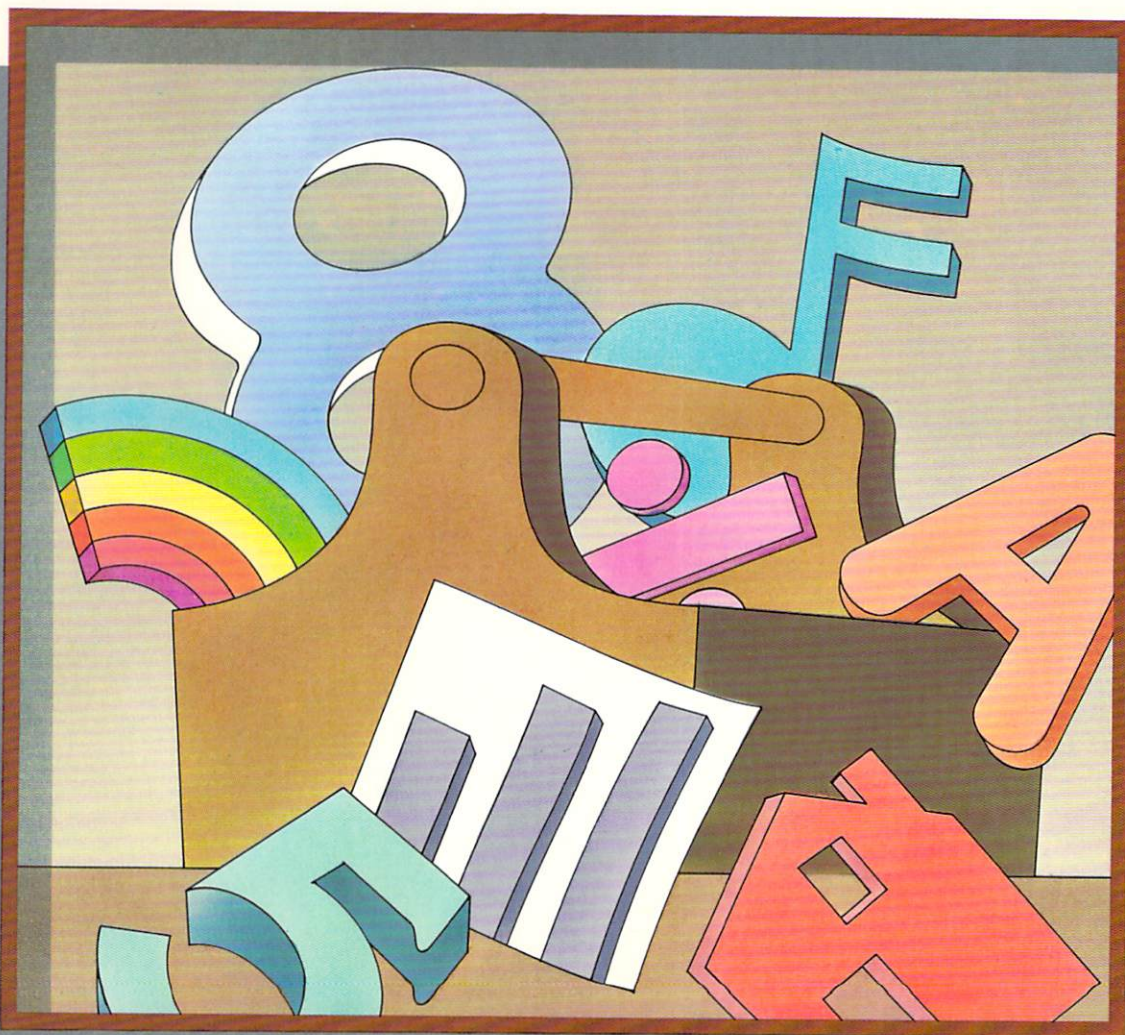


The Tool Kit Series Commodore 64™ Edition

Ted Buchholz and Dave Dusthimer



THE TOOL KIT SERIES
Commodore 64 Edition



Ted Buchholz, a graduate of the University of South Carolina, is an editor with a publishing company in the Washington, D.C. area. He enjoys creative computer play with his children and their friends.

David Dusthimer is an avid home computer user and a self-taught programmer. As the father of three, Dave found himself teaching 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.

Other SAMS books by Ted and Dave are *The Tool Kit Series: VIC 20 Edition* (#22309) and *The Tool Kit Series: TI-99/4A Edition* (#22310).

The Tool Kit Series Commodore 64 Edition

**by
Ted Buchholz and Dave Dusthimer**

Howard W. Sams & Co., Inc.
4300 WEST 62ND ST. INDIANAPOLIS, INDIANA 46268 USA

Copyright © 1984 by Ted Buchholz and Dave Dusthimer

FIRST EDITION
FIRST PRINTING—1984

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. While every precaution has been taken in the preparation of this book, the publisher assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

International Standard Book Number: 0-672-22314-7
Library of Congress Catalog Card Number: 84-50651

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 Commodore 64 is no exception. In this book, you will find all the tools you need to make your microcomputer 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 learning 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 to learn programming 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. Soon we 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 15-line subroutines. Color, sound and music, 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 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 that works well for the average microcomputer owner.

This book can 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 will give you, the user, quicker and more satisfying results than traditional approaches. Since the subroutines are mostly educational and recreational, they will appeal to both parents and children.

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

TED BUCHHOLZ
AND DAVE DUSTHIMER

This book is dedicated to Weldon.

Contents

CHAPTER 1

The Commodore 64 and How It Works.....	9
What Makes the Commodore 64 Work?—How Much Memory Does the Commodore 64 Have?—The Binary System—How Do You Write Programs in BASIC?—What Is the Best Way To Start Learning How To Use the Commodore 64?—Using the Commodore Graphics Characters—Using the Color Keys and the Reverse Function—Editing on the Commodore 64—Using the Commodore Direct Mode	

CHAPTER 2

The Building Blocks of Commodore BASIC.....	17
Commodore BASIC—Commands and Statements—Functions—Inputting Procedures and Hints—Variables	

CHAPTER 3

Commodore 64 Color.....	29
Keyboard Coloring—Screens and Borders—Color in PRINT Statements—Color Subroutines—Color Choice—Random Color	

CHAPTER 4

Sound and Music Subroutines.....	41
POKEing Sound—Sound Variables—The Noise Maker—Sound Encyclopedia—POKEing More Than One Voice—Music Lesson Subroutines—Playing Chords on the Commodore 64—The Commodore 64 Organ—Commodore Song Book—Sound Subroutines Using DATA Files—Arcade-type Sounds—Putting Sound and Color Together	

CHAPTER 5

Graphics Subroutines.....	59
Creating Graphics With the PRINT Command—Creating Graphics Using POKE—Character Codes—Bar Charts or Graphs—Pictures With the Commodore 64	

CHAPTER 6

Animation Subroutines.....	75
How Does the Computer Animate?—Animation Using the PRINT Command—Using Strings in Animation—Animation Using POKE	

CHAPTER 7

Calculating Subroutines: Facts, Figures, and Conversions.....	89
Simple Calculations—Calculating Subroutines—Answers to Chapter Problems	

