program.
—Line 110 clears the screen and activates the low-resolution mode.
—Line 120 states the value of X (the column number) as Column 1 to Column 35. (NOTE: In reverse to the Row,Column method that other microcomputers use, the Adam computer uses a Column,Row system when entering an x,y axis statement; see your *SmartBASIC Programming Manual*.)
—Line 130 sets the display color as dark red.
—Line 140 plots the red block.
—Line 150 sets the color as 0.
—Line 160 plots a black block which erases the red block.
—Line 170 gets the next value of x.

Lines 130 and 140 display a red block and lines 150 and 160 plot a black block that "erases" the trail of the red block.

If you want the block to travel from right to left instead of from left to right, all you need to do is change line 120 so that it reads like this:

```
120 FOR x=35 TO 1 STEP -1
```

This change tells the computer that the first value of x is now 35, not 1. The STEP −1 tells the computer to move at intervals of one in a minus direction. In other words, the first location of our red block is Column 35, Row 20; the next location is Column 34, Row 20, etc. The computer will continue to subtract one from the last screen location and will move all the way across the screen. Now that we know how to move the block from left to right and from right to left, let's see if we can put the two directions together.

This new program sends the red block over to the right side of the screen and then brings it back to the left.

```
100 REM  Back+Forth
110 HOME: GR
120 FOR x = 1 TO 35
130 COLOR = 3: PLOT x, 20
140 COLOR = 0: PLOT x, 20
150 NEXT x
```

```
160 FOR x = 35 TO 1 STEP -1
170 COLOR = 3: PLOT x, 20
180 COLOR = 0: PLOT x, 20
190 NEXT x
```

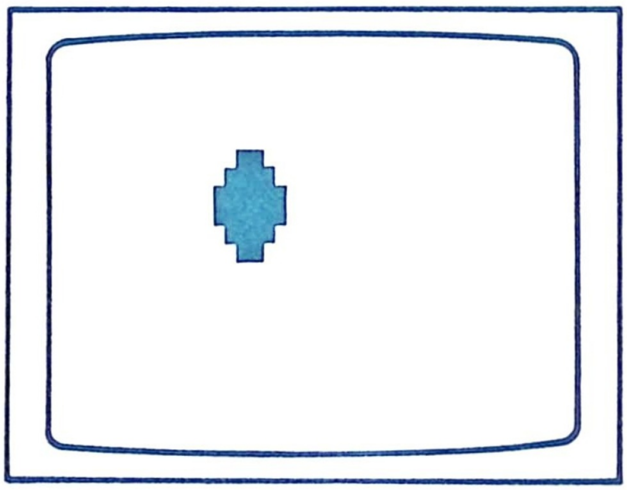
Not bad, huh? All we did was combine two FOR/NEXT loops together. Lines 120 to 150 display the red block as it moves from left to right and lines 160 to 190 display the block as it moves from right to left.

If you like, you can add another line to the program to slow the displaying process. Try adding either or both of these lines:

```
135 FOR i = 1 TO 50: NEXT i
175 FOR i = 1 TO 50: NEXT i
```

These lines will make your display smoother and easier to see.

Now let's try something a little more fun than moving blocks. Try the following program:



```
100 REM  Blob
110 HOME: GR
120 FOR x = 1 TO 29: y = 20
130 COLOR = 7:GOSUB 400
140 FOR i = 1 TO 40: NEXT i
150 COLOR = 0: GOSUB 400
160 NEXT x
170 END
400 HLIN x+2,x+3 AT y
410 HLIN x+1,x+4 AT y+1
```

```
420 HLIN x,x+5 AT y+2
430 HLIN x+1,x+4 AT y+3
440 HLIN x+2,x+3 AT y+4
450 RETURN
```

—Line 100 names the program.
—Line 110 clears the screen and activates the low-resolution screen.
—Line 120 sets the value of x (or column) as Column 1 to Column 29. Also, it sets the value of y (or row) at 20.
—Line 130 sets the color to light blue and then branches to line 400 and displays the blob.
—Line 140 is the timing loop that slows the display.
—Line 150 sets the color to black and branches to line 400 and erases the blob.
—Line 160 gets the next value of x.
—Line 170 ends the program.
—Lines 400 to 450 plot the blob in terms of the x and y variables.

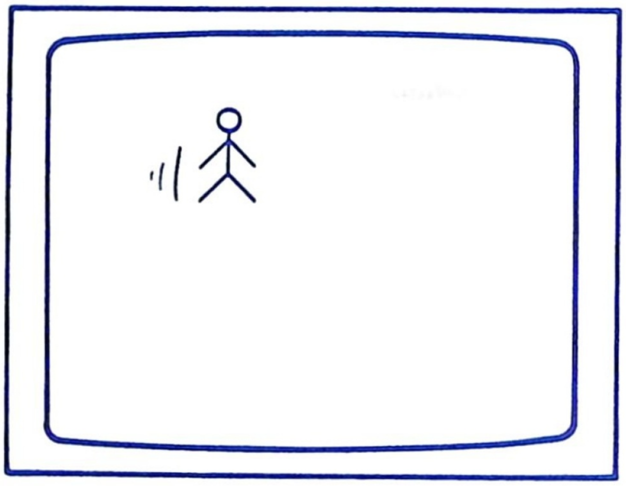
The plotting subroutine (lines 400–450) first plots the blob in blue and then in black. The colors are controlled by lines 130 and 150. Now let's move the blob back across the screen. Here is how:

```
100 REM  Blob Back
110 HOME: GR
120 FOR x = 1 TO 29: y = 20
130 COLOR = 7: GOSUB 400
140 FOR i = 1 TO 40: NEXT i
150 COLOR = 0: GOSUB 400
160 NEXT x
180 FOR x = 29 TO 1 STEP -1
190 COLOR = 7: GOSUB 400
200 FOR i = 1 TO 40: NEXT i
210 COLOR = 0: GOSUB 400
220 NEXT x
230 END
400 HLIN x+2,x+3 AT y
410 HLIN x+1,x+4 AT y+1
420 HLIN x,x+5 AT y+2
430 HLIN x+1,x+4 AT y+3
440 HLIN x+2,x+3 AT y+4
450 RETURN
```

—Lines 180 to 220 display the blob's trip back across the screen.

In line 180, the column variable is stated as 29 to 1 in increments of minus 1. Once the column variable is stated this way, all we need to do is plot the blob again. This is done in lines 190 and 200.

Before we move on, let's do a little horizontal animation in the high-resolution mode. Remember in the last chapter how we drew characters on the screen. Well, now we will make some of those characters move. First, let's draw two stick figures to use and call them "Jack." Input this program with 1000-series line numbers and save it. We will use this as a subroutine again later.



```
1000 REM  Jump
1010 HPLOT x-1,y TO x+1,y
1020 HPLOT x-1,y+1 TO x+1,y+1
1030 HPLOT x,y+1 TO x,y+4
1040 HPLOT TO x+3,y+8
1050 HPLOT x-3,y+7 TO x,y+4
1060 HPLOT x-3,y TO x-1,y+2
1070 HPLOT x+1,y+2 TO x+3,y+1
1080 RETURN
2000 REM  At Rest
2010 HPLOT x-1,y-1 TO x+1,y-1
2020 HPLOT x-1,y TO x+1,y
2030 HPLOT x,y+1 TO x,y+4
2040 HPLOT x-1,y+5 TO x-1,y+8
2050 HPLOT x+1,y+5 TO x+1,y+8
```

```
2060 HPLOT x-1,y+2 TO x-2,y+3
2070 HPLOT x+1,y+2 TO x+2,y+3
2080 RETURN
```

Now let's make Jack move. Enter the following program.

```
100 REM  Jack Over
110 HGR
120 FOR x = 10 TO 250 STEP 10
130 y = 130
140 HCOLOR = 3: GOSUB 1000
150 HCOLOR = 0: GOSUB 1000
160 HCOLOR = 3: GOSUB 2000
170 HCOLOR = 0: GOSUB 2000
180 NEXT x
190 FOR x = 250 TO 10 STEP -10
200 HCOLOR = 3: GOSUB 1000
210 HCOLOR = 0: GOSUB 1000
230 HCOLOR = 3: GOSUB 2000
240 HCOLOR = 0: GOSUB 2000
250 NEXT x
260 END
1000 REM  Jump
1010 HPLOT x-1,y TO x+1,y
1020 HPLOT x-1,y+1 TO x+1,y+1
1030 HPLOT x,y+1 TO x,y+4
1040 HPLOT TO x+3,y+8
1050 HPLOT x-3,y+7 TO x,y+4
1060 HPLOT x-3,y TO x-1,y+2
1070 HPLOT x+1,y+2 TO x+3,y+1
1080 RETURN
2000 REM  At Rest
2010 HPLOT x-1,y-1 TO x+1,y-1
2020 HPLOT x-1,y TO x+1,y
2030 HPLOT x,y+1 TO x,y+4
2040 HPLOT x-1,y+5 TO x-1,y+8
2050 HPLOT x+1,y+5 TO x+1,y+8
2060 HPLOT x-1,y+2 TO x-2,y+3
2070 HPLOT x+1,y+2 TO x+2,y+3
2080 RETURN
```

—Line 100 names the program.
—Line 110 activates the high-resolution mode.

—Line 120 states the column variable as Column 10 to Column 250 and instructs the computer to cycle in increments of 10.
—Line 130 states the row variable as 130.
—Line 140 sets the color to white and branches to line 1000 to display Jack jumping.
—Line 150 erases Jack jumping by plotting the display in black.
—Lines 160 and 170 work just like lines 140 and 150, except that they display and erase Jack at rest.
—Line 180 gets the next value of x.
—Line 190 states the column variable as 250 to 10.
—Lines 200 to 240 bring Jack back across the screen.
—Lines 1000 to 1080 display Jack in the jumping state.
—Lines 2000 to 2080 display Jack at rest.

Any character can be displayed in this way. We will do more work with high-resolution animation later on, but first, let's look at how to do vertical animation.

VERTICAL ANIMATION

Adam can animate vertically also. The process is pretty much the same except that your FOR/NEXT loops work with row numbers instead of column numbers.

Let's put Jack to work again. Type in the following program.

```
100 REM  Up and Down
110 HGR
120 FOR y = 160 TO 10 STEP -10
130 x = 120
140 HCOLOR = 3: GOSUB 1000
150 HCOLOR = 0: GOSUB 1000
160 HCOLOR = 3: GOSUB 2000
170 HCOLOR = 0: GOSUB 2000
180 NEXT y
190 FOR y = 10 TO 160 STEP 10
200 HCOLOR = 3: GOSUB 1000
210 HCOLOR = 0: GOSUB 1000
230 HCOLOR = 3: GOSUB 2000
240 HCOLOR = 0: GOSUB 2000
250 NEXT y
260 END
```

```
1000 REM  Jump
1010 HPLOT x-1,y TO x+1,y
1020 HPLOT x-1,y+1 TO x+1,y+1
1030 HPLOT x,y+1 TO x,y+4
1040 HPLOT TO x+3,y+8
1050 HPLOT x-3,y+7 TO x,y+4
1060 HPLOT x-3,y TO x-1,y+2
1070 HPLOT x+1,y+2 TO x+3,y+1
1080 RETURN
2000 REM  At Rest
2010 HPLOT x-1,y-1 TO x+1,y-1
2020 HPLOT x-1,y TO x+1,y
2030 HPLOT x,y+1 TO x,y+4
2040 HPLOT x-1,y+5 TO x-1,y+8
2050 HPLOT x+1,y+5 TO x+1,y+8
2060 HPLOT x-1,y+2 TO x-2,y+3
2070 HPLOT x+1,y+2 TO x+2,y+3
2080 RETURN
```

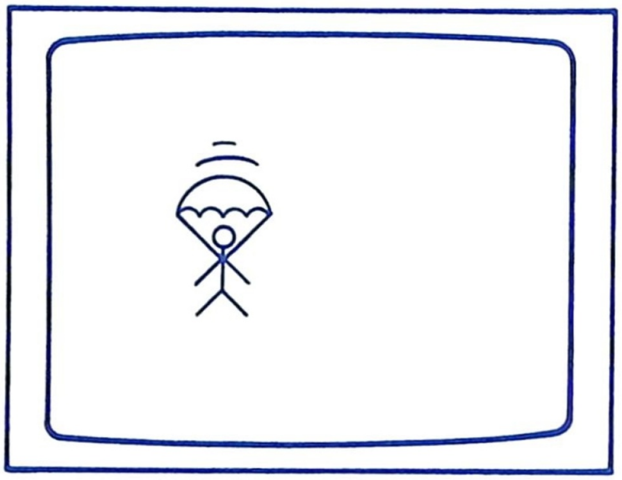
—Line 100 names the program.
—Line 110 activates the high-res mode.
—Line 120 sets the row variable as Row 160 to Row 10 in steps of −10. This makes Jack move up the screen.
—Lines 130 to 170 are the same as the lines in the last program.
—Line 180 gets the next value of y.
—Line 190 states the row variable as Row 10 to Row 160 in steps of 10. This brings Jack back down.
—Lines 200 to 2080 are exactly the same as the same lines in the last program.

When you run this program, you will see a lightning bolt hit our friend Jack and really knock him for a loop.

Jack Saved by IF...THEN

In our last program, we let the little man fall to his death. We can give him a softer landing by using an IF...THEN statement. In this next program, we will tell the computer to give our little friend a parachute when he reaches a certain altitude.

We are really telling the computer to test for a certain condition or value of y, and are telling it what to do when it finds that condition present.



```
100 REM  Saved
110 HGR
120 FOR y = 160 TO 10 STEP -10
130 x = 120
140 HCOLOR = 3: GOSUB 1000
150 HCOLOR = 0: GOSUB 1000
160 HCOLOR = 3: GOSUB 2000
170 HCOLOR = 0: GOSUB 2000
180 NEXT y
190 FOR y = 10 TO 160 STEP 10
195 IF y = 80 GOTO 300
200 HCOLOR = 3: GOSUB 1000
210 HCOLOR = 0: GOSUB 1000
230 HCOLOR = 3: GOSUB 2000
240 HCOLOR = 0: GOSUB 2000
250 NEXT y
300 FOR y = 80 TO 160 STEP 10
310 HCOLOR = 3: GOSUB 3000
330 HCOLOR = 0: GOSUB 3000
340 NEXT y
350 END
1000 REM  Jump
1010 HPLOT x-1,y TO x+1,y
1020 HPLOT x-1,y+1 TO x+1,y+1
1030 HPLOT x,y+1 TO x,y+4
1040 HPLOT TO x+3,y+8
1050 HPLOT x-3,y+7 TO x,y+4
1060 HPLOT x-3,y TO x-1,y+2
1070 HPLOT x+1,y+2 TO x+3,y+1
```

```
1080 RETURN
2000 REM  At Rest
2010 HPLOT x-1,y-1 TO x+1,y-1
2020 HPLOT x-1,y TO x+1,y
2030 HPLOT x,y+1 TO x,y+4
2040 HPLOT x-1,y+5 TO x-1,y+8
2050 HPLOT x+1,y+5 TO x+1,y+8
2060 HPLOT x-1,y+2 TO x-2,y+3
2070 HPLOT x+1,y+2 TO x+2,y+3
2080 RETURN
3000 REM  Chute and Jack
3010 HPLOT x+2,y TO x+5,y
3020 HPLOT x+1,y+1 TO x+6,y
3030 HPLOT x,y+2 TO x+7,y+2
3040 HPLOT x,y+3 TO x+7,y+3
3050 HPLOT TO x+4,y+7
3060 HPLOT x,y+3 TO x+3,y+7
3070 HPLOT x+3,y+8 TO x+4,y+8
3080 HPLOT x+3,y+9 TO x+4,y+9
3090 HPLOT x+2,y+10 TO x+5,y+10
3100 HPLOT x+1,y+11 TO x+6,y+11
3110 HPLOT x+3,y+11 TO x+3,y+12
3120 HPLOT TO x+1,y+15
3130 HPLOT x+4,y+11 TO x+4,y+12
3140 HPLOT TO x+6,y+15
3150 RETURN
```

—Line 100 names the program.
—Line 110 activates the high-resolution mode.
—Line 120 states the row values used to move the man up the screen.
—Line 130 states the column value as 120.
—Lines 140 to 170 display the man moving up the screen.
—Line 180 gets the next value of y.
—Line 190 states the row values for the man moving down the screen.
—Line 195 states that when y equals 80, the program branches to line 300. This is the life-saving line.
—Lines 200 to 240 state that as long as y doesn't equal 80, these lines display the man falling.
—Line 250 sets the next value of y.
—Line 300 states the row values for the parachute drop to the ground.
—Lines 310 to 330 display the man and the chute.
—Line 340 gets the next value of y.

—Line 350 ends the program.
—Lines 1000 to 1080 plot the man jumping.
—Lines 2000 to 2080 plot the man at rest.
—Lines 3000 to 3150 plot the man and his chute.

We will use the IF...THEN and IF...GOTO statements a lot from now on. These statements are an excellent way of branching the program when a certain condition exists.

If you want to experiment with vertical animation in the low-resolution mode, you can. Simply use FOR/NEXT loops to represent row numbers from 39 to 0 (with a negative step) to move up the screen and from 0 to 39 to move down the screen.

Now that we know how to animate vertically and horizontally, we are ready to experiment with some interesting animation programs. All of the programs that we discuss can be changed to take care of practically any animation need you might have. The basic subroutines we are about to show you will be used a great deal in the last three chapters when we begin building games.

FUN WITH ANIMATION

We all know how popular PAC MAN® is. Well, the Adam computer can make a similar "critter" with the following program.

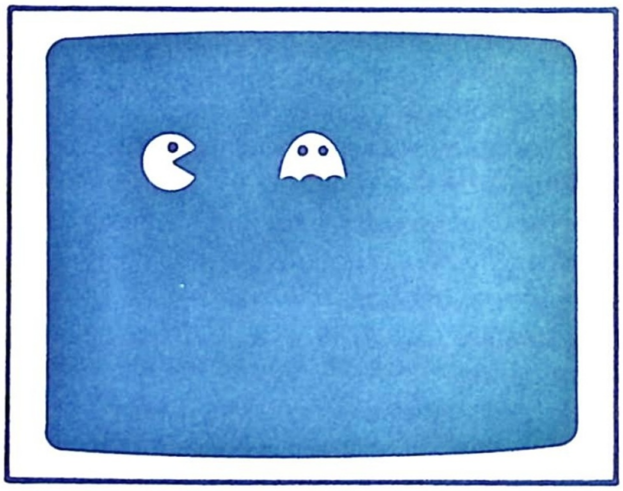


Fig. 6-3. Adam Man.

```
100 REM  Adam Man
110 HGR: y = 120
120 FOR x = 8 TO 208 STEP 8
130 a = x + 24
140 HCOLOR = 15:GOSUB 3000
145 FOR i = 1 TO 100: NEXT i
150 HCOLOR = 0: GOSUB 3000
160 HCOLOR = 3: GOSUB 1000
165 FOR i = 1 TO 100: NEXT i
170 HCOLOR = 4: GOSUB 1000
200 NEXT x
210 END
1000 REM  Mouth Open
1010 HPLOT x,y TO x,y+8
1020 HPLOT TO x+8,y+8
1030 HPLOT TO x+4,y+3
1040 HPLOT TO x+8,y+3
1050 HPLOT TO x+8,y
1060 HPLOT TO x,y
1070 HPLOT x+2,y+2
1080 RETURN
3000 HPLOT a,y+3 TO a,y+8
3010 HPLOT TO a+8,y+8
3020 HPLOT TO a+8,y+3
3030 HPLOT TO a+4,y
3040 HPLOT TO a,y+3
3050 HPLOT a+2,y+4
3060 HPLOT a+4,y+4
3070 RETURN
```

—Line 100 names the program.
—Line 110 activates the high-resolution mode and defines y (the row) as 120.
—Line 120 moves Adam Man across the screen from Column 8 to Column 208 in steps of 8.
—Line 130 defines a as x + 24.
—Line 140 sets the color to magenta and branches to line 3000.
—Line 145 is a timing loop to slow the display.
—Line 150 sets the color to black and branches to line 3000. This will erase the trail of the monster.
—Line 160 sets the color to white and branches to line 1000.
—Line 165 is a timing loop to slow the display.

—Line 170 sets the color to black and branches to line 1000. This erases the trail of Adam Man.
—Line 200 gets the next value of x.
—Line 210 ends the program.
—Lines 1000 to 1070 plot Adam Man with his mouth open.
—Line 1080 returns to either line 165 or line 200.
—Lines 3000 to 3060 plot the monster.
—Line 3070 returns to either line 145 or line 160.

When you run this program, you will see Adam Man chase the monster madly across the screen. The monster has nothing to fear because he will always be 24 positions ahead of Adam Man. However, you can change the program so that the monster can be eaten. Try using an IF...GOTO statement to accomplish this. We will give you the "tools" you need in the next program.

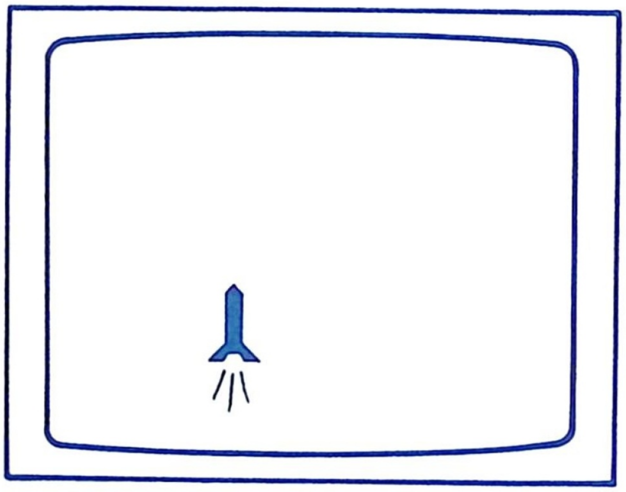


Fig. 6-4. Blast off.

Blast Off

This program uses an IF...GOTO statement to control the outcome of the program. Line 35 tests for a certain condition and will branch to line 400 when that condition exists. Input the program and then we will discuss it.

```
10 REM  Blast Off
20 HGR: x = 100
30 FOR y = 120 TO 0 STEP -10
35 IF y = 20 GOTO 400
40 HCOLOR = 3: GOSUB 120
50 HCOLOR = 0: GOSUB 120
60 NEXT y
70 END
120 HPLOT x+8,y TO x+12,y+4
130 HPLOT x+8,y TO x+4,y+4
140 HPLOT TO x+4,y+19
150 HPLOT TO x,y+23
160 HPLOT TO x+4,y+23
170 HPLOT TO x+6,y+21
180 HPLOT TO x+10,y+21
190 HPLOT TO x+12,y+23
200 HPLOT TO x+16,y+23
210 HPLOT TO x+12,y+19
220 HPLOT TO x+12,y+4
230 RETURN
400 FOR i = 1 TO 15
410 PRINT CHR$(7)
420 PRINT "THE SHIP WAS DESTROYED!"
430 NEXT i
440 GOTO 20
```

—Line 10 names the program.
—Line 20 activates the high-res mode and sets the x variable (the column) as 100.
—Line 30 will plot the rocket up the screen from Row 120 to Row 0 in increments of 10.
—Line 35 says that if y = 20, then the program should branch to line 400.
—Line 40 sets the color to white and branches to line 120 to plot the rocket.
—Line 50 sets the color to black and branches to line 120 to erase the rocket.
—Line 60 gets the next row number.
—Line 70 ends the program.
—Lines 120 to 220 plot the rocket.