CHAPTER 8

Educational Games.....	101
Arithmetic Tester—Multiplication Tables—Shape Recognition Game—Rhyming Recognition Game—Riddle fractions—Painter’s Palette—States and Capitals	

CHAPTER 9

Traditional Games.....	119
Guessing Game—Tic-Tac-Toe—Ted and Dave’s Casino—Hangman	

CHAPTER 10

Arcade Games.....	129
Saucer Blaster—Pothole Derby—Asteroid Shower—Rat Trap	

APPENDIX A

Music Notes.....	140
------------------	-----

APPENDIX B

Screen Values for POKE.....	141
-----------------------------	-----

APPENDIX C

Memory Maps.....	143
Screen character Memory Map—Color Code Memory Map	

APPENDIX D

Sample Memory Map Graph Paper.....	145
------------------------------------	-----

APPENDIX E

ASCII and CHR\$ Codes.....	147
----------------------------	-----

APPENDIX F

Common Error Messages.....	149
----------------------------	-----

APPENDIX G

Saving and Loading Programs.....	151
Using the Datasette Recorder—Using the Disk Drive	

APPENDIX H

PEEK Values.....	153
Index.....	154



The Commodore 64 and How It Works

1

10 THE TOOL KIT SERIES Commodore 64 Edition

Computers are the result of genius-level engineering. Fortunately for us "normal" people, you need not be a genius to use the Commodore 64. Since the "64" can't think like humans, we must learn to think as it does. It isn't hard once you understand how the Commodore 64 works.

WHAT MAKES THE COMMODORE 64 WORK?

At the core of every computer is its CPU, or Central Processing Unit, which controls, interprets, and executes the computer functions. The 64's CPU is a 6510 microprocessor. This microprocessor "chip" is very tiny, but it stores a great deal of information and is very powerful.

Computers can only receive information through their CPU in a binary system. That is, computers only know two things — whether data is "on" or "off," "0" or "1," or a "yes" or "no." Each of the digits readable by the computer (0 and 1) is known as a *bit*. The Commodore 64 is an 8-bit machine, partially because each of its characters is composed of 8 bits.

Look at the following bit-pattern illustration for the character "A."

Bits	Image	Bit Pattern (Binary Code)
1	**	00011000
2	* *	00100100
3	* *	01000010
4	*****	01111110
5	* *	01000010
6	* *	01000010
7	* *	01000010
8		00000000

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

HOW MUCH MEMORY DOES THE COMMODORE 64 HAVE?

There are two types of memory in the Commodore 64 — read-only memory (ROM) and random-access memory (RAM). The ROM in the Commodore 64 hardware is programmed by the manufacturer. The CPU controls computer operations by addressing programs located in ROM (read-only mem-

ory). The ROM is permanent in the computer and we, as users, cannot use the ROM for storage. The RAM (random-access memory) is a temporary storage memory that serves as space for the user's programs. The unexpanded Commodore 64 (the standard Commodore 64 without additional memory) has 38.9 kilobytes of memory storage available. When you turn the Commodore 64 on, it will proudly display:

```
***COMMODORE 64 BASIC V2***  
64K RAM SYSTEM    38911 BASIC BYTES FREE
```

There is a way to find out just how much memory is left after you start programming; we'll discuss that in the next chapter. The longer your programs, the more memory or RAM you will need. Unless you save (on a tape or diskette) the data in RAM, the data disappears when the computer is turned off.

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 the binary machine code into something that is more easily understood by humans. The operating system for the Commodore 64 is a BASIC Interpreter, which allows us to communicate with the computer using BASIC as the programming language. We input data principally using the keyboard, but we can also use joysticks and similar devices. Data are inputted using BASIC 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 which are not so different from the grammar and syntax rules in English. We're lucky because BASIC 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 COMMODORE 64?

You can do a lot with just the Commodore 64 and a screen (either a television set or a monitor). However, when you write programs of some length,

12 THE TOOL KIT SERIES Commodore 64 Edition

you will want to save yourself from having to reenter the program every time you want to use it. You can use the Commodore cassette unit to save programs and files on tape or you can purchase the Commodore disk drive to save programs and files on floppy disks. The disk drive turns your Commodore 64 into a much more powerful machine because the drive can manage a lot of information without tapping the RAM available in the Commodore 64.

You can print your programs or results out on paper using a printer to produce this “hard copy.” There are many other accessories available at computer stores and through mail-order houses that will tailor the Commodore 64 to your personal needs.

WHAT IS THE BEST WAY TO START LEARNING HOW TO USE THE COMMODORE 64?

Familiarize yourself with the chief input device for the Commodore 64 — the keyboard. Using the guide that came with the computer, you can learn how to input characters and use the other functioning keys. Let’s review this briefly so that you can refer to this section and use it as a reference for the rest of the book. You cannot hurt the computer or destroy the ROM by playing around on the keyboard.



Fig. 1-1. The Commodore 64 keyboard.

USING THE COMMODORE GRAPHICS CHARACTERS

All the letter keys and some of the symbol keys have graphics characters shown on their front (Fig. 1-2). In order to type the left graphics character, you must press that key and the **⌘** (Commodore) key at the same time. For the right graphics character, use the **SHIFT** key instead of the Commodore key.

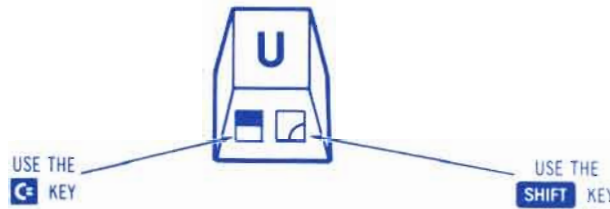


Fig. 1-2. Selecting a graphics character.

USING THE COLOR KEYS AND THE REVERSE FUNCTION

Use the **CTRL** (Control) key together with the number keys to change the color of the cursor and what you are inputting. The control key is also used to activate the **RVS/ON** and **RVS/OFF** keys. By using the reverse mode, you can print the character in the screen color and the background in the cursor color. Using reverse and then changing the colors will make a colorful screen display. Use the space bar while the reverse mode is on to get solid color lines.

EDITING ON THE COMMODORE 64

The Commodore 64 is much easier to use than a regular typewriter because of the features built into the keyboard. The cursor keys can be used to edit any line or character on the screen, whether listed with or without a line number (Fig. 1-3).

PRESS	CRSR			-the cursor moves down
PRESS	SHIFT	or	C + CRSR	-the cursor moves up
PRESS	CRSR			-the cursor moves to the right
PRESS	SHIFT	or	C + CRSR	-the cursor moves to the left

Fig. 1-3. Using the cursor keys.

All of these cursor movements will repeat when the keys are held down continuously. After you edit with the cursor, pressing **RETURN** will enter the changes you have made. Note that if you press **SHIFT** and **RETURN**, this will not make the change you edited. Be cautious when editing on the last screen line as you can stop the program cold. The computer jams and you have to turn the Commodore 64 OFF and then ON to resume operation.

The easiest way to edit is with the **INST/DEL** key. Pressing this key (without using the shift key) simply deletes the character to the left of the cursor. Sometimes we think we use this "delete" key more than any other! Pressing this key together with the shift key will allow you to insert additional characters or spaces anywhere in the text.

14 THE TOOL KIT SERIES Commodore 64 Edition

The **CLR/HOME** key is very useful for editing. When you want to erase everything on the screen, press **SHIFT** (or the Commodore key) and the **CLR/HOME** key. That will rid the screen of all characters and will place the cursor in the top left-hand corner of the screen (home). If you simply want the cursor to "home," use the key unshifted.

When you need to stop a program, press the **RUN/STOP** key. The computer will then show something like:

```
BREAK IN LINE 190
```

telling you where the break occurred in the program when **RUN/STOP** was pressed.

By pressing **SHIFT** and **RUN/STOP**, you have another way of expressing a LOAD command when using a cassette player. It is also faster and easier than typing the command in on the keyboard.

When you press **RUN/STOP** and **RESTORE** at the same time, the computer resets to its original colors and the word "READY" appears. Any programs you have entered are still in memory even though the computer looks like you have just turned it on.

You will erase all programs in memory and start afresh, however, when you type in "NEW" and press **RETURN**. This doesn't clear the screen, but it does forget everything.

USING THE COMMODORE DIRECT MODE

The Commodore 64 is much more than a calculator, but you can use it just like one to solve any problem you wish. You don't need to number the lines like you do in a program. Try it. Type in:

```
PRINT 172 * 4 / 5 (and press RETURN)  
PRINT 60 * (3 - 1) (and press RETURN)
```

You can even have a short program (without line numbers) run in the middle of a regular program that has line numbers. As long as you don't exceed 80 characters with the program (2 screen lines, 40 characters to a line), the short program will be executed as soon as you press **RETURN**. This is helpful when you want to try out a short color or sound routine without disturbing the memory of the program you are working on. For example:

```
10 PRINT "LOUISE AND CHRIS"  
POKE 53280,0:POKE 53281,1 —NOTE: There is no line number.  
20 GOTO 10
```

When you type in the POKE command, and hit **RETURN**, the screen turns white and the border becomes black. When you key in the second line in the

program — GOTO 10 — and then type RUN, the computer prints lots of lines that say LOUISE AND CHRIS.

We hope that knowing something about how the Commodore 64 works as well as knowing how to use the keyboard is good preparation for our next chapter where you gain familiarity with *The Building Blocks of Commodore BASIC*.





The Building Blocks of Commodore BASIC

2

18 THE TOOL KIT SERIES Commodore 64 Edition

Once you learn how the Commodore 64 works and how to speak its language, learning to program is easy. In Chapter 1, we explained what the Commodore 64 can do, and now we are going to teach you to speak COMMODORE BASIC. Remember that the Commodore 64 is a “dumb” machine. It can’t do anything unless you, the user, command it to perform. Before you get carried away with feelings of great power, however, you must also remember that the computer can’t speak English, so you must give your commands in a language that the computer understands.

COMMODORE BASIC

Your Commodore 64 speaks BASIC. You can equip your computer to speak other languages, but this book deals only with standard COMMODORE BASIC.

Your machine has a very limited vocabulary. The Commodore 64 only understands about 70 commands, statements, and functions. In the next few pages, we will explain each command and statement and give you an idea of how it works. You will refer to this part of the book often, so you may want to put some notes in the margin. Let’s learn how to talk to the computer.

COMMANDS AND STATEMENTS

The following are fairly standard BASIC commands and statements. They can be used with most computers that understand BASIC. For convenience, we have alphabetized this listing.

CLR — This erases any variables stored in memory and sets them to zero, but it does not erase the program itself.

CLOSE — This statement simply closes a file that you opened earlier with an OPEN statement. You can close specific files by using a file number following the word CLOSE.

CMD — This is used to display output that usually appears on the screen to a file or a device like a printer. This device has to be OPENed and CLOSEd just like a file.

CONT — For CONTINUE. If you stop a program with a STOP or END statement or the STOP key, you can use CONT to restart the program where you stopped it.

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. You must enclose the parameters in parentheses right after the DEF FN statement.

DIM — DIM lets you set the maximum size of a given array.

END — You guessed it. This statement stops or ends your program. You will most often use this statement to divide subroutines at the end of a program.

FOR/TO/STEP — Loops are easily created using this statement. A 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
```

A STEP command 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, you can just leave the STEP portion of the command out. By using STEP and a negative number, you can also create a negative step. However, you must include a NEXT statement so that the computer will go back and act on the next variable in the FOR/TO/STEP command. For example:

```
60 FOR Y = 15 TO 0 STEP -1  
70 POKE 53281, Y  
80 NEXT Y
```

GET — This is similar to the INPUT statement because data are taken from the keyboard. GET takes the key that is pressed as the value in the variable following the GET statement. We’ll see this used later in the book.

GOSUB — This is one of a couple of branching statements. You must use a RETURN statement to tell the computer to return to the main body of

20 THE TOOL KIT SERIES Commodore 64 Edition

the program. The RETURN statement will send the computer to the line right after the GOSUB statement. For example:

```
40 GOSUB 300
300 POKE 53280,0
310 FOR T = 1 TO 1000: NEXT
320 POKE 53280,6
330 RETURN
```

GOTO — This is another branching statement. Following the GOTO statement and a space, 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.

IF THEN — This is another very useful statement. The condition described after IF must be true in order for the Commodore 64 to perform the statement following THEN. For example, let's consider this program line:

```
30 IF X > 50, THEN 200
```

If X is greater than 50, then the Commodore 64 will GOTO program line 200. If X is less than or equal to 50, then the next line of the program is executed.

INPUT — This statement halts the program and allows you to input a string or character using the keyboard. This statement may be followed by a prompt in quotation marks. Following the last quotation mark, you must type in a semicolon and the name of the variable. For example:

```
100 INPUT "PICK ANY NUMBER", X
300 INPUT X
```

LET — LET allows you to define variables in a program. You don't have to use the LET statement with the Commodore 64. It interprets **LET A = 1** and **A = 1** the same way.

LIST — This command tells the computer to list the program statements in order. It can only be used in the immediate mode. You may also list specific lines of the program by entering the line number after the command. By typing a minus sign in front of the number, the computer will

list all lines up to and including the number. Let's look at how the LIST function works.

- LIST — This lists the entire program.
- LIST 400 — Only line 400 will appear.
- LIST -400 — All lines up to and including 400 will be listed.
- LIST 400- — All lines including and after 400 will be listed.
- LIST 200-400 — Lines 200 through 400 will be listed.

LOAD — This command tells the Commodore 64 to load a stored program from tape or disk. You can use the program name (in quotes) or a string variable to identify the program you want loaded.

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 is used with a FOR/TO statement. 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.

ON GOSUB — This is still another branching statement. If a certain condition exists, the computer will go to the subroutine listed after GOSUB.