—Line 230 returns to line 50 or line 60.
—Line 400 sets a timing loop for the sound and prompt display.
—Line 410 calls the beep sound.
—Line 420 prints the prompt.
—Line 430 gets the next value of i.
—Line 440 loops to line 20 and the display begins again.

When you run this program, you will note that the rocket does just fine until it reaches Row 20. At that point, the rocket disappears, you hear the warning sound, and the prompt appears at the bottom of the screen.

You can set the condition in line 35 to suit your own designs. Also, lines 400 to 430 can be changed to do just about anything you like. Now, let's look at another way to animate.

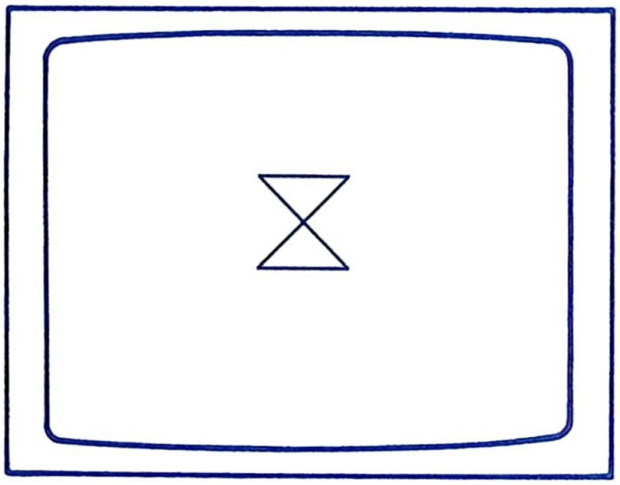


Fig. 6-5. Pinwheel.

Pinwheel

In this program, we will use a different animation routine. Line 110 and line 190 act as the erasers in this program. Let's look at the program in detail.

```
100 REM  Pinwheel
110 HGR: HCOLOR = 3
120 x = 120: y = 80
130 HPLOT x+6,y+8 TO x+13,y
```

```
140 HPLOT TO x+13,y+16
150 HPLOT TO x+6,y+8
160 HPLOT TO x,y
170 HPLOT TO x,y+16
180 HPLOT TO x+6,y+8
185 FOR i = 1 TO 30: NEXT i
190 HGR: HCOLOR = 3
200 HPLOT x,y TO x+13,y
210 HPLOT TO x,y+16
220 HPLOT TO x+13,y+16
230 HPLOT TO x,y
235 FOR i = 1 TO 30: NEXT i
240 GOTO 110
```

—Line 100 names the program.
—Line 110 activates the high-resolution mode and sets the color to white. This line also erases the display plotted in lines 200 to 235 when the loop in line 240 is initiated.
—Line 120 defines the column variable as 120 and the row variable as 80.
—Lines 130 to 185 plot the first part of the Pinwheel.
—Line 190 erases the display plotted in lines 130 to 185 and sets the display color at white for the next part of the Pinwheel.
—Line 240 loops back to line 110.

Using HGR clears the screen in line 190 and then displays the next part of the program. Remember that using HGR in this manner clears the entire screen. So, if you have other graphics on the screen, they will disappear. The following is another example of this type of animation.

Bird

```
100 REM  Bird
110 HGR: HCOLOR = 13
120 x = 104: y = 60
130 HPLOT x,y+6 TO x+3,y
140 HPLOT TO x+18,y+12
150 HPLOT TO x+32,y
160 HPLOT TO x+35,y+6
170 FOR i = 1 TO 30: NEXT i
180 HGR: HCOLOR = 13
```

```
190 HPLOT x,y+28 TO x+18,y+14
200 HPLOT TO x+36,y+28
210 FOR i = 1 TO 30: NEXT i
220 GOTO 110
```

—Line 100 names the program.
—Line 110 activates the high-resolution mode and sets the color to be yellow.
—Line 120 states the column variable as 104 and the row variable as 60.
—Lines 130 to 170 plot the bird's wings in the "up" position.
—Line 180 erases the bird's wings in the "up" position and prepares the next display.
—Lines 190 to 210 display the bird's wings in the "down" position.
—Line 220 loops back to line 110 which erases the "down" position wings and prepares for the next "up" position display.

The bird in this program flaps its wings up and down but it doesn't really go anywhere. In the next program, we change the variables and send our little bird flying.

```
100 REM  Fly
110 HGR: HCOLOR = 13
120 FOR x = 20 TO 250: y = x / 2
130 HPLOT x,y+6 TO x+3,y
140 HPLOT TO x+18,y+12
150 HPLOT TO x+32,y
160 HPLOT TO x+35,y+6
180 HGR: HCOLOR = 13
190 HPLOT x,y+28 TO x+18,y+14
200 HPLOT TO x+36,y+28
210 NEXT x
220 GOTO 110
```

—Line 120 sets our bird in motion. We define the column variable as Column 20 to Column 250. We also define the row variable y as x divided by 2.

This program will make our feathered friend fly on a diagonal from the upper-left corner to the lower-right corner of the screen.

The last two programs in this chapter are somewhat more complicated, but they are based on the principles that we have used all along. We encourage you to use these programs as examples.

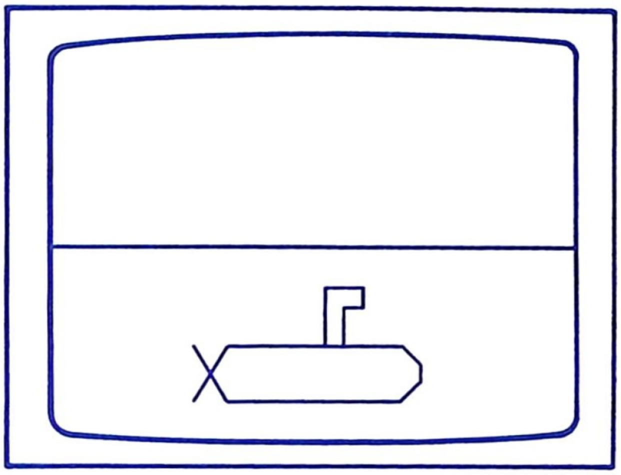


Fig. 6-6. Subshoot.

Subshoot

```
10 REM  Subshoot
20 GOSUB 500
30 FOR x = 10 TO 220 STEP 10:y = 130
35 IF x = 160 GOTO 600
40 HCOLOR = 3: GOSUB 130
50 HCOLOR = 0: GOSUB 130
60 NEXT x
70 END
130 HPLOT x+6,y+14 TO x+27,y+14
140 HPLOT TO x+29,y+13
150 HPLOT TO x+29,y+7
160 HPLOT TO x+27,y+6
170 HPLOT TO x+6,y+6
180 HPLOT TO x,y+12
190 HPLOT TO x,y+7
200 HPLOT x+6,y+14
210 REM  Periscope
220 HPLOT x+18,y+6 TO x+18,y
230 HPLOT TO x+24,y
240 HPLOT TO x+24,y+3
250 HPLOT TO x+22,y+3
260 HPLOT TO x+22,y+6
270 RETURN
500 HGR: HCOLOR = 10
```

```
510 FOR x = 0 TO 250:FOR y = 100 TO 120
520 HPLOT x,y: NEXT y: NEXT x
530 RETURN
600 PRINT CHR$(7)
610 FOR x = 161 TO 230: y = 135
620 HCOLOR = 3: GOSUB 700
630 HCOLOR = 0: GOSUB 700
640 NEXT x
650 FOR x = 161 TO 220 STEP 10
660 y = 130
670 GOTO 40
700 HPLOT x,y TO x+6,y
710 HPLOT x,y+1 TO x+6,y+1
720 HPLOT x,y+2 TO x+6,y+2
730 PRINT CHR$(7)
740 RETURN
```

—Line 10 names the program.
—Line 20 branches to line 500 and plots the water.
—Line 30 states the column variable as Column 10 to Column 220 in increments of 10 and states the row variable as Row 130. This line moves the submarine.
—Line 35 says that when x = 160, the program will branch to line 600 and fire the torpedo.
—Line 40 displays the sub by branching to line 130.
—Line 50 erases the sub by setting the color to black and branching to line 130.
—Line 60 gets the next value of x.
—Line 70 stops the program.
—Lines 130 to 200 plot the submarine.
—Line 210 identifies the following lines as the periscope plotting lines.
—Lines 220 to 260 plot the periscope.
—Line 270 returns to either line 50 or 60.
—Lines 500 to 520 plot the water line.
—Line 530 returns to line 30.
—Lines 600 to 660 set up the routine for plotting and firing the torpedo.
—Line 670 loops to line 40.
—Lines 700 to 730 plot the torpedo.
—Line 740 returns to either line 630 or 640.

When you run this program, you will note that the little submarine makes its way across the screen. When it reaches Column 160, the sub will stop and fire a torpedo. The Adam computer beeps at you as the torpedo is fired. The beep is controlled by line 730. If you like, you can make the sub beep all the way across the screen. Just add a line right after line 260 that is like line 730.

Mars Landing

We think that this is a program that you can really have some fun with. You can add a lot of additional graphics if you like, but we will give you the basics.

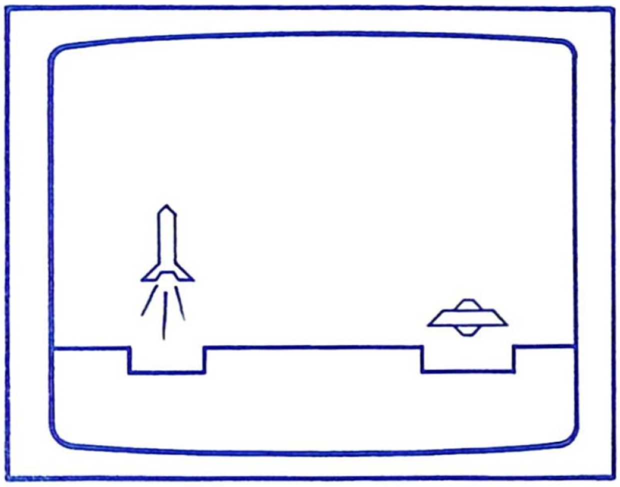


Fig. 6-7. Mars Landing.

Input the following program.

```
100 REM  Mars
110 HGR: GOSUB 700: GOSUB 1000
130 FOR y = 140 TO 20 STEP -16:x = 35
150 HCOLOR = 3: GOSUB 300
160 HCOLOR = 0: GOSUB 300
170 NEXT y
180 FOR y = 10 TO 140 STEP 5
190 x = 140
200 HCOLOR = 3: GOSUB 500
210 HCOLOR = 0: GOSUB 500
220 NEXT y
```

```
230 GOTO 130
300 HPLOT x,y+2 TO x,y+15
310 HPLOT x+1,y TO x+1,y+15
320 HPLOT x+2,y+2 TO x+2,y+15
330 HPLOT x-1,y+12 TO x-1,y+15
340 HPLOT x+3,y+12 TO x+3,y+15
350 RETURN
500 HPLOT x+4,y TO x+5,y
510 HPLOT x+2,y+1 TO x+7,y+1
520 HPLOT x,y+2 TO x+9,y+2
530 HPLOT x+2,y+3 TO x+7,y+3
540 HPLOT x+4,y+4 TO x+5,y+4
550 RETURN
700 HCOLOR = 15
710 FOR x = 40 TO 130: FOR y = 141 TO 159
720 HPLOT x,y: NEXT y: NEXT x
730 FOR x = 0 TO 30: FOR y = 141 TO 159
740 HPLOT x,y: NEXT y:NEXT x
750 FOR x = 155 TO 250: FOR y = 141 TO 159
760 HPLOT x,y: NEXT y: NEXT x
770 RETURN
1000 PRINT "STAND BY FOR THE COUNTDOWN"
1005 FOR i = 1 TO 1000: NEXT i
1010 FOR c = 10 TO 0 STEP -1
1020 PRINT CHR$(7)
1030 PRINT c: FOR i = 1 TO 100: NEXT i
1040 NEXT c
1050 RETURN
```

—Line 100 names the program.
—Line 110 activates the high-res mode and branches to line 700 and then to line 1000.
—Line 130 states the row values as 140 to 20, in increments of 16, and states the column value as 35. This line moves the rocket up the screen.
—Line 150 sets the display color at white and branches to line 300, where the rocket will be displayed.
—Line 160 branches to line 300 and erases the rocket.
—Line 170 gets the next value of y.
—Line 180 states the row value as Row 10 to Row 140 in increments of 5. This line moves the saucer down the screen.
—Line 190 states the column variable as 140.
—Line 200 displays the saucer in white by branching to line 500.
—Line 210 erases the saucer by branching to line 500 and displaying in black.

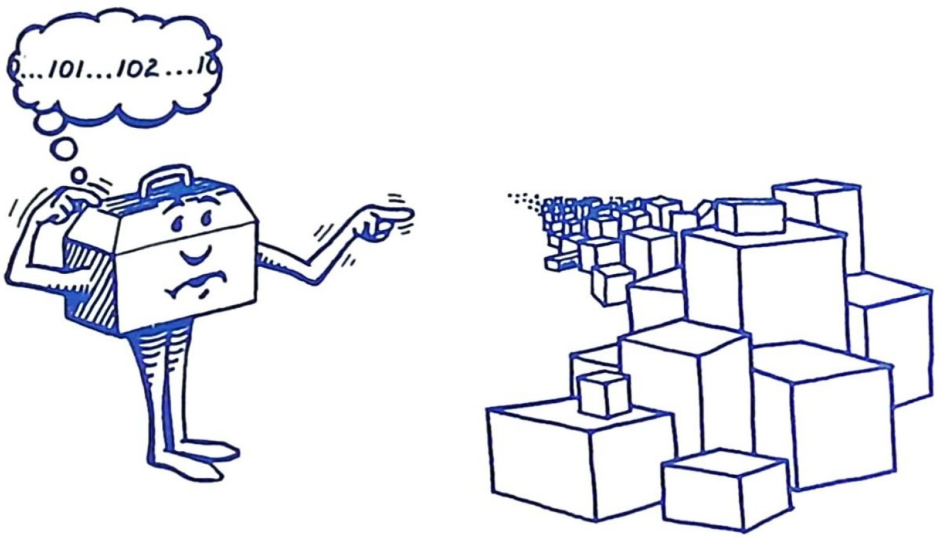
—Line 220 gets the next value of y.
—Line 230 loops back to line 130 and the entire display begins again.
—Lines 300 to 340 plot the rocket moving up the screen.
—Lines 500 to 540 display the saucer moving down the screen.
—Lines 700 to 760 display the surface of the planet. If you want to add more graphic details, this is the best place to do it.
—Lines 1000 to 1040 print the countdown message and sound the countdown beeps.

If you use our design, you can create almost any screen display. An airport scene might be fun or maybe you can create a freeway. Your Adam computer has plenty of memory. We encourage you to experiment with these programs.

Calculating Subroutines: Facts, Figures, and Conversions 7

Your Adam computer can easily become a calculator and can be a helpful problem solver. We think it is smart to use your computer in this way and so you'll be learning how to develop subroutines for fun math games and quizzes in the next chapter.

Don't be anxious about the math because our gentle approach shows you just how easy using arithmetic and algebra is on the Adam. Let's start by reviewing some simple arithmetic operations.



SIMPLE CALCULATIONS

All you have to do to perform simple calculations on the Adam computer is use the PRINT statement. Try this:

```
PRINT 2 + 2          (and press RETURN)

PRINT 5 - 2          (and press RETURN)

PRINT 5 * 2          (The asterisk [*] is the
                     multiplication sign.)

PRINT 6 / 2          (The slash [/] is the division
                     sign.)

PRINT 4 ^ 2          (The ^ is used for exponents.
                     4 ^ 2 is the same as 4².)
```

```
PRINT 1Ø * (4 / 2)      (The parentheses show the
                        computer which operation to
                        perform first.)
```

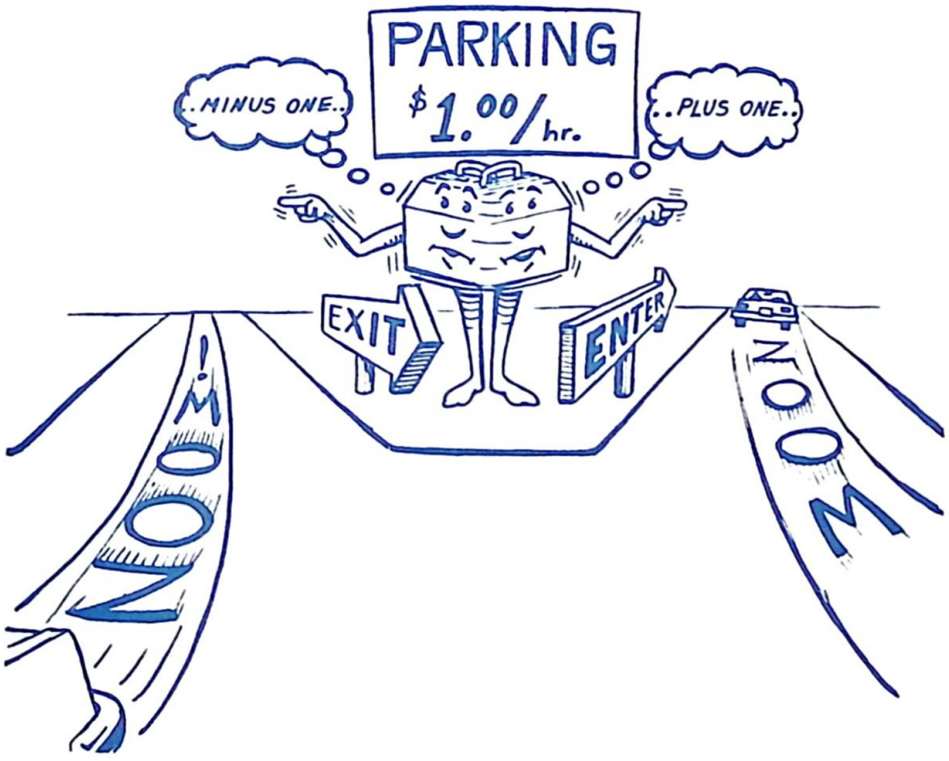
NOTE: Without the parentheses, the computer follows rules of priority. The priorities are:

exponent	∧ (highest priority)
multiplication	*
division	/
addition	+
subtraction	− (lowest priority)

The Adam computer always follows these priorities and always performs mathematical operations from left to right.

Now, let's look at some examples to see how you and the Adam can handle them.

Parking Lot



At your favorite parking lot, there are 20 rows of cars with 15 cars in each row. How many cars are there?

```
PRINT 2Ø * 15
3ØØ
```

There are 300 cars.

In the same parking lot, four cars leave every hour. Two new cars arrive at the same rate, every hour. After three hours, how many cars are in the lot?

Step-by-step, we can work out the problem:

300 cars in the full lot	300
3 hours times 4 cars leaving	− (3 • 4)
3 hours times 2 cars arriving	+ (3 • 2)

Input the following:

```
PRINT 3ØØ - (3 * 4) + (3 * 2)
294
```

There are 294 cars in the lot after three hours.

*PROBLEM 1**

What if 10 cars leave every hour and 6 cars arrive at the same rate? How many would there be in 4 hours?

There are other numeric functions your Adam computer can perform. We encourage you to learn about square roots, logarithms, cosines, and the like using the manual that came with your Adam computer.

CALCULATING SUBROUTINES

We're going to show you how to solve problems by using short programs, or subroutines, on the Adam computer. First, let's look at the following problem.

*NOTE: The answers to all problems are at the end of the chapter.

Gallons and Liters

If one liter is the same as 0.2642 gallon, how many gallons are in 60 liters?

We can solve this problem by multiplying 60 liters times 0.2642 to get the correct number gallons. Using the PRINT statement, this would be

```
PRINT 6Ø * .2642
```

If we want to convert many different amounts of liters into gallons, we can write a program with L = liters and G = gallons.

```
9Ø REM  Gallons
1ØØ HOME
11Ø INPUT l
12Ø g = .2642 * l
13Ø PRINT g
```

—Line 90 names the program.
—Line 100 clears the screen.
—Line 110 allows us to input the number of liters (l) we want to convert.
—Line 120 defines the conversion equation.
—Line 130 prints the value of g calculated in line 120.

Type in this program and see what happens. The ? (question mark) prompts you to key in an amount of liters and the computer then prints the amount of gallons.

Let's make the program more "friendly." Change lines 110 and 130 as we have done here.

```
90 REM  Gallons
100 HOME
110 INPUT " # OF LITERS ";l
120 g = .2642 * l
130 PRINT g;" GALLONS"
```

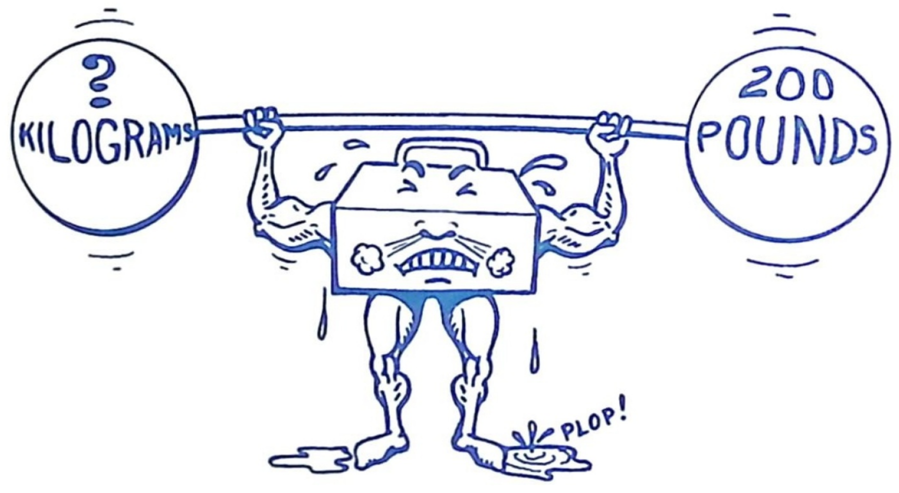
Now the program asks you to input the number of liters and proudly displays for you, in plain English, the liters-to-gallons conversion.

Kilometers and Miles

PROBLEM 2

Write a subroutine for converting from miles to kilometers. For every mile, there are 1.609 kilometers.

Pounds, Ounces, and Kilograms



Let's do a slightly more complex subroutine by adding another input variable.

1 pound = 2.205 kilograms

We want the user to input a number of pounds and ounces and convert this to kilograms. Try the following program out:

```
100 REM  Pounds
110 HOME
120 PRINT "FIRST POUNDS THEN OUNCES"
130 INPUT p
140 INPUT oz
150 pz = p + oz / 16
160 kg = pz / 2.205
170 PRINT pz; " POUNDS ARE"
180 PRINT kg; " KILOGRAMS"
```

—Line 100 names the program.
—Line 110 clears the screen.
—Line 120 prints the opening prompt.
—Line 130 allows us to input a number for pounds.
—Line 140 allows us to input a number for ounces.
—Line 150 defines the equation for changing ounces into pounds.
—Line 160 defines the equation for changing pounds into kilograms.
—Line 170 prints the number of pounds computed in line 150 and "POUNDS ARE," the words in quotes.
—Line 180 prints the number of kilograms computed in line 160 and the kilogram prompt.