```
5 INPUT X
10 ON X GOSUB 35, 100, 200
```

When value X is inputted, the computer will GOSUB line 35, 100, and 200. You must use a RETURN statement just like you do with GOSUB.

ON GOTO — This statement works just like the ON GOSUB statement except that you don't have to use the RETURN statement.

OPEN — OPEN tells a BASIC program to use files stored on memory-storage devices. OPEN provides a necessary link between the storage device and a file number in the program.

22 THE TOOL KIT SERIES Commodore 64 Edition

POKE — This command allows you to input a value into the memory of the Commodore 64. POKEing is mostly used for color, sound, and graphics. For example:

```
10 POKE 53280, 4    POKes a color variable (purple) into mem-
                   4  ory location 53280 (the border color
                   4  address).
20 POKE 54296, 15   POKes a volume (loudness) variable into
                   15  memory location 54296.
```

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. For example:

```
10 PRINT "HALLOWEEN"
20 PRINT X$
```

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.

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

```
400 REM *** SCORING SUBROUTINE
```

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

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.

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 will carry out each program line in sequence, unless you designate a line number where you want the program to start. For example:

```
RUN 190
```

will start the program at line 190.

SAVE — The SAVE command allows you to take a program from the computer's memory and store it on either a cassette tape or a floppy disk. If you are using a cassette recorder for memory storage, simply enter SAVE and you will get a series of prompts that will lead you through the saving process. You can give the program a name, with a name in quotes or with a string variable name, to make location and retrieval easier. You simply SAVE using:

```
SAVE "ARCADE SMASH"
```

or

```
SAVE X$
```

STOP — This statement stops the program and tells you in what line number the break has been made. Use CONT to restart the program.

VERIFY — This checks the program you have just SAVEd with the one in the computer's memory.

FUNCTIONS

There are two types of BASIC functions — *Numeric* and *String*. The arguments of built-in functions are always enclosed in parentheses (). A String function can be identified by having a dollar sign (\$) as the last character of the keyword.

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:

```
ABS(47)
```

—The absolute value is 47, but we know that.

Now try

```
ABS 14*(21*6.1)-20
```

—Now do you see why this command is useful?

24 THE TOOL KIT SERIES Commodore 64 Edition

ASC(X\$) — Each character on the Commodore 64 has an ASCII code number. The ASC function will give you the code number for a string variable or any group of numbers that appear in parentheses. Here's how it works.

```
5 A$ = "D"  
10 PRINT ASC(A$)
```

Type RUN and the number 68 will appear on the screen; this is the ASCII code for the letter D.

ATN(X) — This function is mathematical and we won't use it in this book. This function will give you the arc tangent 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$(68)  
10 PRINT A$
```

Type RUN and this will print the letter D.

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

```
PRINT COS(10)
```

and the Commodore 64 will give you $-.839071529$

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(10) —This will raise 2.718281828 to the tenth power.
```

FRE(X) — This tells you the number of unused (free) bytes available in memory.

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 Commodore will print 99. Try the same program, only let A = INT(-99.887)

LEFT\$(X\$,X) — This function will give you the leftmost X characters in X\$.

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

```
5 A$ = "THE COMMODORE 64 IS A NEAT MACHINE"
20 PRINT LEN(A$)
```

Type RUN and the Commodore 64 will print 34. Note that all the spaces count as a character.

LOG(X) — This function will give you the natural logarithm of a number. To determine the log of a number, simply input:

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

MID\$(S\$,X) — This function will give you a part of a string. You can control the part of the string returned.

```
5 A$ = "MISSISSIPPI"
10 PRINT MID$(A$,8,4)
```

—The 8 tells the Commodore 64 to count 8 places. Then, the 4 tells the computer to print the next 4 places from the beginning point.

Type RUN and you get IPPI.

PEEK(X) — You can PEEK into a memory location in order to learn its contents.

POS(X) — This function will compare two strings and tell you at what point a letter in one string begins occurring in another.

26 THE TOOL KIT SERIES Commodore 64 Edition

RIGHT\$(X\$,@) — This function will give you the rightmost X characters in X\$.

RND(X) — The random function tells the Commodore 64 to generate a random number between 0 and 1.

SGN(X) — This function will return the algebraic sign of the argument. This function will tell you if the number is positive, negative, or zero. This function is useful in programs that require some complicated mathematics.

SIN(X) — SIN represents the trigonometric sine function of X.

SPC(X) — Used only with PRINT, this moves text forward by X spaces of output.

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)
```

STR\$(X) — STR\$ is the opposite of the VAL function. STR\$ changes a specified number (the argument) into a string.

TAB(X) — This function works just like the TAB key on a typewriter. With the PRINT statement, the TAB function tells the computer to begin printing a given number of places from the left side of the page. It looks like this:

```
PRINT TAB (20) "YOUR NAME" — The argument can be any number from 1 to 255.
```

TAN(X) — The TAN function will give you the tangent of the argument.

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

INPUTTING PROCEDURES AND HINTS

If you have read your User's Manual, you already know that entering data

is just like typing except that it is easier to make corrections on the Commodore 64. You should read your User's Manual if you haven't already done so. Here are some simple inputting rules and suggestions.

1. Always number your program lines. We suggest numbering in increments of 10. This gives you space if you need to alter, change, or renumber a program.
2. Get in the habit of using REM or Remark statements to make 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. An example of how to list portions of a program appeared earlier in this chapter under the LIST command description.

VARIABLES

Basically, a variable is something that we assign a value to. We have assigned values to the variables in most of the programs in this book, and you need to understand the concept.

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

```
10 LET X = 1.5
20 LET 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 and 1 Y.

Integer variables are similar to real numeric variables except that they can only be expressed as integers (whole numbers). They are expressed by using the percent (%) sign after the name of the integer variable.

```
X% = 5
AZ% = 77
```

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

28 THE TOOL KIT SERIES Commodore 64 Edition

are designing a *States and Capitals* game and we want a message to indicate an incorrect answer. We can say:

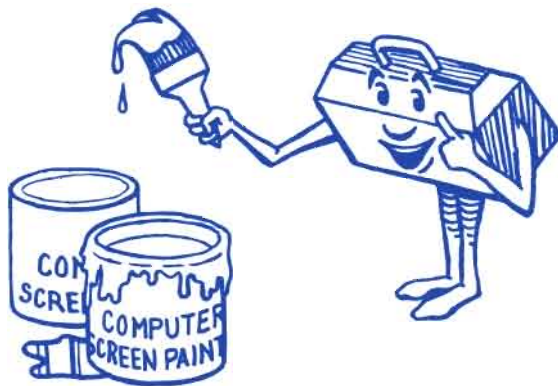
```
10 LET X$ = "WRONG"  
20 PRINT X$
```

We are telling the computer that X\$ means the same thing as the word WRONG. The Commodore 64 really only reads the first two letters of a string variable. The names can be any length but you distinguish one string variable from another with the first two letters of the name.

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. As we identify the variables for you later in the book, remember that there are three types: *Numeric*, *Integer*, and *String*.

Now that we understand how the computer works and we have an introduction to BASIC language commands, statements, and functions, let's begin doing some real programming.





Commodore 64 Color

3

30 THE TOOL KIT SERIES Commodore 64 Edition

How lucky you are to be using the colorful Commodore 64. With color alone, you can dazzle yourself and friends with the “show” programs you will learn in this chapter.

KEYBOARD COLORING

Using color is easy on the Commodore 64. If your tv set or color monitor is adjusted properly, the flashing cursor that appears on the screen is light blue. Anything that you type in will be blue. Try it.

There are sixteen colors that you can choose from the keyboard. Let’s change the cursor color from blue to red. At the same time, press the **CTRL** and **RED** keys. Now the cursor and anything you type is red. Next, change the cursor to green.

At the top of the 64 keyboard are eight color keys, as shown in Fig. 3-1. Try them all out to see how they look.

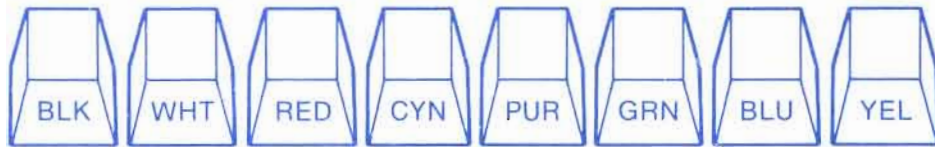


Fig. 3-1. The Commodore 64 color keys.

In order to see the other eight colors that are available on the keyboard, try this (using the Commodore key):

C	1	=	ORANGE
C	2	=	BROWN
C	3	=	LIGHT RED
C	4	=	DARK GRAY
C	5	=	MEDIUM GRAY
C	6	=	LIGHT GREEN
C	7	=	LIGHT BLUE
C	8	=	LIGHT GRAY

NOTE: Blue on a blue background will look invisible, of course. Hiding a character by changing its color to the same color as the background can be useful to us as we develop programs later in the book.

Now, let’s turn the cursor into a paint brush and have some fun. The easiest way to make a brush is by using the Reverse/On key. Press **CTRL** and

32 THE TOOL KIT SERIES Commodore 64 Edition

Now you have a black border and a white screen. 53280 is the location in memory where the border color values are stored; 53281 is for the screen color values. 0 is the color value for black and 1 is the value for white.

Try the following simple program to see the range of combinations that the Commodore 64 can make.

```
10 FORC=0TO15
20 FORD=0TO15
30 POKE53280,C
35 POKE53281,D
40 FORT=1TO100:NEXT
50 NEXTD:NEXTC
```

- Lines 10 and 20 set the C and D variables from values 0 to 15.
- Line 30 POKES the screen border with C as a value.
- Line 35 POKES the screen background with D as a value.
- Line 40 holds that particular screen and border combination for a count of 1 to 500.
- Line 50 completes the FOR . . . NEXT loop by telling the computer to change the values for D and, then, C. The program POKES the first value for the screen border and runs through all 15 screen background colors; then, the border changes to the next value.

For reference, the following list shows you the values you can POKE into locations 53280 and 53281 to obtain the desired screen and border combinations.

COLOR VALUES

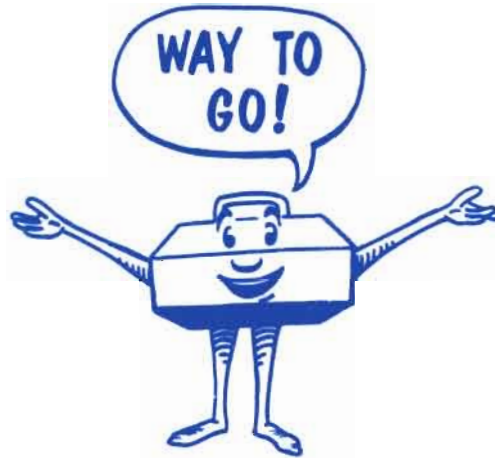
BLACK	— 0	ORANGE	— 8
WHITE	— 1	BROWN	— 9
RED	— 2	LIGHT RED	— 10
CYAN	— 3	DARK GRAY	— 11
PURPLE	— 4	MEDIUM GRAY	— 12
GREEN	— 5	LIGHT GREEN	— 13
BLUE	— 6	LIGHT BLUE	— 14
YELLOW	— 7	LIGHT GRAY	— 15

Let's write a colorful "reward" program. We will use a program like this later in the book to reward you for being successful in a game program.

```
10 PRINT"O"
20 FOR A=0TO15
30 POKE53281,0:POKE53280,A
40 FOR T=1TO100:NEXT T
50 NEXT A
60 FORB=0TO15
70 POKE53281,1:POKE53280,B
80 FOR T=1TO100:NEXT
90 NEXT B
100 GOTO 10
```

- Line 10 clears the screen. Inside the quotes you key in **SHIFT** and **CLR/HOME**.
- Line 20 sets the values of A from 0 to 15.
- Line 30 POKEs the screen background to black and POKEs the border with a value of A.
- Line 40 delays the changes with a timer.
- Line 50 changes to the next A value.
- Lines 60-90 are similar to lines 10-50 except that a white screen background is used.
- Line 100 tells the computer to return to line 10 to begin the program again.

NOTE: In order to stop this program, you push the STOP key since the program has no END or STOP within itself.



COLOR IN PRINT STATEMENTS

When we use the PRINT statement, directions and messages are included in between the quotation marks following PRINT. Remember that when you type:

```
PRINT "GOOD MORNING"
```

the computer responds with

```
GOOD MORNING
```

 (The message inside the quotation marks.)

When we give PRINT statements directions for color or other functions, special symbols appear inside the quotation marks. Try this out; key in:

34 THE TOOL KIT SERIES Commodore 64 Edition

















PRINT "CTRL BLK COMPUTER"

actually shows on the computer as:

PRINT "  COMPUTER"

and prints the word COMPUTER in black letters as soon as the RETURN key is pressed.

Inside PRINT statement quotation marks, the colors will appear as follows:

COLOR	KEY IN	SYMBOL
BLACK	CTRL BLK	
WHITE	CTRL WHT	
RED	CTRL RED	
CYAN	CTRL CYN	
PURPLE	CTRL PUR	
GREEN	CTRL GRN	
BLUE	CTRL BLU	
YELLOW	CTRL YEL	
ORANGE	CTRL 1	
BROWN	CTRL 2	
LIGHT RED	CTRL 3	
DARK GRAY	CTRL 4	
MEDIUM GRAY	CTRL 5	
LIGHT GREEN	CTRL 6	
LIGHT BLUE	CTRL 7	
LIGHT GRAY	CTRL 8	

Note also that inside the PRINT statement, the Reverse On and Reverse Off keys (RVS/ON, RVS/OFF) produce special characters. Let's use our knowledge of color to print a completely colorful statement.

```

10 PRINT"0"
20 PRINT"1"
30 PRINT"2"
40 PRINT"3"
50 PRINT"4"
60 PRINT"5"
70 PRINT"6"
80 PRINT"7"
90 PRINT"8"
100 PRINT"9"
110 PRINT"10"
120 PRINT"11"

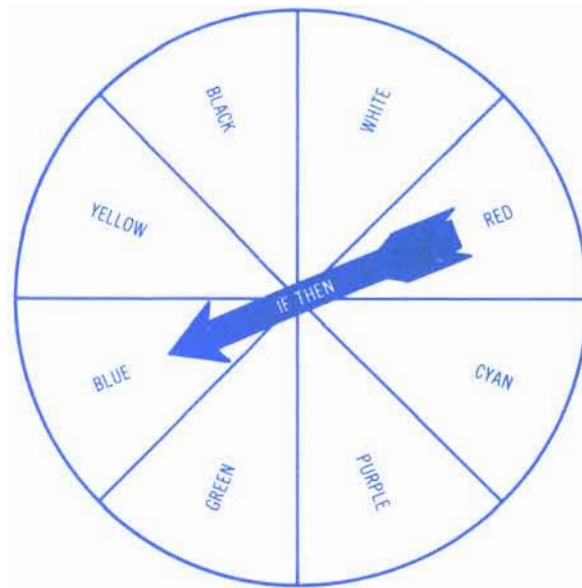
```

COLORFUL 64

Try to figure out what colors we have used. There is a character key in the appendices that may help you. As you use color more in PRINT statements, these symbols will become much more familiar to you. Now make up your own special messages.

COLOR SUBROUTINES

Now that we have covered the basics of coloring the screen and the border as well as using color in programs, let's look at some interesting programs and routines that we can use alone or as parts of other programs.



COLOR CHOICE

Choose your favorite colors and the Commodore 64 will display them for you.