Inches and Centimeters

We know that 1 inch equals 2.54 centimeters. Now, let's write a simple subroutine that we can use to solve for centimeters.

```
90 REM  Inches
100 HOME
110 INPUT "HOW MANY INCHES ? ";i
120 cm = 2.54 * i
130 PRINT i;  "INCHES IS APPROXIMATELY"
140 PRINT cm; " CENTIMETERS."
```

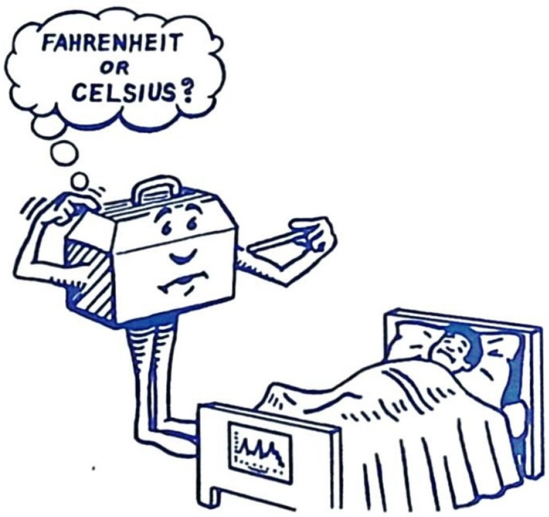
—Line 90 names the program.
—Line 100 clears the screen.
—Line 110 prints the prompt and allows us to enter the number of inches that are to be converted.
—Line 120 defines the conversion equation.

—Lines 130 and 140 print the number of inches, the words contained in quotes, and the equivalent number of centimeters.

PROBLEM 3

Now, using what you learned in the pounds/kilograms subroutine, can you change this into a feet, inches, and centimeters conversion subroutine?

Temperature



Converting from Fahrenheit to Celsius is not easy, but the following subroutine should help.

```
9Ø REM  Degrees
1ØØ HOME
11Ø INPUT "DEGREES FAHRENHEIT ";f
12Ø c = .55556 * (f - 32)
13Ø PRINT f; " DEGREES FAHRENHEIT IS"
14Ø PRINT c; " DEGREES CELSIUS"
```

—Line 90 names the program.
—Line 100 clears the screen.
—Line 110 prints the message and allows us to input degrees.
—Line 120 defines c, or Celsius, using the conversion equation.

—Lines 130 and 140 print the number inputted, the word contained in quotes, and the equivalent degrees Celsius.

Dollars and Deutschmarks

Have you ever looked at the foreign rates of exchange in the financial section of the newspaper? The following subroutine will show you some dollar amounts and their equivalent in Marks. In 1983, one dollar was approximately 2.55 Marks in West Germany.

```
90 REM  Dollars
100 HOME
110 PRINT "DOLLARS", "MARKS"
120 d = 0
130 PRINT d, 2.55 * d
140 d = d + 5
150 IF d < 30 GOTO 130
160 END
```

—Line 90 names the program.
—Line 100 clears the screen.
—Line 110 prints the word Dollars and then prints the word Marks halfway across the screen (because of the comma).
—Line 120 is the first value of d.
—Line 130 prints the value of d and then prints 2.55 times d at a point halfway across the screen.
—Line 140 is the next value of d.
—Line 150 tells the computer to keep figuring the Mark equivalent as long as d is less than 30.
—Line 160 stops the program.

Input the program and RUN it. Do you like the table format?

What happens when you change line 150 to IF d < 50 THEN 130? How about changing the increment in line 140 from d + 5 to d + 2? Experiment with this table format.

PROBLEM 4

You realize that exchange rates change almost daily. Change the *Dollars and Deutschmarks* subroutine so that you can input the most current exchange rate.

PROBLEM 5

Wouldn't it be interesting to display Fahrenheit and Celsius conversions in a table format? Go back to the *Temperature* subroutine and prepare a new program.

Investing

How much money can you make by investing your money? This program shows you how easy it is to make money when you have some. The standard formula for a "return" is:

$$R = M \cdot (1 + I/100)^Y$$

where,

R = the return,
M = money invested,
I = interest rate per period,
Y = the number of periods.

```
90 REM  Invest
100 HOME
110 PRINT "AMOUNT OF MONEY TO INVEST"
120 INPUT m
130 PRINT "WHAT IS THE RATE PER YEAR"
140 INPUT i
150 PRINT "HOW MANY YEARS"
160 INPUT y
170 r = m*(1+i/100)^y
180 PRINT "YOUR RETURN ON ";m;" INVESTED IS ";r
```

None of the lines used in this subroutine are new. This is a fun subroutine to play with.

Profit

Most people are in business to make money, and making money means profits. How do we figure profit?

Profit = Total Revenues − Total Costs

Now let's do a "profitable" subroutine. Again, there is nothing new in this subroutine.

```
90 REM  Profit
100 HOME
110 PRINT "WHAT ARE YOUR TOTAL"
120 PRINT
130 PRINT "REVENUES? "
140 INPUT r
150 PRINT "WHAT ARE YOUR TOTAL COSTS? "
160 INPUT c
170 p = r - c
180 PRINT p; " IS YOUR TOTAL PROFIT"
```

Now we should figure at what "rate" we are profitable. When comparing businesses, we look at the profit as a percentage of revenue. By using this ratio, the performances of businesses can be compared regardless of their size. We have to add lines 190 to 200 to get this ratio.

```
90 REM  Profit
100 HOME
110 PRINT "WHAT ARE YOUR TOTAL "
120 PRINT
130 PRINT "REVENUES?"
140 INPUT r
150 PRINT "WHAT ARE YOUR TOTAL COSTS?"
160 INPUT c
170 p = r - c
180 PRINT p; " IS YOUR TOTAL PROFIT"
190 pp = p / r * 100
200 PRINT pp; " IS YOUR PROFIT PERCENTAGE"
```

—Line 190 computes the percentage of profit.
—Line 200 prints the profit percentage message.

Current Ratio

Banks and financial analysts often look at your current ratio as a measure of your "financial health." It boils down to how much you have and what you owe. The simple formula is:

Current Ratio = Current Assets/Current Liabilities

PROBLEM 6

Write a subroutine for figuring the current ratio.

Loan Analysis

Using what we know about Current Ratio from the last subroutine, let's figure a way to gauge a person's (or a business') credit worthiness in a loan situation.

```
90 REM  Loan Analysis
100 HOME
110 PRINT "WHAT ARE YOUR CURRENT ASSETS?"
120 INPUT a
130 PRINT "WHAT IS YOUR CURRENT LIABILITY?"
140 INPUT l
150 r = a / l
```

```
160 IF r > 1 GOTO 180
170 IF r <= 1 GOTO 220
180 PRINT "YOUR CURRENT RATIO IS ";r
190 PRINT
200 PRINT "YOUR LOAN IS ACCEPTED"
210 END
220 PRINT "YOUR CURRENT RATIO IS ";r
230 PRINT
240 PRINT "YOUR LOAN IS DENIED"
250 END
```

Some banks won't loan you money if your liabilities exceed your assets (or your current ratio is less than 1). Let's assume that banks will only grant loans if the current ratio is greater than 1.

This subroutine is the same as the others we have worked with, except we have used IF/THEN statements to branch the programs, based on certain conditions. These are called conditional branches.

—Line 160 tells the computer to branch to line 180 if r is greater than 1.
—Line 170 tells the computer to branch to line 220 if r is less than or equal to 1.

ANSWERS TO CHAPTER PROBLEMS

Problem 1

Parking Lot

```
PRINT 300 - (4 * 10) + (4 * 6)
284
```

Problem 2

Kilometers and Miles

```
90 HOME
100 INPUT "MILES";m
110 km = 1.609 * m
120 PRINT m; " MILES EQUAL ";km; " KILOMETERS"
```

Problem 3

Feet, Inches, and Centimeters

```
90 HOME
100 PRINT "FEET FIRST THEN INCHES"
110 INPUT f: INPUT i
120 fi = i + (f * 12)
130 cm = 2.54 * fi
140 PRINT fi; " INCHES IS APPROXIMATELY"
150 PRINT cm; " CENTIMETERS"
```

Problem 4

Exchange Rate

```
90 HOME
100 INPUT "EXCHANGE RATE ";ex
110 PRINT "DOLLARS","MARKS"
120 d = 0
130 PRINT d, ex * d
140 d = d + 5
150 IF d<30 GOTO 130
160 END
```

Problem 5

Temperature Table

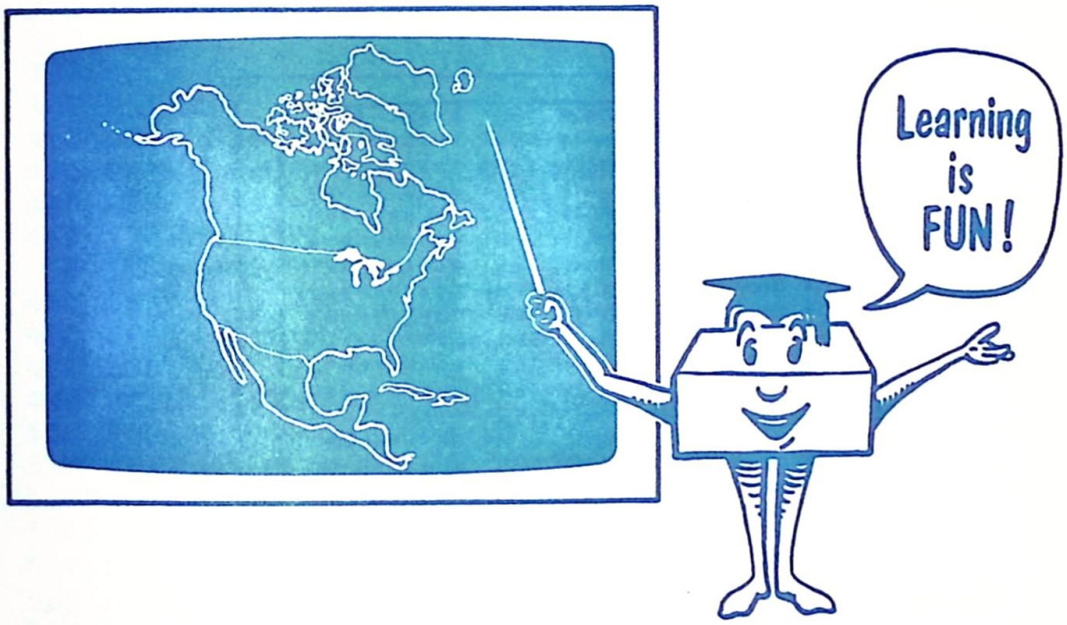
```
90 HOME
100 PRINT "FAHRENHEIT","CELSIUS"
110 f = 0
120 PRINT f, .555556*(f-32)
130 f = f + 20
140 IF f <= 220 GOTO 120
150 END
```

Problem 6

Current Ratio

```
90 HOME
100 PRINT "WHAT ARE YOUR ASSETS?"
110 INPUT a
120 PRINT "WHAT IS YOUR CURRENT LIABILITY"
130 INPUT l
140 r = a / l
150 PRINT r; " IS YOUR CURRENT RATIO"
```

We have now covered all of the basic tools. Armed with the knowledge of how the computer works and a variety of subroutines, we are now ready to tackle some real programming. The next three chapters contain longer programs. Let's begin by creating some educational games.

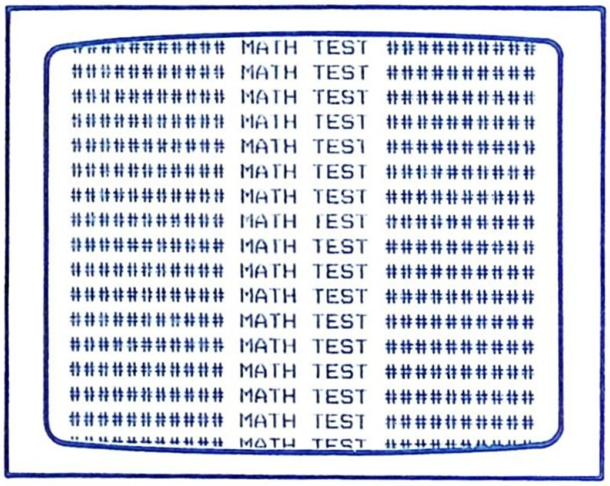


Educational Games 8

The educational games in this chapter are designed so that you can have fun while learning. We explain the structure of each program and encourage you to improve and enhance them with graphics, color, and sound, or wherever you like. The logic for these games can be used to create many more games, and we will give you lots of ideas for those, too.

One of the main reasons we use a computer and write our own programs is to teach our children math and general educational concepts. Our kids enjoy quizzing each other and us, their parents, about all kinds of things—from states and capitals questions to riddles. We hope you have as good a time with these games as we do.

MATH TEST



```
########### MATH TEST ##########
########### MATH TEST ##########
########### MATH TEST ##########
########### MATH TEST ##########
########### MATH TEST ##########
########### MATH TEST ##########
########### MATH TEST ##########
########### MATH TEST ##########
########### MATH TEST ##########
########### MATH TEST ##########
########### MATH TEST ##########
########### MATH TEST ##########
########### MATH TEST ##########
########### MATH TEST ##########
########### MATH TEST ##########
########### MATH TEST ##########
########### MATH TEST ##########
```

This program asks you to add numbers. It stops the game after 10 questions, gives you the score, and ranks you on three levels of competence. Type it in and try it out.

```
100 REM  Math Test
110 FOR x = 0 TO 21
120 PRINT "********** MATH TEST **********"
130 NEXT x
180 INVERSE
190 PRINT "ADD THESE NUMBERS"
200 q = 0: nc = 0
210 n = RND(-n)
230 a = INT(1+10*RND(1))
```

```
240 b = INT(1+10*RND(1))
250 q = q + 1
260 IF q = 10 GOTO 430
270 HOME
280 PRINT "HOW MUCH IS ";a;" PLUS ";b
290 INPUT x
300 IF x = a + b GOTO 360
310 PRINT "WRONG"
320 PRINT "THE ANSWER IS ";(a+b)
330 FOR i = 1 TO 1000
340 NEXT i
350 GOTO 400
360 PRINT "CORRECT"
370 FOR i = 1 TO 1000
380 NEXT i
390 nc = nc + 1
400 PRINT
410 GOTO 230
430 PRINT nc;" QUESTIONS RIGHT OUT OF ";q
435 FOR i = 1 TO 1000: NEXT i
440 j = nc/q
450 IF j = 1 GOTO 490
460 IF (j<1) AND (j>=.7) GOTO 530
470 IF j<.7 GOTO 570
480 END
490 PRINT "VERY GOOD"
500 FOR i = 1 TO 1000
510 NEXT i
520 GOTO 480
530 PRINT "GOOD, PRACTICE A LITTLE"
540 FOR i = 1 TO 1000
550 NEXT i
560 GOTO 480
570 PRINT "YOU NEED A LOT OF PRACTICE"
580 FOR i = 1 TO 1000
590 NEXT i
600 GOTO 480
```

—Line 100 names the program.
—Lines 100 to 130 display and hold the MATH TEST display on the screen.
—Line 180 activates the inverse function.
—Line 190 prints the text words in quotation marks.

—Line 200 sets the initial value of q (the number of questions) as 0 and the initial value of nc (number correct) as 0.
—Lines 210 to 240 are the random number generator. It selects the numbers from 1 to 10 at random.
—Line 250 keeps track of the number of questions asked.
—Line 260 ends the games when 10 questions have been asked.
—Line 270 clears the screen.
—Line 280 prints the number-facts sentence. The two random numbers selected in lines 230 and 240 are printed in place of a and b.
—Line 290 allows us to input an answer.
—Line 300 states that if the answer inputted (x) equals a plus b, the program branches to line 360.
—Line 310 states that if the answer inputted (x) doesn't equal a plus b, the wrong prompt is printed.
—Line 320 prints the correct answer.
—Lines 330 and 340 hold the wrong answer prompts on the screen.
—Line 350 loops to line 400.
—Line 360 prints the word CORRECT.
—Lines 370 and 380 hold the CORRECT prompt on the screen.
—Line 390 keeps track of the number of correct answers.
—Line 400 prints a blank line.
—Line 410 loops to line 230 and another set of numbers is selected.
—Line 430 prints your score.
—Line 440 computes the percentage of correct answers.
—Line 450 states that if your score is 1 or 100%, the program branches to line 490.
—Line 460 states that if your score is 90% to 70%, the program branches to line 530.
—Line 470 states that if the score is below 70%, the program branches to line 570.
—Line 480 exits the program.
—Lines 490 to 510 print and hold the prompt for 100% correct.
—Line 520 loops to line 480 and exits the program.
—Lines 530 to 550 print the prompt for 90% to 70% correct.
—Line 560 loops to line 480.
—Lines 570 to 590 print the prompt for the less than 70% condition.
—Line 600 loops to line 480.

Suggested Program Changes and "Dressing"

You can enhance or dress up the program by adding sound, color, and graphics. Look at your "tools" chapters for ideas. In this program, there are several natural places for program enhancement:

1. When a correct answer is entered.
2. When an incorrect answer is entered.
3. When the score for the round is displayed using lines 490, 530, and 570.

The enhancement can be as simple as a different screen and graphic combination, or as complex as a smiling face with changing background colors and music signalling a job well done.

The program can be easily altered to suit your needs in many ways:

1. *LEVEL*—Change the level of difficulty by changing lines 230 and 240. To make the quiz harder, change the statement to read:

```
230 a = INT(1+20*RND(1))
240 b = INT(1+20*RND(1))
```

   This will give you addition problems with numbers from 1 to 20.

2. *LENGTH*—You can make the quiz longer by changing the number of questions asked before the program ends. Change line 260 to:

```
260 IF q = 20 GOTO 430
```

   This will give you a quiz with twenty questions. You can change the number 20 to any number to get a quiz of any length.

3. *GRADING*—You can set your own grades or competency levels by changing lines 450 to 470. If you would rather have "VERY GOOD WORK" mean 0.9 (or 90%) and higher, change the lines to read:

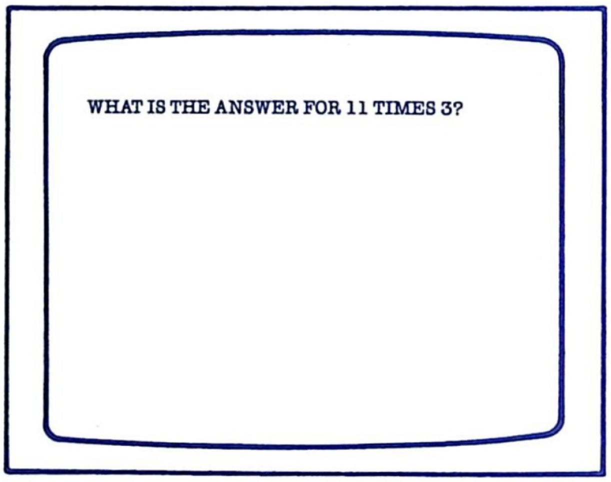
```
450 IF j >= .9 GOTO 490
460 IF (j<.9) AND (j>=.7) GOTO 530
470 IF j<.7 GOTO 570
```

4. *OPERATION*—You can change this game from addition to any other arithmetic operation you want. For subtraction, change lines 300 and 320 to:

```
300 IF x = a - b GOTO 360
320 PRINT (a-b);" IS THE CORRECT ANSWER."
```

You will also need to change the screen prompt to read minus (−) rather than plus (+).

MULTIPLICATION TABLES



WHAT IS THE ANSWER FOR 11 TIMES 3?

Using the same program structure as "MATH TEST," along with READ and DATA statements, you can quiz your knowledge of the multiplication tables. The number of questions asked is the same as the amount of data. When your answers are all correct, the computer smiles for you.

```
90 REM  Math Test
100 HOME
110 PRINT "THIS PROGRAM WILL HELP"
120 PRINT
130 PRINT "YOU LEARN YOUR"
140 PRINT
150 PRINT "MULTIPLICATION TABLES."
160 PRINT
```

```
170 PRINT "HAVE FUN"
180 FOR i = 1 TO 1500: NEXT i
190 q = 0: n = 0
200 READ a
210 IF a = 99 GOTO 420
220 READ b
230 q = q + 1
240 HOME
250 PRINT "WHAT IS ";a;" TIMES ";b
260 PRINT
270 INPUT x
280 PRINT
290 IF x = a*b GOTO 350
300 PRINT "WRONG"
310 PRINT "THE CORRECT ANSWER IS ";(a*b)
320 FOR i = 1 TO 1000
330 NEXT i
340 GOTO 390
350 PRINT "CORRECT"
360 FOR i = 1 TO 1000
370 NEXT i
380 n = n + 1
390 PRINT
400 GOTO 200
420 HOME: PRINT n;" QUESTIONS RIGHT OUT OF ";q
425 FOR i = 1 TO 1500: NEXT i
430 j = n/q
440 IF j = 1 GOTO 490
450 IF (j<>1 AND j>=.6) GOTO 640
460 IF j<.6 GOTO 680
470 DATA 11,3,12,6,15,9,14,4,10,11,13,12,15,16,
    11,12,13,5,7,14,99
480 END
490 HOME
495 PRINT "WAY TO GO": FOR i = 1 TO 1000: NEXT i
500 GR: COLOR = 15: HLIN 10,30 AT 5
510 HLIN 10,30 AT 25: VLIN 5,25 AT 10: VLIN 5,25
    AT 30
520 COLOR = 7: FOR y = 13 TO 14: FOR x = 8 TO 9
530 PLOT y,x: NEXT x: NEXT y: FOR y = 26 TO 27:
    FOR x  = 8 TO 9
540 PLOT y,x: NEXT x: NEXT y: FOR y = 19 TO 21:
    FOR x  = 12 TO 13
550 PLOT y,x: NEXT x: NEXT y: FOR y = 13 TO 27:
    FOR x  = 19 TO 20
```

```
560 PLOT y,x: NEXT x: NEXT y: FOR y = 13 TO 14:
    FOR x  = 17 TO 18
570 PLOT y,x: NEXT x: NEXT y: FOR y = 26 TO 27:
    FOR x  = 17 TO 18
580 PLOT y,x: NEXT x: NEXT y
590 FOR i = 1 TO 1500: NEXT i
600 TEXT
630 GOTO 480
640 PRINT "GOOD WORK"
650 FOR i = 1 TO 1000
660 NEXT i
670 GOTO 480
680 PRINT "NOT SO HOT, TRY AGAIN"
690 FOR i = 1 TO 1000
700 NEXT i
710 GOTO 480
```

—Line 90 names the program.
—Line 100 clears the screen.
—Lines 110 to 180 print and hold the beginning message.
—Line 190 sets the initial value of q (number of questions) and n (number of correct answers).
—Line 200 tells the computer to read "a" from the DATA statement.
—Line 210 states that if a = 99, then the program branches to line 420.
—Line 220 tells the computer to read "b" from the DATA statement.
—Line 230 keeps track of the number of questions asked.
—Line 240 clears the screen.
—Line 250 prints the text question.
—Line 260 prints a blank line.
—Line 270 allows us to input our answer.
—Line 280 prints a blank line.
—Line 290 states that if our answer equals a times b, the program branches to line 350.
—Line 300 prints the wrong answer prompt.
—Line 310 prints the correct answer prompt.
—Lines 320 and 330 hold messages printed in lines 300 and 310 on the screen.
—Line 340 loops to line 390.
—Line 350 prints the "CORRECT" message.
—Lines 360 and 370 hold the message given in line 350 on the screen.