```

10 PRINT"|"
20 PRINT"DO YOU LIKE BLACK,WHITE,RED,CYAN,PURPLE,GREEN,OR YELLOW THE BEST?"
30 A1$="BLACK":A2$="WHITE"
40 A3$="RED"
50 A4$="CYAN"
60 A5$="PURPLE"
70 A6$="GREEN"
80 A7$="BLUE"
90 A8$="YELLOW"
100 PRINT"WHAT IS YOUR FAVORITE COLOR?"
110 GOTO120
116 PRINT"PICK ANOTHER COLOR, PLEASE"
120 INPUT B$
130 PRINT"|"
140 IFB$=A1$ THEN GOSUB 400
150 IFB$=A2$ THEN GOSUB 500
160 IFB$=A3$ THEN GOSUB 600
170 IFB$=A4$ THEN GOSUB 700
180 IFB$=A5$ THEN GOSUB 800
190 IFB$=A6$ THEN GOSUB 900

```

36 THE TOOL KIT SERIES Commodore 64 Edition

```
200 IFB#=#A7# THEN GOSUB 1000
210 IFB#=#A8# THEN GOSUB 1100
250 GOTO 116
400 FOR X=#0T015
410 POKE53281,0:POKE53280,X
420 FOR T=1T0100:NEXT T
430 NEXT X
490 RETURN
500 FOR X=#0T015
510 POKE53281,1:POKE53280,X
520 FOR T=1T0100:NEXT T
530 NEXT X
590 RETURN
600 FOR X=#0T015
610 POKE53281,2:POKE53280,X
620 FOR T=1T0100:NEXT T
630 NEXT X
690 RETURN
700 FORX=#0T015
710 POKE53281,3:POKE53280,X
720 FOR T=1T0100:NEXT T
730 NEXT X
790 RETURN
800 FOR X=#0T015
810 POKE53281,4:POKE53280,X
820 FOR T=1T0100:NEXT T
830 NEXT X
890 RETURN
900 FORX=#0T015
910 POKE53281,5:POKE53280,X
920 FOR T=1T0100:NEXT T
930 NEXT X
990 RETURN
1000 FORX=#0T015
1010 POKE53281,6:POKE53280,X
1020 FOR T=1T0100:NEXT T
1030 NEXT X
1090 RETURN
1100 FOR X=#0T015
1110 POKE53281,7:POKE53280,X
1120 FOR T=1T0100:NEXT T
1130 NEXT X
1190 RETURN
```

- Line 10 clears the screen.
- Lines 30-100 set A1\$ to A*\$ as color string variables.
- Line 110 asks for your favorite color.
- Line 115 tells the program to skip to line 120 for the first color selection.
- Line 116 is given after the first color choice is made.
- Line 120 asks you to “input” your color choice.
- Lines 140-210 determine which color you have chosen and moves the program to the appropriate subroutine.
- Line 250 sends the program to line 116 to ask for another color selection.

The subroutines in lines 400 through 1190 POKE your color selection on the screen background and change the border colors to all 16 possible values (0 to 15). The RETURN statement at the end of each subroutine returns to the program at the next program line after it was called.

COLOR CHOICE is a program that uses simple IF . . . THEN logic as well as the INPUT statement. Now let's look at another colorful program.

```

10 PRINT"SELECT A COLOR SHOW IN ONE OF THE FOLLOWING COLORS:"
20 PRINT
30 PRINT"P.....PURPLE"
40 PRINT"B.....BLACK"
50 PRINT"G.....GREEN"
60 PRINT"Y.....YELLOW"
70 GETC$:IFC$=""THEN?0
80 IFC$="P" THEN GOSUB200
90 IFC$="B" THEN GOSUB300
100 IFC$="G" THEN GOSUB400
110 IFC$="Y" THEN GOSUB 500
120 IF C$<>"P" AND C$<>"B" AND C$<>"G" AND C$<>"Y" THEN GOSUB600
200 FORX=0TO15
210 POKE53281,4:POKE53280,X
220 FOR T=1TO100:NEXT
230 NEXTX
240 GOT010
290 RETURN
300 FORX=0TO15
310 POKE53281,0:POKE53280,X
320 FOR T=1TO100:NEXT
330 NEXT X
340 GOT010
390 RETURN
400 FORX=0TO15
410 POKE53281,5:POKE53280,X
420 FOR T=1TO100:NEXT
430 NEXT X
440 GOT010
490 RETURN
500 FORX=0TO15
510 POKE53281,7:POKE53280,X
520 FOR T=1TO100:NEXT
530 NEXT X
540 GOT010
590 RETURN
600 PRINT"XXXXXXXXYOU DID NOT HIT ONE OF THE COLOR KEYS!!"
610 FOR T=1TO800:NEXT
620 GOT010
690 RETURN

```

- Lines 10-60 give you instructions for SHOW OF COLOR.
- Line 70 tells the program to GET a value for the string variable C\$. If no value is keyed (" "), then the program waits patiently.
- Lines 80-110 send the program to appropriate subroutines if P, B, G, or Y is pressed.
- Line 120 looks to see if a key other than P, B, G, and Y is being pressed and (if that is the case) sends the program to the "wrong key" subroutine.

Subroutines in lines 200 through 590 perform the color shows in the color that you have selected. The subroutine starting at line 600 tells you that a wrong key was hit and sends the program back to its beginning, after a short delay (1 to 800).

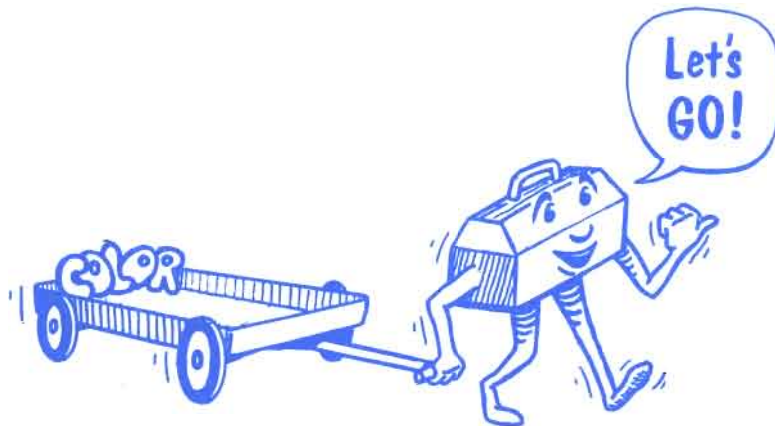
NOTE: The GET statement is similar to the INPUT statement in that both ask the program user to key in data. With GET, there is no question mark (?) asking for input and the **RETURN** key does not have to be pressed as it does with INPUT.

RANDOM COLOR

The last two programs, FAVORITE COLOR and SHOW OF COLOR, offer you a choice of color by controlling the program with IF . . . THEN, INPUT, and GET statements. Let's make a program that produces totally random screen and border combinations.