—Line 380 keeps track of the number of correct answers.
—Line 390 prints a blank line.
—Line 400 loops to line 200.
—Line 420 prints the number of questions that are correct out of the total number of questions asked.
—Lines 430 to 460 evaluate the score and branch to the appropriate line.
—Line 470 contains the data for the multiplication problems.
—Line 480 stops the program.
—Line 490 clears the screen.
—Line 495 prints and holds the "WAY TO GO" prompt.
—Lines 500 to 590 draw and hold the smiling face on the screen.
—Line 600 sends the computer back to the text mode.
—Lines 640 to 660 print the good score message.
—Line 670 loops to line 480.
—Lines 680 to 700 print and hold the poor score message.
—Line 710 loops to line 480.

Suggested Program Changes and "Dressing"

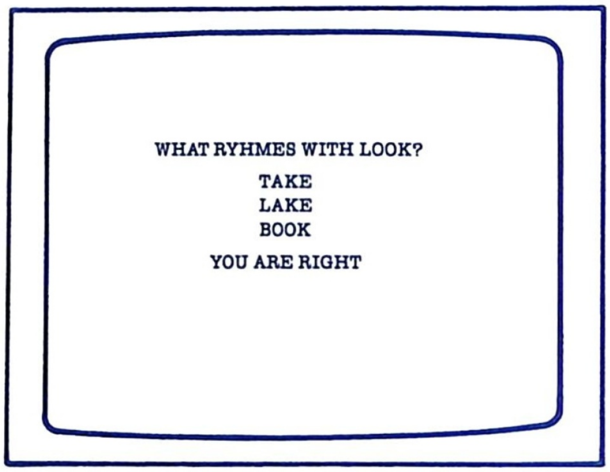
Add sound, color, and more graphics to the scoring prompts. We have used a smiling face to encourage you to be creative. Try adding a frowning face after line 680. Use the Random Color subroutine from Chapter 3 to reinforce a good score. Or, add a little sound. You can change the program to suit yourself.

Increase or decrease the level of difficulty of the problems by changing the numbers in the DATA statement. Or, add more numbers into the DATA statement to make the quiz last as long as you like. Or, try changing the scoring. Maybe you can develop your own point system. Change the operation to addition, division, or subtraction.

The READ and DATA statements are pretty powerful statements and we will use them more with our *States and Capitals* quiz later on.

RHYMING

Here is another program that uses DATA statements. This program asks you to pick out words that rhyme.


WHAT RYHMES WITH LOOK?
TAKE
LAKE
BOOK
YOU ARE RIGHT

```
5 REM  Rhyme
5Ø HOME: GOSUB 2ØØØ
1ØØ q = Ø: n = Ø
122 READ a$
124 IF a$ = "STOP" THEN END
125 READ b$: READ c$: READ d$: READ e$
13Ø q = q + 1
14Ø PRINT "WHAT RHYMES WITH ";a$: PRINT
15Ø PRINT b$: PRINT
16Ø PRINT c$: PRINT
17Ø PRINT d$: PRINT
18Ø PRINT: PRINT: PRINT
19Ø INPUT x$
2ØØ IF x$ = e$ THEN GOSUB 5ØØ
21Ø IF x$ <> e$ THEN GOSUB 6ØØ
24Ø GOTO 122
5ØØ REM  Correct Answer
51Ø HTAB 1Ø: PRINT "YOU ARE RIGHT"
52Ø FOR i = 1 TO 3: PRINT CHR$(7): NEXT i
53Ø FOR i = 1 TO 5ØØ: NEXT i
54Ø n = n + 1
55Ø PRINT n;" RIGHT OUT OF ";q: FOR i = 1 TO
    1ØØØ:  NEXT i
56Ø HOME: RETURN
6ØØ REM  Wrong Answer
61Ø HTAB 1Ø: PRINT "YOU ARE WRONG"
62Ø PRINT CHR$(7)
63Ø FOR i = 1 TO 5ØØ: NEXT i
```

```
635 HOME
640 RETURN
1000 DATA  "CAT","PUT","SAT","SIT","SAT"
1010 DATA  "LOVE","LEAVE","DOVE","LIVE","DOVE"
1020 DATA  "LOOK","TAKE","LAKE","BOOK","BOOK"
1030 DATA  "WALK","WAKE","TALK","TAKE","TALK"
1040 DATA  "MAN","CAN","MINE","AND","CAN"
1050 DATA  "END","SEND","AND","BIN","SEND"
1060 DATA  "STOP"
2000 PRINT "THIS IS A RHYMING GAME": PRINT
2010 PRINT "THE COMPUTER WILL ASK YOU TO": PRINT
2020 PRINT "PICK THE WORDS THAT RHYME.": PRINT
2030 PRINT "JUST TYPE IN THE RIGHT ANSWER.":PRINT
2040 PRINT "PRESS THE LOCK KEY FIRST": PRINT
2050 FOR i = 1 TO 2000: NEXT i
2060 HTAB 12: VTAB 12: PRINT "GOOD LUCK"
2070 FOR i = 1 TO 1000: NEXT i
2080 FOR i = 1 TO 10: PRINT CHR$(7)
2090 NEXT i: HOME
2100 RETURN
```

—Line 5 names the program.
—Line 50 branches to line 2000.
—Line 100 sets the initial value of questions asked (q) and number correct (n).
—Line 122 tells the computer to read a$ from the DATA statement.
—Line 124 says that if we are done, exit.
—Line 125 reads b$, c$, d$, and e$ from the DATA statement.
—Line 130 keeps track of the number of questions asked.
—Line 140 prints the rhyming questions.
—Lines 150 to 180 print the possible rhyming words.
—Line 200 says that if your input is the same as e$, then the program branches to line 500.
—Line 210 says that if your input is not equal to e$, then the program branches to line 600.
—Line 240 loops to line 122.
—Lines 500 to 560 display the correct answer prompt and play the correct answer beep.
—Lines 600 to 640 display the wrong answer prompt and play the wrong answer beep.
—Lines 1000 to 1060 contain the rhyming word data.
—Lines 2000 to 2100 display the beginning screen prompts.

This is a good game for little children; it helps with reading and phonetic skills. You can fill the DATA statements with any words you like. As with the other games in this chapter, we urge you to add your own embellishments. This game asks you to match 6 sets of rhyming words. If you want more questions, add additional data statements. Let's try another game.

RIDDLE FRACTIONS

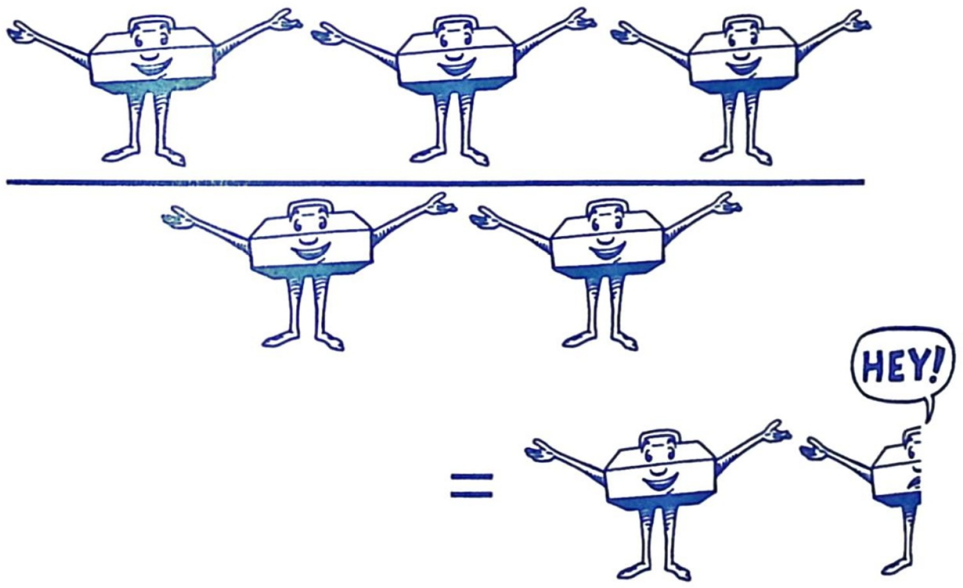
This riddle game is fun and tests your ability to understand fractions. The computer asks you to figure out clues to the riddle using your knowledge of fractions. Then you have to use your imagination to answer the riddle! For example:

MIDDLE 1/2 OF MATE = AT
FIRST 1/2 OF RUNNER = RUN

If you cannot answer the riddle, key in anything and the program will give you a hint. You must answer the question correctly with the hint. The computer will ask *forever,* using the hint, until you finally get the riddle—just like your friends do when they tell you a riddle! Try it!

```
100 REM  Riddle
110 READ a$
120 IF a$ = "END" GOTO 350
130 READ b$, c$, d$, e$, f$, g$, h$, i$, j$, k$
140 READ l$, m$, n$
150 HOME: PRINT "SOLVE THIS RIDDLE"
160 PRINT
170 PRINT a$: PRINT b$: PRINT c$: PRINT d$: PRINT
    e$:  PRINT f$: PRINT g$: PRINT h$: PRINT i$:
    PRINT j$
180 INPUT x$
190 IF x$ = k$ GOTO 260
200 PRINT "HERE IS A HINT"
210 PRINT
220 PRINT l$: PRINT m$: PRINT n$
230 INPUT y$
240 IF y$ = k$ GOTO 260
250 GOTO 200
260 PRINT "YOU ARE RIGHT": PRINT: PRINT: PRINT
    "THE  ANSWER WAS ": PRINT: PRINT k$
```

```
270 FOR i = 1 TO 1000
280 NEXT i
290 GOTO 350
300 DATA  "FIRST 1/2 OF WHATEVER","MIDDLE 3/5 OF
    CHASE","FIRST 1/2 OF FIVEFOLD"
310 DATA  "LAST 1/2 OF BUCKEYES","MIDDLE 3/5 OF
    HANDY","LAST 5/7 OF ARRESTS"
320 DATA  "LAST 1/3 OF BEACON","MIDDLE 1/9 OF
    CRABAPPLE","FIRST 5/9 OF WATERFALL"
330 DATA  "LAST 3/7 OF GRABBED","THE MISSISSIPPI
    RIVER","MIDDLE 3/5 OF OTHER"
340 DATA  "FIRST 11/13 OF MISSISSIPPIAN","LAST
    5/6 OF  DRIVER"
350 END
```



—Line 100 names the program.
—Line 110 reads a$ from the DATA statement.
—Line 120 states that when all the DATA has been read, the program will branch to line 350.
—Line 130 reads b$ to k$ from the DATA statement.
—Line 140 reads l$, m$, and n$ from the DATA statement.
—Line 150 prints the "SOLVE THIS RIDDLE" prompt.
—Line 160 prints a blank row.
—Line 170 prints the clues a$ to j$.

—Line 180 allows you the opportunity to input the answer of your choice.
—Line 190 says that if your answer equals k$, the program is to branch to line 260.
—Lines 200 to 220 display the "hint" prompts.
—Line 230 allows you to input your second guess.
—Line 240 says that if y$ or your second guess = k$, then the program branches to line 260.
—Line 250 loops to line 200.
—Line 260 prints the "right answer" prompt.
—Lines 270 to 280 hold the "right answer" prompt on the screen.
—Line 290 loops to line 350.
—Lines 300 to 340 contain the riddle data.
—Line 350 stops the program.

We recommend using your own riddles in the DATA statements. Turning the riddles into the "fraction code" is half the fun. The other half is trying them out on your friends. The following riddles are corny, but maybe they will bring to mind some of your favorites.

What kind of bridge makes people most anxious?
ANSWER: A suspension bridge

What has teeth but cannot bite or chew?
ANSWER: A comb

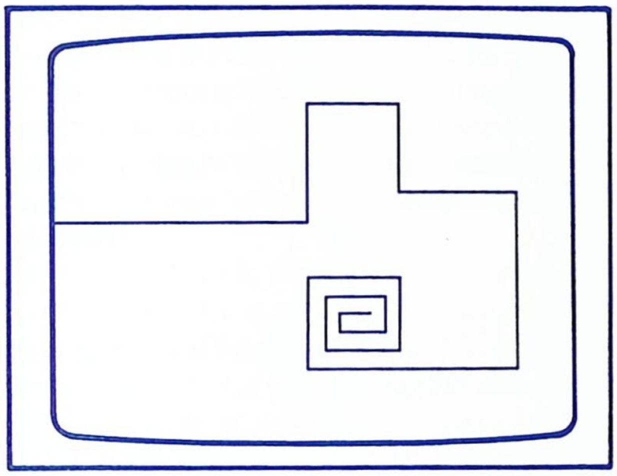
What does a pet canary say on Halloween?
ANSWER: Twick or Tweet

What do you get when you cross a centipede with a parrot?
ANSWER: A walkie-talkie

Remember that, in DATA statements, you can fill only four lines before you must add another line number.

COLORED PENS

This program received more attention from the kids in our neighborhood than any other in the book. It is called COLORED PENS. There is no score to keep—just plain artistic fun. By inputting different color codes, you can



draw in all sixteen colors. The direction keys are listed here so that you can get familiar with them.

Color and Direction Codes

When the program runs, you will be asked to select a color. Once you select a color by pressing one of the color code numbers (any number from 0 to 15), the Adam will draw in that color until you change it.

The direction of the "Color Pen" is controlled by the t, f, g, and v keys.

Pressing t will draw upward.
Pressing v will draw downward.
Pressing f will draw to the left.
Pressing g will draw to the right.

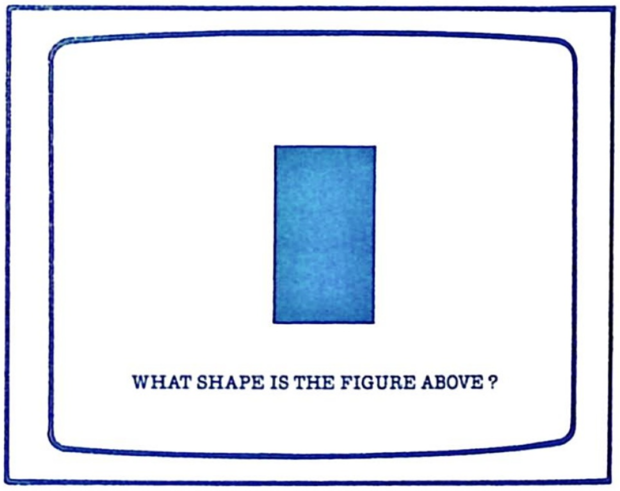
If you want to change color while you are drawing, press the c key. This will give the color code prompt. Input the color you want to change to and continue to draw.

Pressing the q key will erase the screen and you can begin over again. If you want to stop drawing, just press the p key. The keys are defined using their ASCII numbers. We have used the lowercase code numbers for each letter. Enter the following program and have some fun.

```
50 REM  Colorpens
100 HOME
110 GOSUB 400
120 x = 20: y = 20
130 GR
140 INPUT "ENTER THE COLOR CODE ";c
142 IF c<0 OR c>15 GOTO 140
145 COLOR = c
150 PLOT x,y
160 GET a$
170 IF a$ = CHR$(113) THEN 120
180 IF a$ = CHR$(112) THEN END
190 IF a$ = CHR$(99) GOTO 140
200 IF a$ = CHR$(116) THEN y = y - 1
210 IF a$ = CHR$(118) THEN y = y + 1
220 IF a$ = CHR$(102) THEN x = x - 1
230 IF a$ = CHR$(103) THEN x = x + 1
240 IF x>=38 THEN x = 37
250 IF x<=1 THEN x = 1
260 IF y >=38 THEN y = 37
270 IF y<=1 THEN y = 2
280 GOTO 150
400 HTAB 10: VTAB 10: PRINT "LEARN TO DRAW":PRINT
410 HTAB 10: VTAB 12: PRINT "WITH ADAM"
420 FOR i = 1 TO 2000: NEXT i: HOME
430 PRINT "JUST ENTER YOUR COLOR CODE": PRINT
440 PRINT "USE THESE KEYS TO DRAW": PRINT: PRINT
450 HTAB 5: PRINT "UP","PRESS t": PRINT
470 HTAB 5: PRINT "DOWN","PRESS v": PRINT
480 HTAB 5: PRINT "RIGHT","PRESS g": PRINT
490 HTAB 5: PRINT "LEFT","PRESS f": PRINT
500 FOR i = 1 TO 5000: NEXT i: HOME
510 PRINT "PRESS c TO CHANGE COLORS": PRINT
520 PRINT "PRESS q TO CLEAR THE SCREEN": PRINT
530 PRINT "PRESS p TO STOP"
540 FOR i = 1 TO 5000: NEXT i
550 RETURN
```

—Line 50 names the program.
—Line 100 clears the screen.
—Line 110 branches to line 400.
—Line 120 sets the first row and column values as Row 20,Column 20.
—Line 130 activates the low-resolution mode.
—Lines 140 to 145 allow you to input one of the color code numbers.

—Line 150 plots the block first at Row 20, Column 20 and later according to the x and y values stated in lines 200 to 230.
—Line 160 tells the computer to expect some input from the keyboard.
—Line 170 clears the screen if the q key is pressed.
—Line 180 stops the program if the p key is pressed.
—Line 190 branches to line 140 and allows you to select another color if the c key is pressed.
—Line 200 sets y = y − 1 if t is pressed; this moves the "brush" up.
—Line 210 sets y = y + 1 if v is pressed; this moves the "brush" down.
—Line 220 sets x = x − 1 if f is pressed; this moves the "brush" to the left.
—Line 230 sets x = x + 1 if g is pressed; this moves the "brush" to the right.
—Lines 240 to 270 keep the brush on the screen.
—Line 280 branches to line 150.
—Lines 400 to 550 display the beginning prompts and directions for drawing.



FIGURES

This is a shape recognition program. In this program, the computer selects a random number between 1 and 4. Depending on the number selected, the computer will draw a circle, square, rectangle, or triangle. You then identify the shape and the computer will tell you if you are right or wrong. Input the following program.

```
100 REM  Figures
110 n = RND(-n)
130 r = INT(1 + 4 * RND(1))
150 IF r = 1 THEN GOSUB 1000
160 IF r = 2 THEN GOSUB 2000
170 IF r = 3 THEN GOSUB 3000
180 IF r = 4 THEN GOSUB 4000
190 PRINT "WHAT SHAPE IS THE FIGURE ABOVE": FOR i
    = 1  TO 1000: NEXT i
200 PRINT "CIRCLE=c","TRIANGLE=t"
210 PRINT "SQUARE=s","RECTANGLE=r"
220 INPUT a$
230 IF r = 1 AND a$ = "c" THEN GOSUB 5000
240 IF r = 1 AND a$ <> "c" THEN GOSUB 6000
250 IF r = 2 AND a$ = "s" THEN GOSUB 5000
260 IF r = 2 AND a$ <> "s" THEN GOSUB 6000
270 IF r = 3 AND a$ = "r" THEN GOSUB 5000
280 IF r = 3 AND a$ <> "r" THEN GOSUB 6000
290 IF r = 4 AND a$ = "t" THEN GOSUB 5000
300 IF r = 4 AND a$ <> "t" THEN GOSUB 6000
310 GOTO 110
1000 REM  Circle
1010 HGR: HCOLOR = 6
1020 cx = 120: cy = 80: q = 20
1030 bq = 0: eq = 6.6: s = .3
1040 HPLOT cx+q,cy
1050 FOR i = bq TO eq STEP s
1060 x = q * COS(i) + cx
1070 y = -q * SIN(i) + cy
1080 HPLOT x,y
1090 NEXT i
1100 RETURN
2000 HGR: HCOLOR = 13
2010 FOR x = 110 TO 130: FOR y = 70 TO 90
2020 HPLOT x,y: NEXT y: NEXT x
2030 RETURN
3000 HGR: HCOLOR = 5
3010 FOR x = 110 TO 130: FOR y = 75 TO 85
3020 HPLOT x,y: NEXT y: NEXT x
3030 RETURN
4000 HGR: HCOLOR = 12
4010 HPLOT 110,80 TO 130,80
4020 HPLOT TO 120,50
4030 HPLOT TO 110,80
4040 RETURN
```

```
5ØØØ HOME: PRINT "YOU ARE CORRECT"
5Ø1Ø FOR i = 1 TO 5ØØ: NEXT i
5Ø2Ø FOR i = 1 TO 1Ø: PRINT CHR$(7): NEXT i
5Ø3Ø RETURN
6ØØØ HOME: PRINT "SORRY, YOU ARE WRONG"
6Ø1Ø FOR i = 1 TO 5ØØ: NEXT i
6Ø2Ø PRINT CHR$(7)
6Ø3Ø RETURN
```

—Line 100 names the program.
—Lines 110 to 130 tell the Adam computer to randomize.
—Lines 150 to 180 evalute the random number and branch to line 1000 to draw the circle, line 2000 to draw the square, line 3000 to draw the rectangle, or line 4000 to draw the triangle.
—Lines 190 to 210 print the beginning instructions.
—Line 220 allows you to input either a "c", "s", "r", or "t" to identify the shape on the screen.
—Lines 230 to 300 evaluate your input. If the correct letter corresponds to the correct random number, the program branches to line 5000. If the letters and numbers do not correspond, the program branches to line 6000 for the "wrong answer" display.
—Line 310 loops back to line 110.
—Lines 1000 to 1100 plot the circle.
—Lines 2000 to 2030 plot the square.
—Lines 3000 to 3030 plot the rectangle.
—Lines 4000 to 4040 plot the triangle.
—Lines 5000 to 5030 display the "correct answer" display and play the "correct answer" sounds.
—Lines 6000 to 6030 display the "wrong answer" display and play the "wrong answer" sounds.

You can change the shapes that are to be identified to anything you like. You can also add a scoring device if you like. Let your imagination guide you and have some fun.

STATES AND CAPITALS

This is our centerpiece educational game because it provides such a useful program structure for the *States and Capitals* game as well as many other educational games.

You are asked to name the capitals of all fifty states and are then scored and evaluated at the end of the game. Both color and sound greet you in this game in the scoring subroutines as well as when you enter an incorrect or correct response.

The structure of the overall program is similar to *Math Test* and *Multiplication Tables*. It uses a powerful BASIC tool—arrays. With the DIM statement, we can put the states and capitals in an array and call them at random. Input the program and see how well you can do.

```
50 REM  States
100 HOME
110 GOSUB 1000
120 q = 0: nc = 0: n = 0
130 DIM a$(50,2)
135 RESTORE
140 FOR z = 1 TO 50:READ a$(z,1), a$(z,2): NEXT z
144 INPUT "ENTER A NUMBER PLEASE? ";n
146 HOME
150 n = RND(-n)
160 z = INT(1 + 50 * RND(n))
180 IF q = 10 GOTO 270
190 q = q + 1
200 PRINT "WHAT IS THE CAPITAL OF"
202 PRINT a$(z,1); " ? "
220 INPUT x$
225 GOSUB 700
230 PRINT
240 IF x$ = a$(z,2) THEN GOSUB 800
250 IF x$ <> a$(z,2) THEN GOSUB 900
260 GOTO 160
270 per = nc/q
275 HOME: PRINT "YOU ANSWERED ";nc;"/";q;"
    CORRECTLY."
280 IF per >.89 THEN 3000
290 IF (per <.89) AND (per >=.7) THEN 4000
300 GOTO 5000
500 DATA  "ALABAMA","MONTGOMERY","NEW
    YORK","ALBANY",  "NEVADA","CARSON
    CITY","OHIO","COLUMBUS"
510 DATA  "IOWA","DES MOINES","SOUTH CAROLINA",
    "COLUMBIA","OREGON","SALEM","TEXAS","AUSTIN"
520 DATA  "KENTUCKY","FRANKFORT","NEW JERSEY",
    "TRENTON","MASSSACHUSETTS","BOSTON","ARIZONA",
    "PHOENIX"
```

```
530 DATA  "TENNESSEE","NASHVILLE","WYOMING",
    "CHEYENNE","MINNESOTA","ST. PAUL","MARYLAND",
    "ANNAPOLIS"
540 DATA  "NEW MEXICO","SANTA FE","NORTH
    CAROLINA","RALEIGH","CONNECTICUT","HARTFORD"
550 DATA  "ILLINOIS","SPRINGFIELD","MAINE",
    "AUGUSTA","MICHIGAN","LANSING","GEORGIA",
    "ATLANTA","ALASKA","JUNEAU"
560 DATA  "MONTANA","HELENA","ARKANSAS","LITTLE
    ROCK","CALIFORNIA","SACRAMENTO","COLORADO",
    "DENVER","DELAWARE","DOVER"
570 DATA  "FLORIDA","TALLAHASSEE","HAWAII",
    "HONOLULU","IDAHO","BOISE","INDIANA",
    "INDIANAPOLIS"
580 DATA  "KANSAS","TOPEKA","LOUISIANA","BATON
    ROUGE","MISSISSIPPI","JACKSON"
590 DATA  "UTAH","SALT LAKE CITY","NEBRASKA",
    "LINCOLN","NEW HAMPSHIRE","CONCORD","NORTH
    DAKOTA","BISMARK"
600 DATA  "OKLAHOMA","OKLAHOMA CITY",
    "PENNSYLVANIA","HARRISBURG","RHODE ISLAND",
    "PROVIDENCE"
610 DATA  "SOUTH DAKOTA","PIERRE","VERMONT",
    "MONTPELIER","VIRGINIA","RICHMOND",
    "WASHINGTON","OLYMPIA"
620 DATA  "WEST VIRGINIA","CHARLESTON",
    "WISCONSIN","MADISON","MISSOURI","JEFFERSON
    CITY"
695 REM  Convert Characters to Uppercase
700 IF x$ = "" THEN RETURN
705 x = LEN(x$): te$ = "": FOR i = 1 TO x
710 t$ = MID$(x$,i,1): IF ASC(t$)>96 THEN t$=
    CHR$(ASC(t$)-32)
720 te$ = te$ + t$
730 NEXT i: x$ = te$: RETURN
800 PRINT "YOU ARE CORRECT"
810 FOR i = 1 TO 1000: NEXT i
820 FOR i = 1 TO 5: PRINT CHR$(7): NEXT i
820 nc = nc + 1
840 HOME: RETURN
900 PRINT "WRONG, ANSWER IS ";a$(z,2)
910 FOR i = 1 TO 1500: NEXT i
920 PRINT CHR$(7)
930 HOME: RETURN
995 REM  Draws blue section of flag
```

```
1000 HGR: HCOLOR = 9
1010 FOR x = 0 TO 76: HPLOT 4,x+0 TO 110, x+0:
    NEXT x
1015 REM  Place the short stripes on the flag
1020 c = 2: HCOLOR = c
1030 FOR x = 1 TO 77 STEP 11: FOR nl = 1 TO 10:
    HPLOT   112,x-1+nl TO 250,x-1+nl: NEXT nl
1040 c = 5 - c: HCOLOR = c: NEXT x
1045 REM  Draws rest of stripes
1050 FOR x = 1 TO 66 STEP 11: FOR nL = 1 TO 10:
    HPLOT  4,x+76+nl TO 250,x+76+nl: NEXT nl
1060 c = 5 - c: HCOLOR = c: NEXT x
1065 REM  Place the stars on the flag
1070 row = 4
1072 FOR line = 1 TO 4
1073 c = 3: HCOLOR = c
1074 FOR i = 10 TO 104 STEP 8
1078 FOR j = 0 TO 3: HPLOT i+j,row TO i+j,row+4:
    NEXT  j
1080 c = 12 - c: HCOLOR = c: NEXT i
1083 REM  Second row of stars
1084 pt=10: ad=6: c=9: HCOLOR=c: row=row+8
1086 FOR ns = 1 TO 10
1088 FOR nr = 0 TO 2: HPLOT pt+nr,row TO
    pt+nr,row+4:  NEXT nr
1090 c=12-c: HCOLOR=c: ad=16-ad: pt=pt+ad
1092 NEXT ns
1093 row = row + 8
1094 NEXT line
1158 REM  Last line of stars printed
1160 c = 3:HCOLOR = 3
1162 FOR i = 10 TO 104 STEP 8
1164 FOR j = 0 TO 3:HPLOT i+j,68 TO i+j,72: NEXT j
1166 c=12-c: HCOLOR=c
1290 RETURN
3000 PRINT "YOU REALLY KNOW YOUR STATES"
3010 PRINT "AND CAPITALS.  GOOD JOB"
3020 FOR i = 1 TO 3000: NEXT i
3030 END
4000 PRINT "OK, BUT YOU COULD USE SOME"
4010 PRINT "PRACTICE."
4020 FOR i = 1 TO 3000: NEXT i
4030 END
5000 PRINT "YOU NEED TO STUDY SOME MORE."
5010 FOR i = 1 TO 3000: NEXT i
5020 END
```

—Line 50 names the program.
—Line 100 clears the screen.
—Line 110 branches to line 1000.
—Line 120 sets the initial value of the number of questions (q) and the number of correct answers (nc).
—Line 130 dimensions array a$. There are 50 components in the array.
—Line 135 restores the DATA statement.
—Line 140 defines z as 1 to 50 and reads a$ from DATA.
—Lines 150 to 160 select a random number from 1 to 50. This will select a state.
—Line 180 states that when 10 questions are asked, it branches to line 270. This can be changed to make a longer quiz.
—Line 190 keeps track of the number of questions asked.
—Lines 200 to 202 print the question in quotes and a "state" from the DATA statement.
—Line 220 allows you to input the "capital."
—Line 225 jumps to line 700.
—Line 240 states that if x$ (your answer) equals a$(z,2), then branch to line 800.
—Line 250 states that if x$ (your answer) doesn't equal a$(z,2), then branch to line 900.
—Line 260 loops back to line 160.
—Lines 270 to 300 compute the score percentage and branch to the appropriate line.
—Lines 500 to 620 hold the states and capitals data.
—Lines 700 to 730 convert all answers to uppercase characters.
—Lines 800 to 840 display the "correct answer" prompt and play the "correct answer" sound.
—Lines 900 to 930 display the "wrong answer" prompt and play the "wrong answer" sound.
—Lines 1000 to 1290 plot and display the flag.
—Lines 3000 to 3030 display the prompt for 100% to 90% correct.
—Lines 4000 to 4020 display the prompt for 80% to 70% correct.
—Lines 5000 to 5020 display the prompt for 70% or less correct.

Suggested Program Changes and "Dressing"

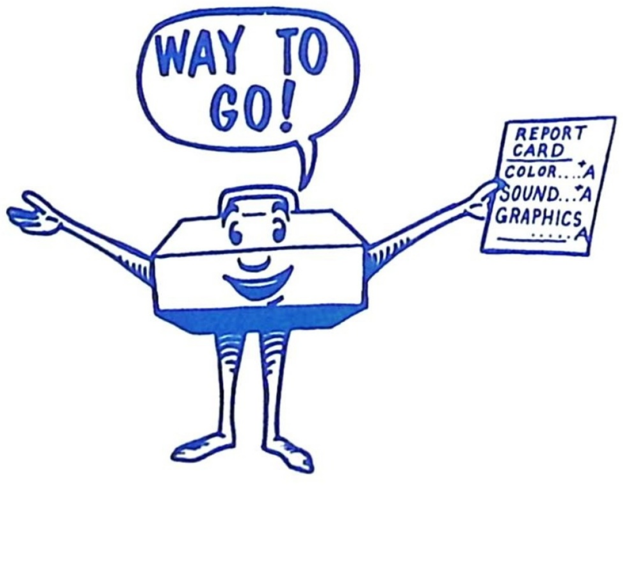
We have included color and sound subroutines in this game. You should improve them for yourself. There are a number of ways to adapt this game.

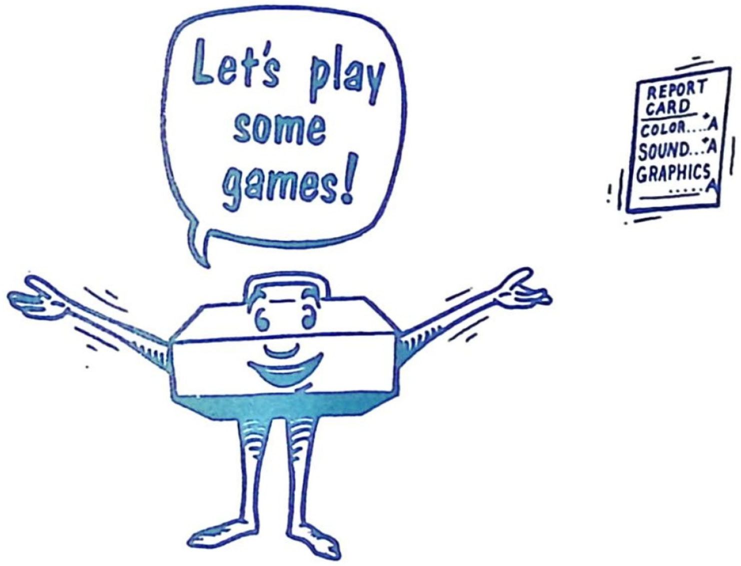
1. Change the data for the *States and Capitals*—either more data or less data will change the game.
2. Change the scoring levels.

This is the last educational game we use as an illustration. We hope you have lots of ideas to use for games and quizzes now. Some of the kinds of ideas we would like to suggest include:

Countries and Capitals
Continents and Countries
Inventors and Objects
Historical Events and Dates
Mythological Characters
Countries and Their Currency
Biblical Names
Diseases and Their Meanings
Music and Musicians
Chemical Elements and Their Symbols
Countries and Their Leaders
Words and Their Opposite Meanings
Numbers and Their Spelling
Roman Numerals and Numbers
Works of Art and Artists

In the next chapter on traditional games, we expand our knowledge of BASIC programming while continuing to have fun.





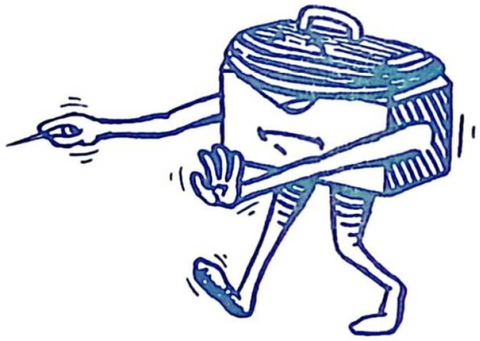
Traditional Games 9

For our purposes, Traditional Games are those games that have been played for years with dice, cards, paper and pencil, or gameboards. In this chapter, we will look at four such games and change them so that they can be played with your Adam computer. Using these games as models, you should be able to create or adapt your own traditional games. Let's begin with a very easy game.

One of the oldest and simplest of the traditional computer games is the guessing game. In *Guessing Game,* you can guess colors, sounds, numbers, or just about anything else you want to try. Regardless of what is guessed, the program will be about the same. The computer picks a number, color, or whatever, by using the *randomize command* and the *random function*. Then you try and guess the computer's pick by using either INPUT statements or the GET statement. The computer checks to see if you guessed correctly or incorrectly and lets you know the outcome. Let's look at a color guessing game.

GUESSING GAME

MAGENTA	WHITE
YELLOW	DARK BLUE
BLACK	GREY 1
MEDIUM RED	CYAN
DARK RED	GREY 2
LIGHT BLUE	LIGHT RED
LIGHT YELLOW	LIGHT GREEN
MEDIUM GREEN	DARK GREEN



```
50 REM  Color Pic
100 HOME
105 INPUT "ENTER A NUMBER PLEASE? ";n
110 n = RND(-n)
120 c = INT(1+15*RND(1))
140 t = 0
145 HOME
150 PRINT "TO EXIT, TYPE 99"
152 INPUT "PICK A COLOR CODE ";y
154 IF y = 99 THEN END
155 PRINT
```

```
16Ø t = t + 1
17Ø PRINT "THAT WAS TRY NUMBER ";t
19Ø IF c<>y GOTO 29Ø
2ØØ GR: COLOR = c
21Ø FOR y = Ø TO 39: FOR x = Ø TO 3Ø
22Ø PLOT x,y
23Ø NEXT x: NEXT y
24Ø PRINT "YOU GUESSED CORRECTLY"
25Ø PRINT "IT TOOK YOU ";t;" TRIES"
26Ø FOR i = 1 TO 3ØØØ
27Ø NEXT i: TEXT
28Ø GOTO 11Ø
29Ø PRINT "YOU GUESSED WRONG"
3ØØ PRINT "TRY AGAIN"
31Ø FOR i = 1 TO 2ØØØ: NEXT i
32Ø GOTO 145
```

This game is simple, but it has all the components of a playable guessing game. The computer picks a color from 1 to 15 in line 120 and you must guess the correct color. Let's look at each line of the program.

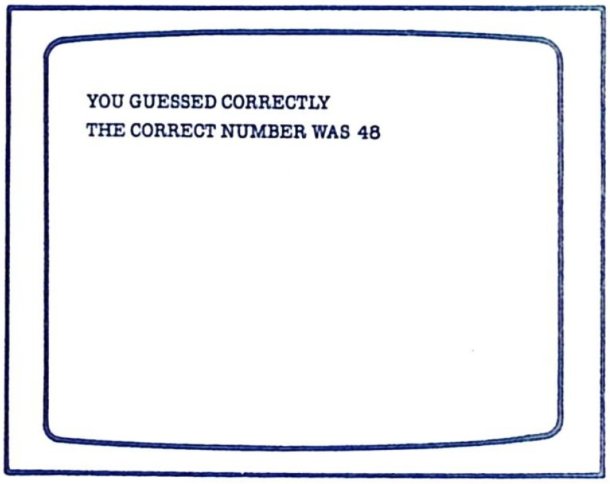
—Line 50 names the program.
—Line 100 clears the screen.
—Lines 110 to 120 tell the Adam computer to randomize and select a color number and then call that color c.
—Line 140 sets the number of tries, or t, at 0.
—Lines 150 to 154 print the prompt and allow us to select a color number from 1 to 15. It names our selection y.
—Line 155 prints a blank line.
—Line 160 keeps track of the number of tries.
—Line 170 prints the try number.
—Line 190 states that if c < > y, the program branches to line 290.
—Line 200 activates the low-resolution mode and defines the color as the random number selected in line 120.
—Lines 210 to 230 display a screen that is full of color if you guess the right color.
—Line 240 prints the "correct guess" prompt.
—Line 250 tells you how many tries it took you to guess correctly.
—Line 260 holds the "correct guess" prompt on the screen.
—Line 270 activates the text mode.
—Line 280 branches to line 110.
—Lines 290 and 300 print the "wrong guess" prompt.

—Line 310 holds the "wrong guess" prompt on the screen.
—Line 320 branches to line 145.

When you run the program, the Adam computer asks you to input a number to seed the random number generator. Then you enter a color code. If you guess correctly, the screen will fill with the color that the Adam picked at random. The program even keeps track of how many guesses it takes you to get the color right.

GUESSING A NUMBER

This next program lets you try to guess a number that the Adam computer selects at random. But, in this game, we will let the computer help us with our guesses. If our guess is too high or too low, the computer will tell us. Let's try *"NUMBER PIC."*



```
YOU GUESSED CORRECTLY
THE CORRECT NUMBER WAS 48
```

```
5Ø REM  Number Pic
1ØØ HOME
1Ø5 INPUT "ENTER A NUMBER PLEASE? ";n
11Ø n = RND(n)
12Ø x = INT(1+5Ø*RND(1))
14Ø t = Ø
144 HOME
145 PRINT "PICK A NUMBER FROM"
15Ø INPUT "     1 TO 5Ø ";y
152 IF y < 1 OR y > 5Ø THEN 145
```

```
155 PRINT
160 t = t + 1
170 PRINT "THAT WAS TRY NUMBER ";t
190 IF x > y GOTO 250
200 IF x < y GOTO 280
220 HOME
225 PRINT "YOU GUESSED CORRECTLY"
230 PRINT: PRINT "THE CORRECT NUMBER WAS ";x
235 PRINT: PRINT "IT ONLY TOOK YOU ";t;" TRIES"
236 PRINT: PRINT: PRINT: PRINT
237 INPUT "PLAY AGAIN? (Y/N) ";y$
238 IF y$ <> "Y" AND y$ <> "y" AND y$ <> "N" AND
    y$ <>  "n" THEN 237
239 IF y$ = "Y" OR y$ = "y" THEN 110
240 END
250 PRINT "YOUR GUESS ";y;" IS TOO LOW"
260 FOR i = 1 TO 2000: NEXT i: GOTO 144
280 PRINT "YOUR GUESS ";y;" IS TOO HIGH"
290 FOR i = 1 TO 2000: NEXT i: GOTO 144
```

This program works exactly like *"COLOR PIC,"* except that the computer gives us the "too high" or "too low" hints. Line 190 says that if x is greater than our guess (y), then the program should branch to line 250 and tell us our guess was too low. Line 200 tests for the "high guess" condition in the same way.

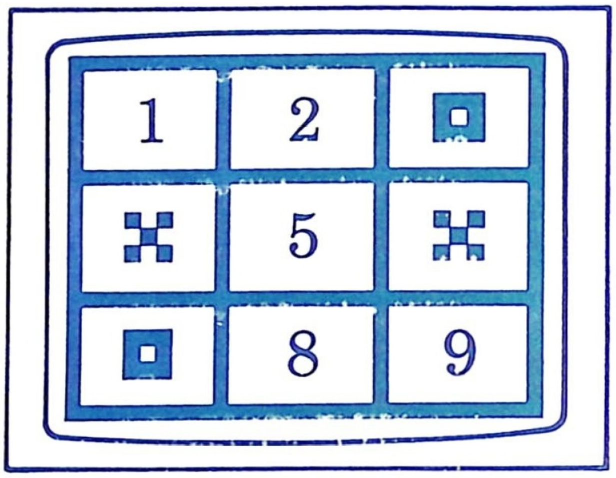
This is a good game for you to experiment with. Add some sound to signal a correct answer, or perhaps some color. Nest your sound and color additions in the "win condition" statements between lines 225 and 235. Have fun.

TIC-TAC-TOE

Let's try another old favorite, Tic-Tac-Toe. Instead of pencil and paper, you can play on the display screen.

This is a two-player game. The program will display nine blocks—three across by three down. Each square contains a number associated with that particular square; just press the number for that square. The computer will tell you who won the game.

Enjoy.



```
10 REM  Tic-Tac-Toe
50 HOME
70 DIM tbl(3,3)
100 PRINT "  LET'S PLAY TIC-TAC-TOE": PRINT:
    PRINT: PRINT
110 PRINT "  THE NUMBERS ON THE SCREEN": PRINT
115 PRINT "WILL HELP YOU PLACE YOUR X'S": PRINT
120 PRINT "AND O'S IN THE CORRECT BOX."
140 FOR i = 1 TO 10: PRINT CHR$(7);: NEXT i: PRINT
150 FOR i = 1 TO 2500: NEXT i
152 FOR i = 1 TO 3: FOR j = 1 TO 3: tbl(i,j) = 0:
    NEXT  j: NEXT i
160 HOME
162 PRINT: PRINT "ENTER PLAYER #1's": INPUT "NAME
    ? ";  p1$
163 IF p1$ = "" THEN 162
164 PRINT: PRINT "ENTER PLAYER #2's": INPUT "NAME
    ? ";  p2$
165 IF p2$ = "" THEN 164
166 HOME
167 PRINT: PRINT: PRINT p1$: INPUT "DO YOU WANT
    X's OR  O's ?";c1$
168 IF c1$ <> "X" AND c1$ <> "x" AND c1$ <> "O"
    AND  c1$ <> "o" THEN 167
169 IF c1$ = "X" OR c1$ = "x" THEN c2$ = "O":
    GOTO 171
170 c2$ = "X"
171 GOSUB 2000
172 player = 1
173 FOR move = 1 TO 9
```

```
174 HOME
175 IF player = 1 THEN PRINT p1$;" --- ";c1$;"'s."
177 IF player = 2 THEN PRINT p2$;" --- ";c2$;"'s."
180 PRINT "What square ? ";
180 GET sn$: IF sn$ < "1" OR sn$ > "9" THEN 174
190 PRINT: sn = VAL(sn$)-1: rw = Ø
200 IF sn > 5 THEN rw = 2: sn = sn - 6: GOTO 230
210 IF sn > 2 THEN rw = 1: sn = sn - 3
230 srow = (rw*13)+2: bc = (sn*13)+4
235 IF tbl(rw+1, sn+1) <> Ø THEN 174
240 GOSUB 1800
250 IF player = 1 AND (c1$ = "X" OR c1$ = "x")
    THEN  GOSUB 1500: GOTO 310
260 IF player = 1 AND (c1$ = "O" OR c1$ = "o")
    THEN  GOSUB 1600: GOTO 310
270 IF player = 2 AND (c1$ = "X" OR c1$ = "x")
    THEN  GOSUB 1500: GOTO 310
280 IF player = 2 AND (c1$ = "O" OR c1$ = "o")
    THEN  GOSUB 1600: GOTO 310
310 tbl(rw+1, sn+1) = (2*player) - 3
318 REM  Check for a winner
319 REM  First check horizontally
320 FOR i = 1 TO 3: total = Ø
330 total = tbl(i,1) + tbl(i,2) + tbl(i,3)
340 IF ABS(total) = 3 THEN 450
350 NEXT i
355 REM  Check vertically
360 FOR i = 1 TO 3: total = Ø
370 total = tbl(1,i) + tbl(2,i) + tbl(3,i)
380 IF ABS(total) = 3 THEN 450
390 NEXT i
395 REM  Check Diagonally
400 IF ABS(tbl(1,1) + tbl(2,2) + tbl(3,3)) = 3
    THEN  450
410 IF ABS(tbl(1,3) + tbl(2,2) + tbl(3,1)) = 3
    THEN  450
420 player = 3 - player
430 NEXT move
435 HOME: PRINT CHR$(7): PRINT "Chalk one up for
    the cat !":  GOTO 480
445 REM  We have a winner
450 PRINT CHR$(7); CHR$(7); CHR$(7)
455 HOME: IF player = 1 THEN PRINT p1$
460 IF player =2 THEN PRINT p2$
470 PRINT "is the winner! "
480 FOR i = 1 TO 5000: NEXT i
490 HOME: INPUT "Play again ? (Y/N) ";y$
```

```
500 IF y$<>"Y" AND y$<>"y" AND y$<>"N" AND
    y$<>"n"  THEN 490
510 IF y$="y" OR y$="Y" THEN TEXT: GOTO 152
520 END
1497 REM  Draw an X in the square
1500 FOR i = 1 TO 8: PLOT bc+(i-1),srow+i
1510 PLOT bc+(8-i), srow+i: NEXT i
1520 RETURN
1597 REM  Draw an O in the square
1600 HLIN bc+1,bc+4 AT srow+1: VLIN srow+2,
    srow+6 AT  bc
1610 VLIN srow+2, srow+6 AT bc+5: HLIN bc+1,bc+4
    AT  srow+7
1620 RETURN
1798 REM  Erases the number from a square
1800 COLOR = 0
1806 FOR offset = 1 TO 12: VLIN srow, srow+10 AT
    bc+offset-4: NEXT offset
1808 COLOR = 2
1810 RETURN
2000 REM  Plots the board
2010 GR: COLOR = 2
2020 HLIN 0,39 AT 0: HLIN 0,39 AT 13: HLIN 0,39
    AT 26:  HLIN 0,39 AT 39
2030 VLIN 0,39 AT 0: VLIN 0,39 AT 13: VLIN 0,39
    AT 26:  VLIN 0,39 AT 39
2033 COLOR = 3
2036 REM  Draws the #1 in upper left-hand corner
2040 PLOT 5,4: PLOT 6,3: PLOT 7,2: VLIN 2,11 AT 7
2050 HLIN 5,9 AT 11
2052 REM  Draws the #2 in middle top row
2054 PLOT 17,4: PLOT 18,3: PLOT 19,2: PLOT 20,2:
    PLOT  21,2
2056 PLOT 22,3: PLOT 23,4: PLOT 23,5: PLOT 22,6:
    PLOT  21,6: PLOT 20,6
2058 PLOT 19,7: PLOT 18,7: PLOT 17,8: PLOT 17,9:
    PLOT  17,10
2060 HLIN 17,23 AT 11
2062 REM  Places the #3 in top right row
2064 PLOT 30,3: HLIN 31,35 AT 2: VLIN 3,5 AT 36:
    HLIN  33,35 AT 6
2066 VLIN 7,11 AT 36: HLIN 31,35 AT 11: PLOT 30,10
2078 REM  Places the #4 in middle row - left side
2080 VLIN 15,20 AT 4: HLIN 4,9 AT 20: VLIN 15,24
    AT 8
```