```
10 PRINT"␣"  
20 C=INT(RND(0)*15)  
30 D=INT(RND(0)*15)  
40 POKE53280,C  
50 POKE53281,D  
60 FOR T=1TO100: NEXT T  
70 GOTO10
```

- Line 10 clears the screen.
- Lines 20 and 30 set the color value of the variables using the INT (integer) and RND (random number generator) functions.
 - A. RND(0) produces random numbers between 0 and 1, such as .012 and .469.
 - B. The *15 multiplies the random number by 15 (the maximum value for color combinations).
 - C. The INT function rounds off the random number times 15 to the nearest integer (or whole number). This gives the computer a value that can be POKEd into the memory locations for screen and border colors.
- Line 40 POKEs the border with random color variable C.
- Line 50 POKEs the screen background with random color variable D.
- Line 60 delays the program for a count of 1 to 100.
- Line 70 sends the program to the start.



We hope you have enjoyed learning to use color on the Commodore 64. The last three programs were designed to be fun as well as to introduce you to some BASIC programming and logic.

In the next few chapters, we will cover sound, graphics, and animation. We will use color subroutines heavily in these chapters and throughout the rest of the book. Experiment and enjoy.



Sound and Music Subroutines

4

42 THE TOOL KIT SERIES Commodore 64 Edition

Your Commodore 64 has a great voice. It has surprising musical capabilities with a range of a full eight octaves through three voices. You can learn to “play” the Commodore 64 with a minimum of practice. In this chapter, we will show you how to make beautiful music with your “64” and how to use sound subroutines to add some pizzazz to your future programs. Let’s get started POKEing around with sound.

POKEing SOUND

You can play the three voices by using POKE statements. You access the sound-related memory locations by POKEing. For the rest of the steps in playing sound, let’s consider only Voice 1 as the example. Later in this chapter, you will learn the POKE settings for Voices 2 and 3.

First, you set the volume for sound. One memory location sets the volume for the voices and noise; its location is 54296. There are 15 volume settings available, 0–15 (0 is off).

The next two settings define how a note will be played. *Attack/Decay* values tell the computer how fast the note should rise (attack) and then fall (decay) from its peak volume level. For Voice 1, the *Attack/Decay* POKE setting is 54277, with values from 0 to 255. *Sustain/Release* settings tell how long to play the note (sustain) and the rate at which it should be slowed (released). For Voice 1, *Sustain/Release* POKE setting is 54278, with values of 0 to 255.

To determine the values to use to POKE for the *Attack/Decay* and *Sustain/Release*, look at Tables 4-1 and 4-2. These tables are adapted from the *Commodore 64 User’s Guide*.

Table 4-1. ATTACK/DECAY Values

Voice 1 Setting	High Attack	Medium Attack	Low Attack	Lowest Attack	High Decay	Medium Decay	Low Decay	Lowest Decay
54277	128	64	32	16	8	4	2	1

To get an *Attack/Decay* setting, simply add an attack rate to a decay rate. For high attack and medium decay: $128 + 4 = 132$; therefore, the entire setting would be POKE 54277,132. To get the highest attack rate possible, add all the attack values together ($128 + 64 + 32 + 16 = 240$). For the highest attack rate and medium decay, the setting should be POKE 54277,244.

Sustain/Release settings are determined just like the *Attack/Decay* values. For a high sustain level and a high release level, POKE 54278,136.

The fourth setting determines the note you want to play. For each note you play, you must POKE high and low frequencies for the note. If you want to POKE an “F” note for Voice 1, you might choose the following high and low

Table 4-2. SUSTAIN/RELEASE Values

Voice 1 Setting	High Sustain	Medium Sustain	Low Sustain	Lowest Sustain	High Release	Medium Release	Low Release	Lowest Release
54278	128	64	32	16	8	4	2	1

frequency settings. (Note: Frequency settings for various musical notes are listed in an appendix of your *Commodore 64 User's Guide*.)

High frequency POKE 54273,45
 Low frequency POKE 54272,198

The high and low frequencies may range from 0 to 255 depending upon the note or effect you want.

The last sound setting you must select for Voice 1 is the waveform setting. There are four to choose from and each has its own starting and stop values. This is shown in Table 4-3.

Table 4-3. Waveform Settings for Voice 1

Waveform	Start	Stop
Triangle	POKE 54276,17	POKE 54276,16
Sawtooth	POKE 54276,33	POKE 54276,32
Pulse	POKE 54276,65	POKE 54276,64
Noise	POKE 54276,129	POKE 54276,128

The different waveforms produce very different results.

Triangle This waveform is relatively smooth and produces sounds like reed instruments.

Sawtooth This waveform is more rough and produces brassy sounds like trumpets and other brass instruments.

Pulse This waveform "pulses" on and off at a variable rate that you control. For Voice 1:

High Pulse — POKE 54275,0-15
 Low Pulse — POKE 54274,0-255

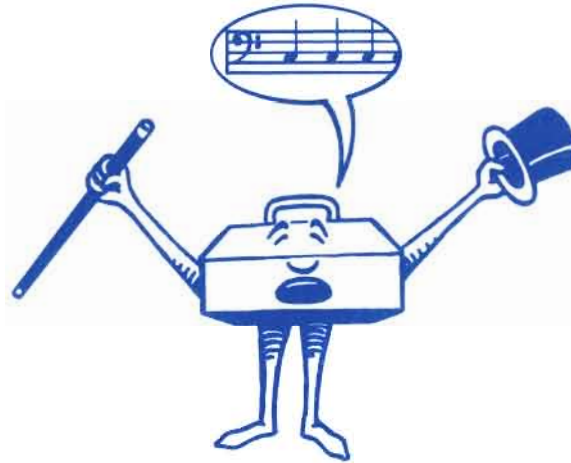
This waveform produces sounds most like a piano, an instrument with abrupt changes in notes.

Noise This waveform is random and produces noises and sound.

Input these lines:

44 THE TOOL KIT SERIES Commodore 64 Edition

```
10 POKE54296,15
20 POKE54277,17
25 POKE54278,65
30 POKE54273,34
40 POKE54272,75
50 POKE54276,17
```



You have just played Middle C of the fifth octave. If you want the tone to stop, you can POKE directly into the computer; use POKE 54296,0 (turns the volume off) or push **STOP** and then **RESTORE**. Let's look at each program line:

- Line 10 POKES the volume at 15, the highest setting.
- Line 20 POKES a low attack value (16) and low decay setting (1), for $16 + 1 = 17$.
- Line 25 POKES a medium high sustain setting (64) and a low release value(1), for $64 + 1 = 65$.
- Line 30 POKES the high frequency for Middle C.
- Line 40 POKES the low frequency for Middle C.
- Line 50 POKES the triangle waveform.

If you want Middle C to play for a specific period of time or duration, then let's lengthen the program. This is shown in the following program.