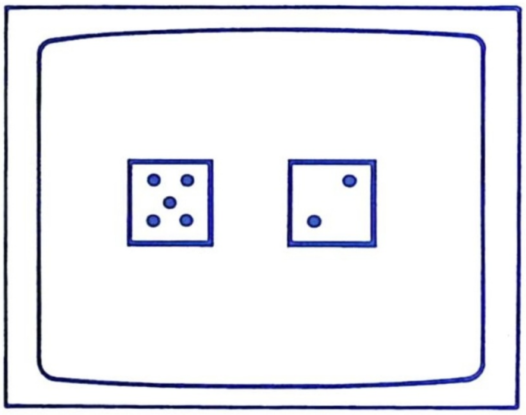
```
2082 REM  Places the #5 in middle row - middle
2084 HLIN 17,20 AT 15: VLIN 15,19 AT 17: HLIN
    18,20 AT  19
2086 VLIN 20,23 AT 21: HLIN 18,20 AT 24: PLOT 17,23
2090 REM  Places the #6 in middle row - right
2092 HLIN 32,35 AT 15: PLOT 36,16: PLOT 31,16:
    VLIN  17,22 AT 30
2094 PLOT 31,23: HLIN 32,35 AT 24: VLIN 23,21 AT
    36
2096 HLIN 35,32 AT 20: PLOT 31,21: PLOT 31,22
2098 REM  Places the #7 in bottom row - left
2100 HLIN 4,8 AT 28: PLOT 8,29: PLOT 8,30: PLOT
    7,31:  PLOT 7,32
2102 PLOT 6,33: PLOT 6,34: PLOT 5,35: PLOT 5,36
2104 REM  Places the #8 in bottom row - middle
2106 HLIN 18,20 AT 28: VLIN 29,31 AT 17: VLIN
    29,31 AT  21
2108 HLIN 18,20 AT 32: VLIN 33,35 AT 17: VLIN
    33,35 AT  21
2110 HLIN 18,20 AT 36
2112 REM  Places the #9 in bottom row - right
2114 HLIN 31,35 AT 28: VLIN 29,31 AT 30: VLIN
    29,31 AT  36
2116 HLIN 31,35 AT 32: PLOT 35,33: PLOT 34,34
2117 PLOT 33,35: PLOT 32,36: PLOT 31,37
2118 COLOR = 2
2120 RETURN
```

—Line 10 names the program.
—Line 50 clears the screen.
—Line 70 dimensions the table size.
—Lines 100 to 150 display the beginning prompts.
—Line 152 initializes the array to 0's.
—Line 160 clears the screen.
—Lines 162 to 170 enter players' name and the players' chosen marker, either an X or an O.
—Line 171 branches to line 2000 and the playing board is plotted.
—Line 172 sets the counter for the player number.
—Lines 173 to 430 start the game play. They allow 9 moves and check for a winner after each move. They first check horizontally for a winner, then vertically, and finally diagonally.
—Line 435 indicates that the game was a draw.
—Lines 445 to 480 indicate and name the winner.

—Lines 490 to 520 check to see if you want to play again.
—Lines 1497 to 1520 draw an X in the appropriate square.
—Lines 1597 to 1620 draw on O in the appropriate square.
—Lines 1798-1810 blank out a square before drawing in the X or O.
—Lines 2000-2120 set up the board.

Play the game and see if you don't find it a little more exciting than playing the old way. Our kids enjoy it and the game helps the little ones recognize numbers.

TED AND DAVE'S CASINO



How about a little gambling for you older folks. We call this next game "TED AND DAVE'S CASINO." At your command, the Adam computer rolls the dice. You begin play with $10.00. If you roll a seven or eleven, you win $2.00. If you roll less than a four or a twelve, you lose $2.00. If you roll any other number, the house takes a dollar for the privilege of rolling. The program gives you the rules and tells you how to play.

Here is the listing. Each line is explained in the section following the program.

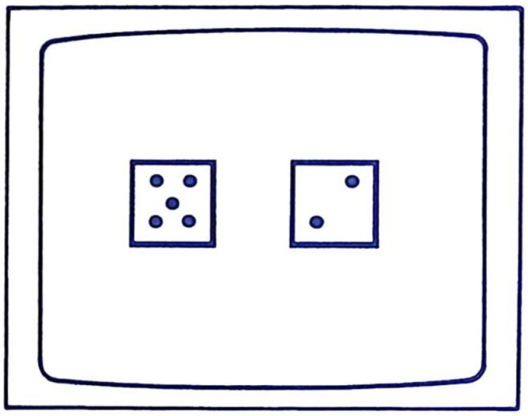
```
50 REM Dice
100 HOME
110 m = 10
120 GOSUB 3000
125 n = RND(-n)
130 IF m > 0 THEN 140
```

```
132 HOME: PRINT CHR$(7): PRINT "YOU ARE BROKE !!"
    : END
140 INPUT "TO ROLL ... PRESS RETURN "; k$
150 GOSUB 2000
160 GR: COLOR = 15
170 HLIN 7, 17 AT 10: HLIN 7, 17 AT 20: VLIN 10,
    20 AT  7: VLIN 10, 20 AT 17
180 HLIN 24, 34 AT 10: HLIN 24, 34 AT 20: VLIN
    10, 20  AT 24: VLIN 10, 20 AT 34
190 IF rd <> 1 GOTO 210
200 PLOT 29, 15
210 IF rd <> 2 GOTO 230
220 PLOT 27, 17: PLOT 31, 13
230 IF rd <> 3 GOTO 250
240 PLOT 27, 17: PLOT 31, 13: PLOT 29, 15
250 IF rd <> 4 GOTO 280
260 PLOT 26, 12: PLOT 32, 12
270 PLOT 26, 18: PLOT 32, 18
280 IF rd <> 5 GOTO 310
290 PLOT 26, 12: PLOT 32, 12
300 PLOT 26, 18: PLOT 32, 18: PLOT 29, 15
310 IF rd <> 6 GOTO 350
320 PLOT 26, 12: PLOT 32, 12
330 PLOT 26, 18: PLOT 32, 18
340 PLOT 26, 15: PLOT 32, 15
350 IF ld <> 1 GOTO 370
360 PLOT 12, 15
370 IF ld <> 2 GOTO 390
380 PLOT 10, 17: PLOT 14, 13
390 IF ld <> 3 GOTO 410
400 PLOT 10, 17: PLOT 12, 15: PLOT 14, 13
410 IF ld <> 4 GOTO 430
420 PLOT 9, 12: PLOT 15, 12: PLOT 9, 18: PLOT 15,
    18
430 IF ld <> 5 GOTO 450
440 PLOT 9, 12: PLOT 15, 12: PLOT 9, 18: PLOT 15,
    18:  PLOT 12, 15
450 IF ld <> 6 GOTO 480
460 PLOT 9, 12: PLOT 15, 12
470 PLOT 9, 18: PLOT 15, 18: PLOT 9, 15: PLOT 15,
    15
480 FOR i = 1 TO 10: NEXT i
490 IF (rd+ld = 7) OR (rd+ld = 11) GOTO 1000
500 IF (rd+ld = 12) OR (rd+ld < 4) GOTO 1300
510 GOTO 1600
```

—Lines 490 to 520 check to see if you want to play again.
—Lines 1497 to 1520 draw an X in the appropriate square.
—Lines 1597 to 1620 draw on O in the appropriate square.
—Lines 1798-1810 blank out a square before drawing in the X or O.
—Lines 2000-2120 set up the board.

Play the game and see if you don't find it a little more exciting than playing the old way. Our kids enjoy it and the game helps the little ones recognize numbers.

TED AND DAVE'S CASINO



How about a little gambling for you older folks. We call this next game "TED AND DAVE'S CASINO." At your command, the Adam computer rolls the dice. You begin play with $10.00. If you roll a seven or eleven, you win $2.00. If you roll less than a four or a twelve, you lose $2.00. If you roll any other number, the house takes a dollar for the privilege of rolling. The program gives you the rules and tells you how to play.

Here is the listing. Each line is explained in the section following the program.

```
50 REM Dice
100 HOME
110 m = 10
120 GOSUB 3000
125 n = RND(-n)
130 IF m > 0 THEN 140
```

```
132 HOME: PRINT CHR$(7): PRINT "YOU ARE BROKE !!"
    : END
140 INPUT "TO ROLL ... PRESS RETURN "; k$
150 GOSUB 2000
160 GR: COLOR = 15
170 HLIN 7, 17 AT 10: HLIN 7, 17 AT 20: VLIN 10,
    20 AT  7: VLIN 10, 20 AT 17
180 HLIN 24, 34 AT 10: HLIN 24, 34 AT 20: VLIN
    10, 20  AT 24: VLIN 10, 20 AT 34
190 IF rd <> 1 GOTO 210
200 PLOT 29, 15
210 IF rd <> 2 GOTO 230
220 PLOT 27, 17: PLOT 31, 13
230 IF rd <> 3 GOTO 250
240 PLOT 27, 17: PLOT 31, 13: PLOT 29, 15
250 IF rd <> 4 GOTO 280
260 PLOT 26, 12: PLOT 32, 12
270 PLOT 26, 18: PLOT 32, 18
280 IF rd <> 5 GOTO 310
290 PLOT 26, 12: PLOT 32, 12
300 PLOT 26, 18: PLOT 32, 18: PLOT 29, 15
310 IF rd <> 6 GOTO 350
320 PLOT 26, 12: PLOT 32, 12
330 PLOT 26, 18: PLOT 32, 18
340 PLOT 26, 15: PLOT 32, 15
350 IF ld <> 1 GOTO 370
360 PLOT 12, 15
370 IF ld <> 2 GOTO 390
380 PLOT 10, 17: PLOT 14, 13
390 IF ld <> 3 GOTO 410
400 PLOT 10, 17: PLOT 12, 15: PLOT 14, 13
410 IF ld <> 4 GOTO 430
420 PLOT 9, 12: PLOT 15, 12: PLOT 9, 18: PLOT 15,
    18
430 IF ld <> 5 GOTO 450
440 PLOT 9, 12: PLOT 15, 12: PLOT 9, 18: PLOT 15,
    18:  PLOT 12, 15
450 IF ld <> 6 GOTO 480
460 PLOT 9, 12: PLOT 15, 12
470 PLOT 9, 18: PLOT 15, 18: PLOT 9, 15: PLOT 15,
    15
480 FOR i = 1 TO 10: NEXT i
490 IF (rd+ld = 7) OR (rd+ld = 11) GOTO 1000
500 IF (rd+ld = 12) OR (rd+ld < 4) GOTO 1300
510 GOTO 1600
```

```
1000 PRINT "YOU WIN": FOR i = 1 TO 3: PRINT
    CHR$(7); :  NEXT i: PRINT
1010 m = m+2
1020 PRINT "YOU HAVE $"; m
1030 FOR i = 1 TO 3000: NEXT i
1040 HOME: GOTO 130
1300 PRINT "YOU LOSE": PRINT CHR$(7)
1310 m = m-2
1320 PRINT "YOU HAVE $"; m
1330 FOR i = 1 TO 3000: NEXT i
1340 HOME: GOTO 130
1600 PRINT "HOUSE TAKES $1.00"
1610 m = m-1
1620 PRINT "YOU HAVE $"; m
1630 FOR i = 1 TO 3000: NEXT i
1640 HOME: GOTO 130
2000 rd = INT(1+6*RND(1))
2030 ld = INT(1+6*RND(1))
2050 RETURN
3000 HTAB 5: PRINT "WELCOME TO TED AND DAVE'S":
    PRINT
3010 HTAB 15: PRINT "CASINO": PRINT
3020 FOR i = 1 TO 3000: NEXT i
3030 HOME
3040 HTAB 5: PRINT "THIS IS A DICE GAME": PRINT
3050 HTAB 5: PRINT "PRESS RETURN TO ROLL": PRINT
3060 HTAB 5: PRINT "YOU BEGIN WITH $10.00.": PRINT
3070 FOR i = 1 TO 4000: NEXT i
3080 HOME
3090 HTAB 5: PRINT "IF YOU ROLL": PRINT: PRINT
3095 HTAB 5: PRINT "A 7 OR 11", "YOU WIN $2.00":
    PRINT
3100 HTAB 5: PRINT "LESS THAN 4", "YOU LOSE $2.00"
3110 PRINT: HTAB 5: PRINT "A 12", "YOU LOSE $2.00"
3120 PRINT: HTAB 5: PRINT "IF YOU ROLL ANY": PRINT
3130 HTAB 5: PRINT "OTHER NUMBER, THE HOUSE":
    PRINT
3140 HTAB 5: PRINT "TAKES $1.00 FOR THE ROLL."
3150 FOR i = 1 TO 6000: NEXT i
3160 HOME
3170 HTAB 12: PRINT "TRY YOUR LUCK": FOR i = 1 TO
    3000: NEXT i
3172 HOME: PRINT: PRINT: PRINT
3174 INPUT "ENTER A NUMBER PLEASE "; n
3180 HOME: RETURN
```

—Line 50 names the program.
—Line 100 clears the screen.
—Line 110 defines m (your amount of money) at the beginning of the game as $10.00.
—Line 120 branches to line 3000.
—Line 125 starts the random number generator.
—Line 130 checks to see if you have money enough to continue to play.
—Line 140 tells you to press RETURN to roll the dice.
—Line 150 branches to line 2000 and selects two random numbers that are between 1 and 6.
—Line 160 activates the low-resolution mode and sets the color to white.
—Lines 170 and 180 plot the line around the dice display.
—Lines 190 to 470 tell the computer what to do for each possible value of ld (the left dice) and rd (the right dice).

If rd is not equal to 1, the program jumps a line and checks to see if rd = 2 and so on. If ld is equal to 1, line 200 will display the dice character representing 1. The computer continually scans these lines and displays the left- and right-hand dice.

—Line 480 causes the computer to hold for a brief time.
—Lines 490 to 510 are the lines used to identify the scoring conditions.

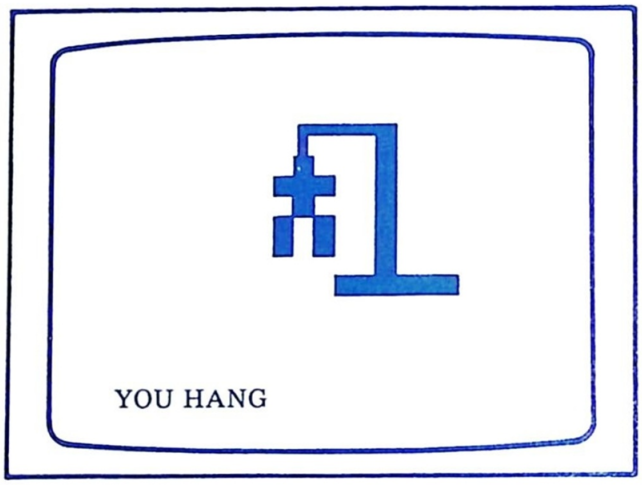
Line 490 is the "win" condition.
Line 500 is the "lose" condition.
Line 510 is the "score" condition with the house taking $1.00 for the roll.
Line 520 loops to line 130.

—Lines 1000 to 1040 display the "win" condition, compute your winnings, print the amount of money you have, and sound a 3-beep signal.
—Lines 1300 to 1340 display the "lose" condition, compute how much money you have, and sound a 1-beep signal.
—Lines 1600 to 1640 display the "house wins" condition and compute how much money you have.
—Lines 2000 to 2050 select the random numbers and call them rd (right dice) and ld (left dice).
—Lines 3000 to 3180 display the opening prompts and directions.

You can add more graphics if you like. Or you can change the stakes by changing lines 1010, 1310, and 1610. You can also change the winning conditions by altering lines 490 to 510. When you are tired of gambling, try the next game.

HANG MAN

This game uses an array to match the letters entered by Player One and the guesses entered by Player Two. Player One actually sets the size of the array when he or she enters the number of letters in the word. Then, as each letter is entered, the array is filled.



Let's look at the program. If you don't fully understand the concept of arrays, don't panic.

```
1 REM   Hang
5 HOME: GOSUB 2ØØØ
1Ø HTAB 3: INPUT "ENTER YOUR WORD ?"; wd$
12 IF wd$ = "" THEN END
14 IF LEN(wd$) > 1Ø THEN 1Ø
2Ø DIM x(1Ø), y$(1Ø)
25 FOR i = 1 TO 1Ø: x(i) = Ø: y$(i) = "_": NEXT i
26 w$ = ""
29 REM   Use the ascii for each letter and
    convert to  uppercase ascii
3Ø wl = LEN(wd$): FOR i = 1 TO wl
32 x(i) = ASC(MID$(wd$, i, 1))
34 IF x(i) > 96 AND x(i) < 123 THEN x(i) = x(i)-32
35 w$ = w$+CHR$(x(i))
36 NEXT i
37 wd$ = w$
39 REM  Draws the hanging scene
```

```
40 GR: COLOR = 2
42 HLIN 16, 27 AT 2: REM   top
44 VLIN 2, 36 AT 27: REM   right side
46 HLIN 12, 32 AT 36: REM   bottom line
48 VLIN 3, 5 AT 16: REM  noose
50 REM  Centers the line for each letter of the
   word
52 bc = INT((30-wl)/2): nm = 0
54 GOSUB 400
57 IF go = 1 THEN 230
58 REM  Start game
62 PRINT: PRINT "PRESS A LETTER ";
64 GET lt$: IF lt$ = "" THEN 64
66 PRINT lt$: flg = 0
70 lt = ASC(lt$): IF lt > 96 AND lt < 123 THEN lt
   =  lt-32
75 FOR i = 1 TO wl
80 IF x(i) = lt THEN y$(i) = CHR$(lt): GOSUB 200:
   flg  = 1
83 NEXT i
85 GOSUB 400: PRINT: IF flg = 1 AND go = 1 THEN
   230
86 IF flg = 1 THEN 92
87 REM  Miss
88 nm = nm+1: COLOR = 3
89 IF nm = 7 THEN GOSUB 160: PRINT "YOU HANG!";
   CHR$(7): GOTO 230
90 ON nm GOSUB 100, 110, 120, 130, 140, 150
91 PRINT "WRONG LETTER !": PRINT CHR$(7);
92 PRINT "CARE TO GUESS ? (Y/N) ";
93 GET c$: IF c$ <> "Y" AND c$ <> "y" AND c$ <>
   "N"  AND c$ <> "n" THEN 93
94 IF c$ = "N" OR c$ = "n" THEN 54
95 GOTO 300
99 REM  Draw the head
100 HLIN 15, 17 AT 6: PLOT 14, 7: PLOT 18, 7:
    VLIN 8,  10 AT 13
102 VLIN 8, 10 AT 19: PLOT 14, 11: PLOT 18, 11:
    HLIN  15, 17 AT 12
104 RETURN
108 REM  Draw the neck
110 VLIN 13, 14 AT 16: RETURN
118 REM  Draw the body
120 FOR tn = 1 TO 9: VLIN 15, 25 AT tn+11: NEXT
    tn:  RETURN
```

```
 128 REM  Draw the right arm
 13Ø VLIN 13, 14 AT 25: VLIN 14, 15 AT 24: VLIN
     15, 16  AT 23
 132 VLIN 16, 17 AT 22: PLOT 21, 17: RETURN
 138 REM  Draw the left arm
 14Ø VLIN 13, 14 AT 7: VLIN 14, 15 AT 8: VLIN 15,
     16  AT 9
 142 VLIN 16, 17 AT 1Ø: PLOT 11, 17: RETURN
 148 REM  Draw the right leg
 15Ø HLIN 18, 19 AT 26: HLIN 18, 19 AT 27: HLIN
     19, 2Ø  AT 28
 152 HLIN 2Ø, 21 AT 29: HLIN 21, 22 AT 3Ø: HLIN
     22, 23  AT 31
 154 RETURN
 158 REM  Draw left leg
 16Ø HLIN 13, 14 AT 26: HLIN 13,14 AT 27: HLIN 12,
     13  AT 28
 162 HLIN 11, 12 AT 29: HLIN 1Ø, 11 AT 3Ø: HLIN 9,
     1Ø  AT 31
 164 RETURN
 198 REM  Correct guess - check for a winner
 2ØØ FOR z = 1 TO wl: IF ASC(y$(z)) <> x(z)  THEN
     RETURN
 21Ø NEXT z
 22Ø go = 1: RETURN
 228 REM  Game over
 23Ø PRINT "GAME OVER ! !"
 24Ø GOTO 1Ø3Ø
 297 REM     check for a guess
 3ØØ HOME: INPUT "ENTER YOUR GUESS "; g$
 3Ø5 t$ = ""
 31Ø FOR i = 1 TO LEN(g$)
 312 tl = ASC(MID$(g$, i, 1))
 314 IF tl > 96 AND tl < 123 THEN tl = tl-32
 316 t$ = t$+CHR$(tl): NEXT i
 318 g$ = t$: IF g$ = wd$ THEN PRINT "YOU ARE
     CORRECT!!"; CHR$(7); CHR$(7): GOTO 23Ø
 32Ø PRINT "INCORRECT GUESS!"
 33Ø FOR I = 1 TO 2ØØØ: NEXT i: GOTO 54
 4ØØ HOME: HTAB bc: FOR i = 1 TO wl: PRINT y$(i);
     :  NEXT i
 41Ø HTAB(24): PRINT "("; nm; ")"
 42Ø RETURN
 1Ø3Ø INPUT "ANOTHER GAME? (Y/N) "; c$
 1Ø33 IF c$ <> "y" AND c$ <> "Y" AND c$ <> "N" AND
     c$  <> "n" THEN 1Ø3Ø
```

```
1045 IF c$ = "N" OR c$ = "n" GOTO 1060
1050 CLEAR: TEXT: HOME: GOTO 10
1060 PRINT "SEE YOU LATER"
1070 FOR i = 1 TO 3000: NEXT i
1080 TEXT: HOME: END
2000 GR
2010 FOR c = 1 TO 15: COLOR = c
2015 REM       -- H --
2020 HLIN 6, 9 AT 7: VLIN 4, 11 AT 5: VLIN 4, 11
   AT 9
2025 REM       -- A --
2030 VLIN 4, 11 AT 13: VLIN 4, 11 AT 17
2040 HLIN 14, 17 AT 4: HLIN 14, 17 AT 8
2045 REM       -- N --
2050 VLIN 4, 11 AT 21: VLIN 4, 11 AT 25
2060 VLIN 4, 11 AT 23
2070 PLOT 22, 4: PLOT 24, 11
2075 REM       -- G --
2080 VLIN 4,11 AT 29: HLIN 29, 33 AT 4
2090 HLIN 29, 33 AT 11: PLOT 33, 9: PLOT 33, 10
2100 HLIN 31, 33 AT 8
2105 REM       -- H --
2110 HLIN 6, 9 AT 22: VLIN 19, 26 AT 5: VLIN 19,
   26 AT  9
2115 REM       -- I --
2120 VLIN 19, 26 AT 13
2125 REM       -- G --
2130 VLIN 19, 26 AT 17: HLIN 17, 21 AT 26
2140 HLIN 17, 21 AT 19
2150 PLOT 21, 24: PLOT 21, 25
2170 HLIN 19, 21 AT 23
2175 REM       -- H --
2180 HLIN 26, 29 AT 22: VLIN 19, 26 AT 25: VLIN
   19, 26  AT 29
2190 NEXT c
2200 FOR i = 1 TO 500: NEXT i
2210 TEXT: HTAB 5: PRINT "HERE IS HOW YOU PLAY.":
   PRINT
2220 HTAB 5: PRINT "PLAYER 1 ENTERS THE ": PRINT
2225 HTAB 5: PRINT "WORD.  THE WORD CANNOT": PRINT
2230 HTAB 5: PRINT "BE LONGER THAN 10": PRINT
2240 HTAB 5: PRINT "LETTERS.": PRINT
2275 PRINT
2280 HTAB 5: PRINT "PLAYER 2 GUESSES": PRINT
2285 HTAB 5: PRINT "LETTERS BY PRESSING": PRINT
2290 HTAB 5: PRINT "A LETTER.": PRINT
```

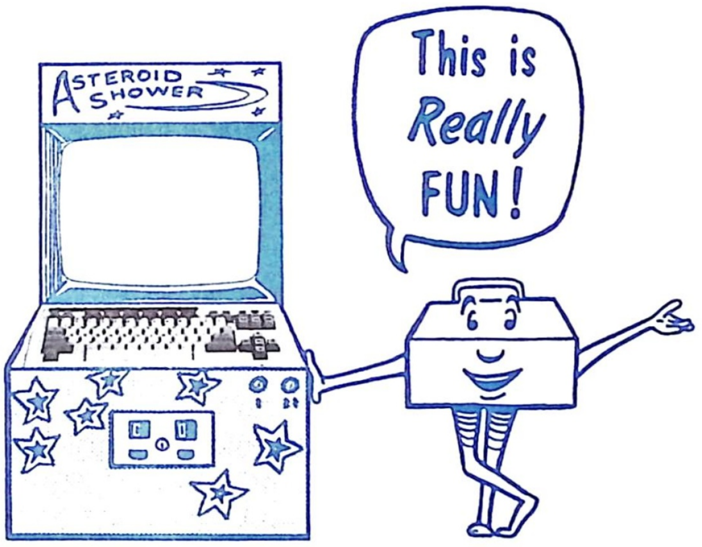
```
2295 FOR i = 1 TO 10000: NEXT i
2297 HOME
2300 HTAB 5: PRINT "YOU ARE ONLY ALLOWED 7": PRINT
2310 HTAB 5: PRINT "INCORRECT GUESSES OR ": PRINT
2320 HTAB 5: PRINT "MISSED LETTERS. AFTER EACH":
    PRINT
2321 HTAB 5: PRINT "GUESS, YOU'LL BE GIVEN": PRINT
2322 HTAB 5: PRINT "A CHANCE TO GUESS THE WORD.":
    PRINT
2324 HTAB 5: PRINT "YOU CAN GUESS AS OFTEN": PRINT
2325 HTAB 5: PRINT "AS YOU LIKE.": PRINT
2326 HTAB 5: PRINT "IF YOU MISS 7 LETTERS,": PRINT
2327 HTAB 12: PRINT "YOU HANG !": PRINT: HTAB 12:
    PRINT "LET'S PLAY."
2330 FOR i = 1 TO 9000: NEXT i
2340 HOME: RETURN
```

—Line 1 names the program.
—Line 5 branches to line 2000.
—Lines 10 to 14 allow you to input the word.
—Lines 20 to 26 dimension the arrays. One array, x, holds the ASCII values for each of the letters in the word. The y$ array holds the dashes (—) until a correct letter has been pressed.
—Lines 29 to 37 convert all ASCII codes to uppercase letters.
—Lines 40 to 48 activate the graphic screen and draw the hanging platform.
—Lines 50 to 52 display the dashes for the word.
—Line 57 checks to see if the game is over.
—Lines 62 to 86 enter a letter and place it in array y$, if it is a correct guess.
—Lines 87 to 91 indicate that a wrong letter was guessed and draw an appropriate body part on the hanging platform.
—Lines 92 to 95 ask if you would like to take a guess.
—Lines 99 to 164 draw different parts of the body.
—Lines 198 to 220 indicate a correct guess.
—Lines 230 to 240 give the GAME OVER prompt.
—Lines 297 to 330 enter your guess and check for a winning guess.
—Lines 400 to 420 display the dashes along with the number of incorrect guesses.
—Lines 1030 to 1033 ask if you want to play again.
—Lines 1060 to 1080 plot the end of the game and stop the program.

—Lines 2000 to 2340 plot the "HANG HIGH" game title and the beginning prompts and directions.

Player One must select a word with no more than 10 letters, otherwise any letters in any combination can be used. Remember that the Adam computer doesn't know how to spell. If a word is misspelled when it is entered, the same misspelling must occur to score a correct guess. If you like, you can enter a word backwards and play backwards Hang Man.

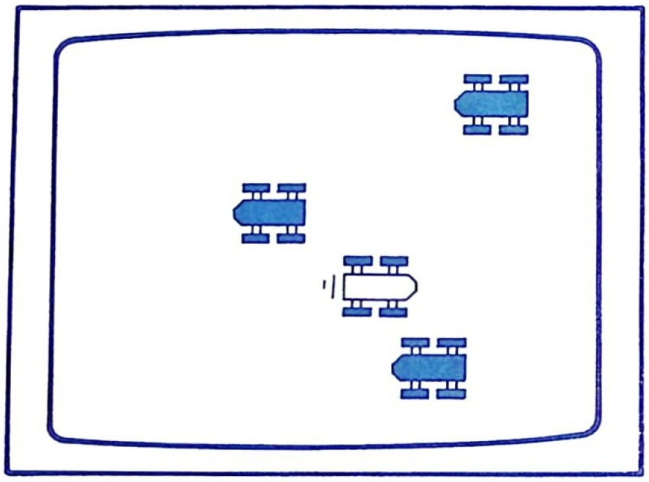
We hope that you have enjoyed the games in this chapter. Study how we have done things, like scoring and entering the right-and-wrong answer conditions. These concepts will be used again in the next chapter. Use our games as models and try some games of your own. Try Blackjack or some other type of guessing or matching game. Begin with a simple idea and then dress it up some. When you are ready, we will move into the exciting world of *ARCADE GAMES*.



Arcade-Type Game Programs 10

Creating arcade games can be a lot of fun, and can be extremely frustrating if you are not careful. The best advice we can offer is to begin with a very simple game. Then, take the game apart one piece at a time and build your game—step by step. That is exactly the way we did the games in this chapter. Don't get too fancy at first. If you do, you will get extremely frustrated.

Let's try a very simple arcade game first.



RACE

In this game, you are in the white race car that moves from left to right across the screen. There are three other cars that you must avoid. The P key will move your car up the screen. The L key will move the car down the screen. The O key moves the car in a straight line across the screen. Be sure to press the LOCK key before you start. We used the ASCII codes for uppercase letters, so if you don't use the LOCK key (or the shift key), nothing will happen. Enter the program and then read through the description. This program has all the elements of an arcade game. It has graphics, animation, color, and a scoring condition.

```
10 REM   Race
50 HOME: GOSUB 6000
100 GR: HOME
110 y = 10: b = 20: c = 30
115 q =  20
120 x = 34: a = 30: p = 2: d = 34
```

```
150 COLOR = 1: GOSUB 1000
170 COLOR = 3: GOSUB 2000
190 COLOR = 8: GOSUB 3000
191 COLOR = 15: GOSUB 4000
192 IF (p+2 = x OR p-2 = x OR p+4 = x OR p-4 = x)
    AND  (q = y OR q-2 = y OR q+2 = y) THEN 5000
196 IF (p+2 = a OR p-2 = a OR p = a OR p+4 = a OR
    p-4  = a) AND (q = b OR q-2 = b OR q+2 = b)
    THEN 5000
198 IF (p+2 = d OR p-2 = d OR p+4 = d OR p-4 = d)
    AND  (q = c OR q-2 = c OR q+2 = c) THEN 5000
205 GET a$
207 COLOR = 0: GOSUB 1000: GOSUB 2000: GOSUB 3000
    :  GOSUB 4000
210 IF a$ = "P" OR a$ = "p" THEN q = q-2
220 IF a$ = "L" OR a$ = "l" THEN q = q+2
223 IF a$ = "X" OR a$ = "x" THEN 800
225 IF a$ = "O" OR a$ = "o" THEN q = q
226 IF q = 0 OR q > 36 GOTO 5000
230 x = x-4: a = a-2: p = p+2: d = d-2
231 IF x < 3 THEN x = 34
232 IF a < 3 THEN a = 30
234 IF p > 29 THEN p = 2
236 IF d < 3 THEN d = 34
240 GOTO 150
790 REM  Exit game
800 TEXT: HOME: END
999 REM  Draws the top car
1000 HLIN x, x+4 AT y
1010 HLIN x, x+4 AT y+1
1020 PLOT x+1, y-1: PLOT x+3, y-1
1030 PLOT x+1, y+2: PLOT x+3, y+2
1040 RETURN
1999 REM  Draws the middle car
2000 HLIN a, a+4 AT b
2010 HLIN a, a+4 AT b+1
2020 PLOT a+1, b-1: PLOT a+3, b-1
2030 PLOT a+1, b+2: PLOT a+3, b+2
2040 RETURN
2999 REM  Draws the bottom car
3000 HLIN d, d+4 AT c
3010 HLIN d, d+4 AT c+1
3020 PLOT d+1, c-1: PLOT d+3, c-1
3030 PLOT d+1, c+2: PLOT d+3, c+2
3040 RETURN
3999 REM  Draws the white car
```

```
4000 HLIN p, p+4 AT q
4010 HLIN p, p+4 AT q+1
4020 PLOT p+1, q-1: PLOT p+3, q-1
4030 PLOT p+1, q+2: PLOT p+3, q+2
4040 RETURN
4999 REM  Crashes
5000 FOR i = 1 TO 5: PRINT CHR$(7): NEXT i
5040 PRINT "YOU CRASHED!"
5050 END
6000 TEXT
6005 HTAB 5: PRINT "START YOUR ENGINES": PRINT
6010 PRINT: HTAB 5: PRINT "PRESS P TO MOVE UP":
    PRINT
6020 PRINT: HTAB 5: PRINT "PRESS L TO MOVE DOWN":
    PRINT
6030 PRINT: HTAB 5: PRINT "PRESS O TO GO
    STRAIGHT": PRINT
6035 PRINT: HTAB 5: PRINT "PRESS X TO EXIT":
    PRINT: PRINT
6040 HTAB 5: PRINT "IF YOU RUN OFF THE TRACK":
    PRINT
6050 HTAB 5: PRINT "OR HIT ANOTHER CAR": PRINT
6060 HTAB 5: PRINT "YOU CRASH."
6070 FOR i = 1 TO 6000: NEXT i
6080 RETURN
```

—Line 10 names the program.
—Line 50 clears the screen and branches to line 6000 where the directions and the beginning prompts are displayed.
—Line 100 activates the low-resolution mode and clears the screen.
—Lines 110 to 115 state the row values for all of the cars.
—Lines 120 to 140 state the column values for the cars. Note that three cars will move from right to left and one car will move from left to right.
—Lines 150 to 198 plot the cars and check for collisions.
—Line 205 allows you to press a key without interrupting the program.
—Line 210 states that if we press the P key, q (the column location) decreases by 2. This moves the car up.
—Line 220 states that if we press the L key, q (the column location) increases by 2. This moves the car down.
—Line 223 says that we can exit the game by pressing the X key.
—Line 225 states that if we press the O key, the value of q will stay constant, so the car moves straight across the screen.

—Line 226 sets the limits for the roadway and branches to line 5000.
—Lines 230 and 240 keep the cars on the screen in a wraparound effect.
—Line 800 exits the game.
—Lines 1000 to 1040 plot Car number 1.
—Lines 2000 to 2040 plot Car number 2.
—Lines 3000 to 3040 plot Car number 3.
—Lines 4000 to 4040 plot the controlled car.
—Lines 5000 to 5050 display the crash condition and play a 5-tone beep.
—Lines 6000 to 6080 display the beginning directions.

The object of *RACE* is to make it across the screen without colliding with one of the other cars. If you study this game, you will find ways to change things around to create an entirely different game.

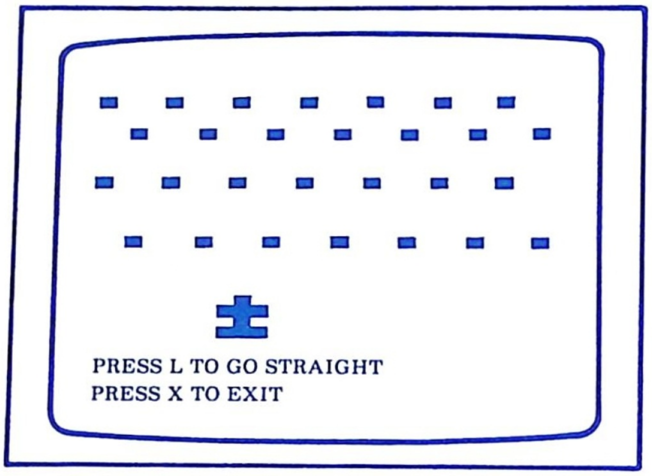
If you like, try building your own program. Here is an idea for you to work on. Plot three or four cars at the left side of the screen. Use a random number generator to pick a winner. Depending on which number the Adam computer selects, the cars can move at different speeds. You might even want to bet on the outcome of the race. If you pick the winner, the Adam computer can reward you. This is how we would organize the program.

1. Name the program.
2. Branch to a subroutine that displays the directions.
3. Plot all three cars at the starting position.
4. Have the Adam computer select a random number between one and three.
5. Insert an imput command to allow you to select a winner.
6. Put in three IF/THEN statements that branch to three different subroutines that plot the cars at different speeds.
7. Add another IF/THEN statement that evaluates your input selection versus Adam's selected random number. If they are equal, you should branch to a "win" condition.

Try working through this outline and you will have an original game.

POTHOLE DERBY

The purpose of this game is to drive your car to the top of the screen without hitting a pothole or running off the road. To move the car up the screen, press the L key. You can turn left by pressing the P key and right by



pressing the O key. Be sure that you press the LOCK key before you begin. Again, we have used the ASCII numbers for uppercase letters.

You begin with 50 points. When you hit a pothole, the Adam computer subtracts 10 points from your score.

The complete program is listed later, but here is how we started. First, we put the car on the screen so that we could control the left, right, and upward direction.

```
100 HOME
110 GR
120 s = 50
130 x = 20: y = 35
140 COLOR = 11: GOSUB 2000
145 COLOR = 15: GOSUB 1000
160 IF y < 4 THEN COLOR = 0: GOSUB 2000: GOTO 130
170 IF x < 3 OR x > 33 GOTO 3000
180 GET a$
185 COLOR = 0: GOSUB 2000
186 y = y-1
190 IF a$ = "X" OR a$ = "x" THEN TEXT: HOME: END
200 IF a$ = "O" OR a$ = "o" THEN x = x-1
210 IF a$ = "P" OR a$ = "p" THEN x = x+1
220 IF a$ = "L" OR a$ = "l" THEN y = y-1
2000 PLOT x, y: PLOT x, y-1: PLOT x, y-2: PLOT x,
   y-3
2010 PLOT x-1, y-2: PLOT x+1, y-2
2020 PLOT x-1, y: PLOT x+1, y
2030 RETURN
```

—Line 100 clears the screen.
—Line 110 activates the low-resolution mode.
—Line 120 states the initial value of s (the score).
—Line 130 states the initial row and column coordinates as Column 20, Row 35.
—Line 140 sets the color to light red and branches to line 2000 where the car is plotted.
—Line 145 sets color to white and plots the potholes.
—Lines 160 to 170 keep the car on the screen in a wraparound style.
—Line 185 waits for a key to be pressed.
—Line 186 moves the car up the screen by subtracting one (1) from the last row number (y).
—Line 190 states that if the X key is pressed, the program stops.
—Line 200 says that if O is pressed, the car will move to the left (x = x − 1).
—Line 210 states that if P is pressed, the car will move to the right (x = x + 1).
—Line 220 states that if L is pressed, the car will move up the screen (y = y − 1).
—Lines 2000 to 2030 plot the car.

The preceding program can be thought of as a subroutine for displaying and controlling a graphics character. If you change the following lines, the car will move down the screen instead of up. Change the lines so they look like this.

```
130 x = 20: y = 1
160 IF y > 34 THEN COLOR = 0: GOSUB 2000: GOTO 130
186 y = y+1
220 IF a$ = "L" OR a$ = "l" THEN y = y+1
```

These changes can give a whole new dimension to the game as you will see later in the *Asteroid Shower* game which we will be discussing later in the chapter.

Once you have the car in place, you will need to add the potholes and the road. This we did (see the complete program for details) and, then, we finally added a scoring subroutine. Here is the entire program listing.

```
 5Ø REM  Potholes
 1ØØ HOME
 1Ø5 GOSUB 4ØØØ
 11Ø GR
 12Ø s = 5Ø
 13Ø x = 2Ø: y = 35
 14Ø COLOR = 11: GOSUB 2ØØØ
 145 COLOR = 15: GOSUB 1ØØØ
 16Ø IF y < 4 THEN COLOR = Ø: GOSUB 2ØØØ: GOTO 13Ø
 17Ø IF x < 3 OR x > 33 GOTO 3ØØØ
 18Ø GET a$
 185 COLOR = Ø: GOSUB 2ØØØ
 186 y = y-1
 19Ø IF a$ = "X" OR a$ = "x" THEN TEXT: HOME: END
 2ØØ IF a$ = "O" OR a$ = "o" THEN x = x-1
 21Ø IF a$ = "P" OR a$ = "p" THEN x = x+1
 22Ø IF a$ = "L" OR a$ = "l" THEN y = y-1
 221 IF y >= 2Ø THEN row = 2Ø: SC = 8: GOTO 23Ø
 222 IF y < 2Ø AND y >= 14 THEN row = 14: SC =5:
     GOTO  23Ø
 224 IF y < 14 AND y >= 9 THEN row = 9: sc = 8:
     GOTO  23Ø
 225 row = 6: sc = 5
 23Ø IF (y-3 = row) AND (x = sc OR x = sc+5 OR x =
     sc+1Ø OR x = sc+15 OR x = sc+2Ø OR x = sc+25
     OR x  = sc+3Ø) THEN 3ØØØ
 232 IF (y-2 = row) AND (x-1 = sc OR x-1 = sc+5 OR
     x-1  = sc+1Ø OR x-1 = sc+15 OR x-1 = sc+2Ø OR
     x-1 =  sc+25 OR x-1 = sc+3Ø) THEN 3ØØØ
 234 IF (y-2 = row) AND (x+1 = sc OR  x+1 = sc+5
     OR x+1  = sc+1Ø OR x+1 = sc+15 OR x+1 = sc+2Ø
     OR x+1 =  sc+25 OR x+1 = sc+3Ø) THEN 3ØØØ
 236 IF (y-2 = row) AND (x = sc OR x = sc+5 OR x =
     sc+  1Ø  OR x = sc+15 OR x = sc+2Ø OR x =
     sc+25 OR x =  sc+3Ø) THEN 3ØØØ
 238 IF (y = row) AND (x-1 = sc OR x-1 = sc+5 OR x-
     1 =  sc+1Ø OR x-1 = sc+15 OR x-1 = sc+2Ø OR x-
     1 = sc+25  OR x-1 = sc+3Ø) THEN 3ØØØ
 24Ø IF (y = row) AND (x+1 = sc OR x+1 = sc+5 OR
     x+1 =  sc+1Ø OR x+1 = sc+15 OR x+1 = sc+2Ø OR
     x+1 = sc+25  OR x+1 = sc+3Ø) THEN 3ØØØ
 25Ø IF (y = row) AND (x = sc OR x = sc+5 OR x =
     sc+1Ø  OR x = sc+15 OR x = sc+2Ø OR x = sc+25
     OR x =  sc+3Ø) THEN 3ØØØ
 31Ø GOTO 14Ø
```

```
999 REM  Set up field of potholes
1000 a = 20: b = 14: c = 9: d = 6
1010 PLOT 8, a: PLOT 13, a: PLOT 18, a: PLOT 23,
    a:  PLOT 28, a: PLOT 33, a
1020 PLOT 38, a: PLOT 5, b: PLOT 10, b: PLOT 15,
    b:  PLOT 20, b: PLOT 25, b: PLOT 30, b: PLOT
    35, b
1030 PLOT 8, c: PLOT 13, c: PLOT 18, c: PLOT 23,
    c:  PLOT 28, c: PLOT 33, c: PLOT 38, c
1040 PLOT 5, d: PLOT 10, d: PLOT 15, d: PLOT 20,
    d:  PLOT 25, d: PLOT 30, d: PLOT 35, d
1150 RETURN
1999 REM  Draws the car
2000 PLOT x, y: PLOT x, y-1: PLOT x, y-2: PLOT x,
    y-3
2010 PLOT x-1, y-2: PLOT x+1, y-2
2020 PLOT x-1, y: PLOT x+1, y
2030 RETURN
3000 s = s-10
3004 COLOR = 11: GOSUB 2000
3005 IF s <= 0 THEN PRINT "YOU ARE OUT OF POINTS!"
3010 FOR i = 1 TO 5: PRINT CHR$(7): NEXT i
3020 PRINT "YOU CRASHED", "SCORE "; s
3025 IF s <= 0 THEN END
3026 COLOR = 0: GOSUB 2000
3030 GOTO 140
4000 TEXT
4005 HTAB 10: PRINT "POTHOLE DERBY": PRINT: PRINT
4010 HTAB 5: PRINT "YOU BEGIN WITH 50 POINTS.":
    PRINT
4020 HTAB 5: PRINT "IF YOU HIT A POTHOLE,": PRINT
4030 HTAB 5: PRINT "YOU LOSE 10 POINTS.": PRINT
4040 HTAB 5: PRINT "PRESS P TO MOVE RIGHT": PRINT
4050 HTAB 5: PRINT "PRESS O TO MOVE LEFT": PRINT
4060 HTAB 5: PRINT "PRESS L TO GO STRAIGHT": PRINT
4065 HTAB 5: PRINT "PRESS X TO EXIT"
4070 FOR i = 1 TO 4000: NEXT i
4080 RETURN
```

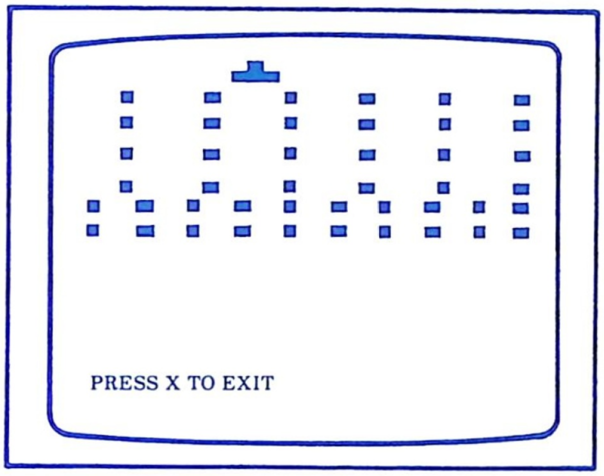
—Line 50 names the program.
—Line 100 clears the screen.
—Line 105 branches to line 4000 and displays the beginning directions and prompts.
—Line 110 activates the low-resolution mode.

—Line 120 states the initial value of s (the score).
—Line 130 states the initial value of the row,column coordinates as Row 35 (for y), Column 20 (for x).
—Line 140 sets the color to light red and branches to line 2000 where the car is plotted.
—Line 145 sets the color to white and branches to line 1000 to display the potholes.
—Line 160 sends the car back to the bottom of the screen when it reaches Row 4.
—Line 170 states the crash condition if you run off the screen or road.
—Line 180 allows you to input a key response into the program.
—Line 185 erases the car before moving it.
—Line 186 moves the car up the screen.
—Line 190 stops the program if the X key is pressed.
—Line 200 moves the car to the left if the O key is pressed.
—Line 210 moves the car to the right if the P key is pressed.
—Line 220 moves the car up the screen if the L key is pressed.
—Lines 221 to 310 define the pothole "hit" conditions and branch to line 3000 if a pothole is hit. If the x, y coordinates of the car equal the coordinates of any of the potholes, the crash condition will occur.
—Line 310 loops to line 140 so that the car can be erased and eventually replotted in a new location.
—Lines 1000 to 1150 plot the potholes.
—Lines 2000 to 2030 plot the car.
—Lines 3000 to 3030 beep the speaker five times, print the crash prompt, subtract ten points from your score, and print the score.
—Lines 4000 to 4080 display the beginning screen prompts and the directions for the game.

By pressing either the Q or P key, you can move the car left or right, respectively. You can plot more potholes, if you like, which will increase the difficulty level of the game. You could also plot more cars to create some traffic. Experiment with your own enhancements. Now, let's use the same design to create an entirely different game.

ASTEROIDS

In this game, you are the saucer. Your mission is to destroy as many asteroids as you can. Every time you hit an asteroid, you get ten points.



Watch your landing, however. There is a nasty Black Hole at the bottom of the screen. If you land to the right of Column 16, the Black Hole gets you and you lose all your points.

```
50 REM Asteroids
100 HOME: GOSUB 3000
110 GR: s = 0: x = 19: y = 4
120 a = 5: b = 8: c = 12: d = 16: e = 18: f = 21
140 COLOR = 3: GOSUB 1000: COLOR = 7: GOSUB 2000
160 IF y >= 38 AND x >= 22 THEN 5000
170 IF x <=1 OR x >= 38 THEN 5000
175 IF y >= 38 AND x < 22 THEN COLOR = 0: GOSUB
    2000:  y = 2
180 GET a$
185 COLOR = 0: GOSUB 2000
186 y = y + 1
190 IF a$ = "X" OR a$ = "x" THEN HOME: END
200 IF a$ = "O" OR a$ = "o" THEN x = x - 1
210 IF a$ = "P" OR a$ = "p" THEN x = x + 1
220 IF a$ = "L" OR a$ = "l" THEN y = y + 1
240 IF (y = a OR y = b OR y = c OR y = d) AND (x
    = 5  OR x = 10 OR x = 15 OR x = 20 OR x = 25
    OR x = 30)  THEN 6000
280 IF (y = e OR y = f) AND (x = 3 OR x = 6 OR x
    = 9  OR x = 12 OR x = 15 OR x = 18 OR x = 21
    OR x = 24)  THEN 6000
290 IF (y = e OR y = f) AND (x = 27 OR x = 30)
    THEN  6000
```

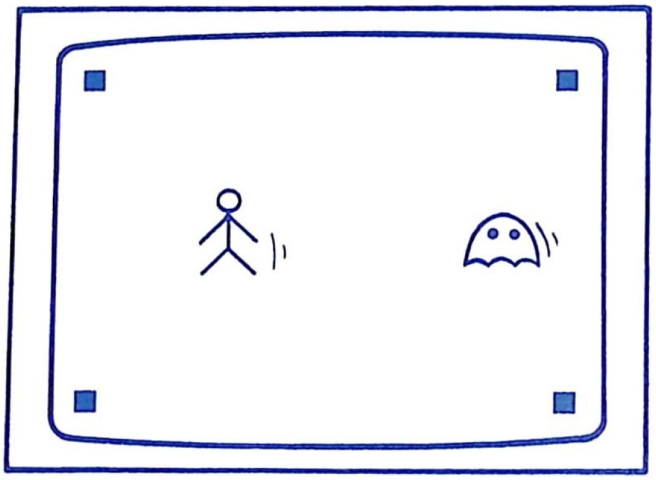
```
320 COLOR = 7: GOSUB 2000
330 GOTO 140
1000 REM  Plots Asteroids
1010 FOR i = 5 TO 30 STEP 5
1020 PLOT i,5
1030 PLOT i,8: PLOT i,12: PLOT i,16
1040 NEXT i
1050 FOR i = 3 TO 30 STEP 3
1060 PLOT i,18: PLOT i,21
1070 NEXT i
1080 RETURN
2000 REM  Saucer
2010 PLOT x,y-1
2020 HLIN x-1, x+1 AT y
2040 RETURN
2999 REM  Instructions
3000 TEXT
3005 HTAB 5: PRINT "ASTEROID SHOWER": PRINT:PRINT
3010 HTAB 5: PRINT "YOU GET 10 POINTS FOR EACH":
    PRINT
3020 HTAB 5: PRINT "ASTEROID YOU HIT": PRINT
3030 HTAB 5: PRINT "IF YOU ARE NOT CAREFUL THE":
    PRINT
3040 HTAB 5: PRINT "BLACKHOLE WILL GET YOU.":
    PRINT
3050 FOR i = 1 TO 6000: NEXT i
3060 HTAB 5: PRINT "PRESS P TO MOVE RIGHT": PRINT
3070 HTAB 5: PRINT "PRESS O TO MOVE LEFT": PRINT
3080 HTAB 5: PRINT "PRESS L TO MOVE STRAIGHT":
    PRINT
3085 HTAB 5: PRINT "PRESS X TO EXIT": PRINT
3090 FOR i = 1 TO 4000: NEXT i
3100 RETURN
4999 REM  Black Hole Got You
5000 FOR i = 1 TO 10: PRINT CHR$(7): NEXT i
5005 PRINT "THE BLACK HOLE GOT YOU !!!"
5010 s = 0: END
5999 REM  Hit An Asteroid
6000 s = s + 10
6005 PRINT CHR$(7)
6010 PRINT "SCORE ";s
6030 GOTO 140
```

—Line 50 names the program.
—Line 100 clears the screen and branches to line 3000 where the directions and beginning prompts are displayed.

- —Line 110 activates the low-resolution mode and sets the initial value of s (the score) at 0. It states the initial row,column coordinates as Row 4, Column 19.
- —Line 120 defines the asteroid "hit" conditions.
- —Line 140 sets the color to dark red and branches to line 2000 to display the asteroids.
- —Lines 160 to 175 check for the "black hole." If the row number equals 35 and the column number is greater than 22, the program branches to line 5000. If you navigate off the screen, the "black hole" gets you (lines 170 and 175).
- —Line 180 allows you to input a key value directly into the program.
- —Line 185 erases the saucer before moving it.
- —Line 186 moves the saucer down the screen.
- —Line 190 states that if the X key is pressed, the program stops.
- —Line 200 says that if the O key is'pressed, the saucer moves to the left.
- —Line 210 states that if the P key is pressed, the saucer moves to the right.
- —Line 220 states that if the L key is pressed, the saucer moves down the screen.
- —Lines 240 to 290 define the asteroid "hit" conditions and branch to line 6000 if an asteroid is hit. If the x, y coordinate of the saucer equals the coordinates of any of the asteroids, the "score" condition will occur.
- —Line 320 sets the color to light blue and branches to line 2000 where the saucer is plotted.
- —Line 330 loops back to line 140 where the saucer is replotted at another location.
- —Lines 1000 to 1080 plot the asteroids.
- —Lines 2000 to 2040 plot the saucer.
- —Lines 3000 to 3100 display the beginning directions and the screen prompts.
- —Lines 5000 to 5010 display the Black Hole prompt and stop the program. The score goes back to zero.
- —Lines 6000 to 6030 sound a beep, add 10 to your score, print your score, and loop back to line 140.

CHASE

Our last arcade-type game is called *CHASE*. You are the stick figure at the center of the screen. Your mission is to reach and touch four blue blocks on the screen. Sounds easy? Well it would be if a nasty little monster would leave you alone.



The monster will chase you. If he gets you, you lose. But, if you get to a blue box before he gets you, you get ten points and you send the monster back to his home location. This game demonstrates the arcade qualities of the high-resolution mode. In this game, you control all four directions. Press W to go up, Z to go down, S to go left, and A to go right. Input the following program.

```
100 REM   Chase
110 HOME: GOSUB 4000
115 HGR
120 s = 0
130 x = 128: y = 80: a = 192: b = 80
140 HCOLOR = 14: GOSUB 5000: HCOLOR = 3: GOSUB
    1000
150 GET a$
155 HCOLOR = 0: GOSUB 1000
160 IF a$ = "W" OR a$ = "w" THEN GOSUB 610
170 IF a$ = "Z" OR a$ = "z" THEN GOSUB 710
180 IF a$ = "S" OR a$ = "s" THEN GOSUB 810
190 IF a$ = "A" OR a$ = "a" THEN GOSUB 910
195 HCOLOR = 3: GOSUB 1000
198 IF f2 = 1 THEN f2 = 0: HCOLOR = 14: GOSUB
    5000
200 IF x = a AND y = b GOTO 2010
```

```
210 IF (x < 3Ø AND x > 2Ø) AND ((y < 3Ø AND y >
    2Ø) OR  (y < 13Ø AND y > 12Ø)) THEN 3ØØØ
23Ø IF (x < 23Ø AND x > 22Ø) AND ((y < 3Ø AND y >
    2Ø)  OR (y < 13Ø AND y > 12Ø)) THEN 3ØØØ
24Ø IF (b > 15Ø OR y > 15Ø OR b < 2Ø OR y < 2Ø OR
    a <  16 OR x < 16 OR a > 224 OR x > 224) THEN
    fl = 1
245 IF fl = 1 THEN HCOLOR = Ø: GOSUB 1ØØØ
25Ø IF b > 15Ø THEN b = 14Ø
252 IF y > 15Ø THEN y = 14Ø
26Ø IF b < 2Ø THEN b = 2Ø
262 IF y < 2Ø THEN y = 2Ø
27Ø IF a < 16 THEN a = 16
272 IF x < 16 THEN x = 16
28Ø IF x > 224 THEN x = 224
282 IF a > 224 THEN a = 224
29Ø IF fl = 1 THEN fl = Ø: HCOLOR = 3: GOSUB 1ØØØ
3ØØ GOTO 15Ø
6ØØ REM  Move Up
61Ø y = y-8
62Ø b = b-16
63Ø RETURN
7ØØ REM  Move Down
71Ø y = y+8
72Ø b = b+16
73Ø RETURN
8ØØ REM  Move Right
81Ø x = x+8
82Ø a = a+16
83Ø RETURN
9ØØ REM  Move Left
91Ø x = x-8
92Ø a = a-16
93Ø RETURN
1ØØØ REM  Plots Man
1Ø1Ø HPLOT x, y-3 TO x, y+1
1Ø2Ø HPLOT x+3, y TO x-3, y
1Ø3Ø HPLOT x+1, y+2 TO x+1, y+4
1Ø4Ø HPLOT x-1, y+2 TO x-1, y+4
1Ø5Ø HPLOT x+1, y-3 TO x+1, y-2
1Ø6Ø HPLOT x-1, y-3 TO x-1, y-2
1Ø7Ø REM  Plots Monster
1Ø8Ø HPLOT a+3, b TO a, b+3
1Ø9Ø HPLOT a+4, b TO a+7, b+3
11ØØ HPLOT a, b+4 TO a,  b+7
111Ø HPLOT a+7, b+4 TO a+7, b+7
```

```
1120 HPLOT a+1, b+7 TO a+6, b+7
1130 RETURN
2000 REM  Dead Condition
2010 HOME
2020 FOR i = 1 TO 10: PRINT CHR$(7): NEXT i
2030 PRINT "YOU ARE DEAD"
2050 END
2999 REM   Score
3000 s = s+10
3015 f2 = 1
3017 PRINT CHR$(7)
3020 PRINT "SCORE "; s
3025 HCOLOR = 0: GOSUB 1080
3030 a = 192: b = 80
3040 GOTO 150
3999 REM  Instructions
4000 HTAB 10: PRINT "CHASE"
4010 PRINT: PRINT
4020 HTAB 5: PRINT "GUIDE YOUR MAN AROUND": PRINT
4030 HTAB 5: PRINT "THE SCREEN": PRINT
4040 HTAB 5: PRINT "AND TRY TO TOUCH THE BOXES.":
     PRINT
4050 HTAB 5: PRINT "EACH TIME YOU TOUCH A BOX,":
     PRINT
4060 HTAB 5: PRINT "YOU SCORE 10 POINTS AND":
     PRINT
4070 HTAB 5: PRINT "SEND THE MONSTER HOME.": PRINT
4080 HTAB 5: PRINT "IF THE MONSTER GETS YOU,":
     PRINT
4090 HTAB 5: PRINT "THE GAME IS OVER."
4100 FOR i = 1 TO 6000: NEXT i
4110 HOME
4120 HTAB 5: PRINT "HERE IS HOW YOU MOVE": PRINT
4130 HTAB 8: PRINT "UP", "PRESS W": PRINT
4140 HTAB 8: PRINT "DOWN", "PRESS Z": PRINT
4150 HTAB 8: PRINT "LEFT", "PRESS A": PRINT
4160 HTAB 8: PRINT "RIGHT", "PRESS S": PRINT
4170 HTAB 12: PRINT "HAVE FUN!"
4180 FOR i = 1 TO 6000: NEXT i
4190 FOR i = 1 TO 20: PRINT CHR$(7); : NEXT i:
     PRINT
4200 RETURN
4999 REM Plots Squares
5000 FOR q = 20 TO 30: FOR r = 20 TO 30
5020 HPLOT q, r: NEXT r: NEXT q
```

```
5Ø3Ø FOR q = 22Ø TO 23Ø: FOR r = 2Ø TO 3Ø: HPLOT
    q, r:  NEXT r: NEXT q
5Ø4Ø FOR q = 2Ø TO 3Ø: FOR r = 12Ø TO 13Ø: HPLOT
    q, r:  NEXT r: NEXT q
5Ø5Ø FOR q = 22Ø TO 23Ø: FOR r = 12Ø TO 13Ø:
    HPLOT q,  r: NEXT r: NEXT q
5Ø9Ø RETURN
```

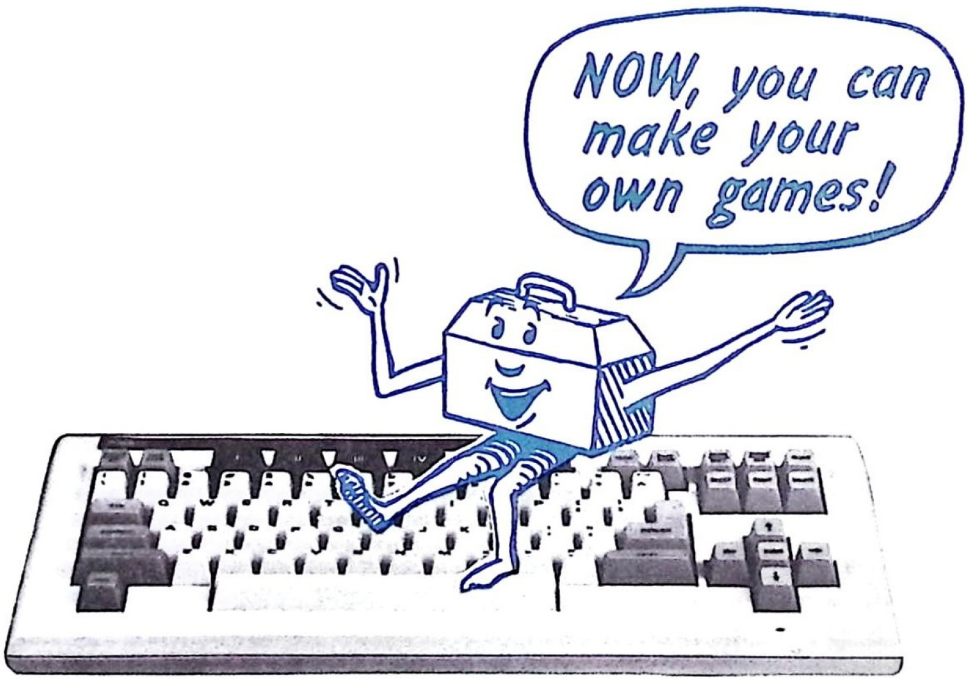
—Line 100 names the program.
—Line 110 clears the screen, branches to line 4000, and plots the beginning screen prompts.
—Line 115 activates the high-resolution mode.
—Line 120 sets the initial value of s (the score) as zero.
—Line 130 sets the initial values of x, y, a, and b. These are the initial row,column coordinates of both the stick figure and the monster.
—Line 140 sets the color to cyan and branches to line 5000 to display the 4 squares. It then sets the color to dark red and draws the man.
—Line 150 allows you to enter a keystroke into the program.
—Line 155 erases the man before moving him.
—Line 160 branches to line 610 when the W key is pressed. Both the monster and the man will move up the screen.
—Line 170 branches to line 710 when the Z key is pressed. Both the man and the monster move down the screen.
—Line 180 branches to line 810 when the S key is pressed. The man moves to the right and the monster moves left.
—Line 190 branches to line 910 when the A key is pressed. The man and the monster move to the left.
—Line 195 redraws the man in a new position.
—Line 198 redraws the 4 squares, but only after the man has landed on one.
—Line 200 states that if the monster and the man are in the same place, the program branches to line 2010.
—Lines 210 to 240 are the "score" conditions. When the man touches one of the blue boxes, the program branches to line 3000.
—Line 245 erases the man before moving him after a score.
—Lines 250 to 282 reset the values of x, y, a, and b to keep the man and the monster on the screen.
—Line 290 redraws the man after moving him.
—Line 300 loops to line 150 and waits for another key to be pressed.
—Lines 600 to 630 move the man up 8 locations and move the monster up 16 locations.

—Lines 700 to 730 move the man down 8 locations and move the monster down 16 locations.
—Lines 800 to 830 move the man to the right 8 locations and the monster 16 locations to the right.
—Lines 900 to 930 move the man to the left 8 locations and the monster 16 locations to the left.
—Lines 1000 to 1130 plot the stick figure and the monster.
—Lines 2000 to 2050 print the "DEAD" condition if the monster gets you.
—Lines 3000 to 3040 calculate and print the score if you touch one of the blue blocks.
—Lines 4000 to 4200 print the beginning screen prompts and directions.
—Lines 5000 to 5090 plot the 4 blue squares.

We have given the machine some intelligence with this program. The monster seems to chase the man. This technique, greatly enhanced, has given all of us many hours of arcade-type excitement.

Now that we have given you some ideas, we hope that you will be able to design your own games. Begin with any simple idea and build your game one step at a time.

Have fun and good luck!



Index

SAMS

The Tool Kit Series: Coleco Adam™ Edition

Get the "tools" you need to program your Adam computer quickly and easily! The "tools" are short 5 to 30-line subroutines that combine color, sound and graphics to form a variety of educational programs and computer games.

By studying the subroutines included, you'll learn how to:

- Read programs in terms of their working parts
- Write simple, efficient programs using subroutines
- Use color, sound, graphics and animation to create entertaining games and quizzes

So why spend long frustrating hours reading complex computer books and manuals? Follow the Tool Kit approach to programming. It's fast and fun for beginners of all ages!

Ted Buchholz is a graduate of the University of South Carolina and currently works as an editor with a publishing company in the Washington, D.C. area. He enjoys creative computer play with his children and their friends.

Dave Dusthimer is an avid home computer user and a self-taught programmer. As the father of three, Dave frequently teaches his children, as well as several neighborhood children, how to use and enjoy computers. A member of IEEE, Dave has edited and developed approximately 200 technical books over the past five years.

Ted and Dave have worked as co-authors on every book in the Sams Tool Kit Series: *Tool Kit Series: Vic-20™ Edition, Tool Kit Series: Commodore 64™ Edition, Tool Kit Series: TI-99/4A™ Edition.*

Howard W. Sams & Co., Inc.
A Publishing Subsidiary of ITT
4300 West 62nd Street, Indianapolis, Indiana 46268 USA

[redacted]

ISBN: 0-672-22312-0