```
10 POKE54296,15
20 POKE54277,17
25 POKE54278,65
30 POKE54273,34
40 POKE54272,75
50 POKE54276,17
55 FOR T=1 TO 200: NEXT
60 POKE54276,128
```

- Line 55 holds the note for a count of 1 to 200.
- Line 60 stops the waveform by POKeing in its stop value, 128.

SOUND VARIABLES

Let's play with the preceding program to learn more about sound.

Volume	Change the value in line 10 to anything from 1 to 15.
Note	Change the values in lines 30 and 40 to play different notes. The values of high and low frequencies for all notes are in an appendix of your <i>User's Guide</i> .
Tone	Change the Attack/Decay and Sustain/Release settings for different results in lines 20 and 25.
Waveform	Try all four waveforms in line 50. Remember to POKE high and low pulse settings when using the pulse waveform.
Duration	Change the range of "T" in the FOR . . . NEXT statement in line 55 to a larger number for a longer duration, or smaller for a shorter duration.

THE NOISE MAKER

Waveform value 129 gives us some fun noises which we can use to enhance our programs and games. These sound effects are fun but you should create some of your own. Input and run this program:

```

10 POKE54296,15
20 POKE54277,9
30 POKE54278,129
40 POKE54273,2
50 POKE54272,30
60 GOSUB200
70 POKE54273,2
80 POKE54272,56
90 GOSUB200
100 POKE54273,2
110 POKE54272,24
120 GOSUB200
150 END
200 POKE54276,129
210 FOR T=1 TO 1000:NEXT
220 POKE54276,128
290 RETURN

```

The program sounds like something is about to crash, but it doesn't. This program has a lot of lines for a simple sound effect even though we use a GOSUB to make it more efficient. Later in this chapter, we will show how other BASIC statements can make playing sounds more efficient.

SOUND ENCYCLOPEDIA

Let's look at the following short program. It lets you select the note, duration, volume, and other variables you want to hear.

46 THE TOOL KIT SERIES Commodore 64 Edition

```
10 PRINT"☐"  
20 INPUT"VOLUME: 1-15";V  
30 INPUT"ATTACK/DECAY: 0-255";AD  
40 INPUT"SUSTAIN/RELEASE: 0-255";SR  
50 INPUT"DURATION: 1 TO ?";D  
60 INPUT"NOTE, HI FREQUENCY";HF  
70 INPUT"NOTE, LO FREQUENCY";LF  
80 INPUT"WAVEFORM: 17,33,65,129?";W  
90 IFW=65 THEN GOSUB200  
100 POKE54296,V  
110 POKE54277,AD:POKE54278,SR  
120 POKE54273,HF:POKE54272,LF  
130 POKE54276,W  
140 FOR T=1TOD:NEXT  
150 POKE54276,W-1:POKE54296,0  
160 GOTO10  
200 INPUT"HI PULSE RATE: 0-15";HP  
210 INPUT"LO PULSE RATE: 0-255";LP  
220 POKE54275,HP:POKE54274,LP  
290 RETURN
```

Using the INPUT statements in lines 20 through 80 lets you select your own variables. Note that we put a subroutine in the program which allows you to INPUT and POKE high and low pulse values if you choose the pulse waveform in line 80.

Now you can control sound more completely! Defining each value as a variable:

V = Volume
AD = Attack/Decay
SR = Sustain/Release
D = Duration
HF = High Frequency
LF = Low Frequency
W = Waveform
HP = High Pulse
LP = Low Pulse

makes programming and playing with sound much easier.

POKEing MORE THAN ONE VOICE

Earlier in this chapter, we said that there are three voices that you can play on the Commodore 64. Up until now, we have used only Voice 1 in all examples. Let's look at the settings (Table 4-4) for all three voices.

Table 4-4. Sound Control Settings

VOICE	ATTACK DECAY	SUSTAIN RELEASE	HIGH FREQ	LOW FREQ	WAVE- FORM	HIGH PULSE	LOW PULSE
1	54277	54278	54273	54272	54276	54275	54274
2	54284	54285	54280	54279	54283	54282	54281
3	54291	54292	54287	54286	54290	54289	54288

We can play all three voices at the same time with POKE statements. This allows us the option to make musical chords rather than just single notes. Try the following program:

```

10 POKE54296,15
20 POKE54277,0:POKE54278,240
30 POKE54273,34:POKE54272,75
50 POKE54284,0:POKE54285,240
60 POKE54280,43:POKE54279,52
70 POKE54291,0:POKE54292,240
80 POKE54287,45:POKE54286,198
100 POKE54276,17:POKE54283,17:POKE54290,17
110 PORT=1T0500:NEXT
120 POKE54276,16:POKE54283,16:POKE54290,16
    
```

Three notes (C, E, and F) are played in the fifth octave at the same time. All these have the same Attack/Decay, Sustain/Release, and Waveform variables. Think of the sound variation you can produce with changes in each voice.

- Lines 20 through 80 set the Attack/Decay, Sustain/Release, and Note frequencies for Voices 1, 2, and 3.
- Line 100 turns the triangle waveform on for all three voices.
- Line 120 stops the triangle waveform for each of the voices.

MUSIC LESSON SUBROUTINES



C SCALE

Let's begin our lesson with a scale. This FUN SCALE is not exactly musically correct, but it is an easy way of approximating a scale for a game or quiz.

```

10 REM**FUN SCALE
20 POKE54296,15
30 POKE54277,9:POKE54278,65
40 FOR Y=25T0250 STEP 25
45 X=Y
50 POKE54273,Y:POKE54272,X
60 POKE54276,17
70 FOR T=1T0100:NEXT
80 POKE54276,16
90 NEXT Y
    
```

In this program, the Commodore plays notes with high and low frequencies from 25 to 250—in steps of 25. Each note is played for a count of 100. Let's learn how to play a real scale now. In order to do so, Table 4-5 gives a chart of musical notes—making the notes and corresponding numbers for two octaves easy to identify.

48 THE TOOL KIT SERIES Commodore 64 Edition

Table 4-5. Frequency Values of Musical Notes

NOTE	FIFTH OCTAVE		FOURTH OCTAVE	
	HIGH FREQ	LOW FREQ	HIGH FREQ	LOW FREQ
C	34	75	16	195
C#	36	85	17	195
D	38	126	18	209
D#	40	200	19	239
E	43	52	21	31
F	45	198	22	96
F#	48	127	23	181
G	51	97	25	30
G#	54	111	26	156
A	57	172	28	49
A#	61	126	29	223
B	64	188	31	165

Please remember that the values in Table 4-5 are approximate and if you think they should be a little higher or lower, that is all right. There are no right answers.

```

5 REM**C SCALE
10 POKE54296,15:POKE54277,9:POKE54278,129
20 H=54273:L=54272
30 POKEH,34:POKEL,75
40 GOSUB 200
50 POKEH,38:POKEL,126
60 GOSUB 200
70 POKEH,43:POKEL,52
80 GOSUB 200
90 POKEH,45:POKEL,198
100 GOSUB 200
110 POKEH,51:POKEL,97
120 GOSUB 200
130 POKEH,57:POKEL,172
140 GOSUB 200
150 POKEH,64:POKEL,188
160 GOSUB 200
170 POKEH,68:POKEL,149
180 GOSUB 200
190 END
200 POKE54276,17
210 FOR T=1TO300:NEXT
220 POKE54276,16
290 RETURN

```

The preceding is a little program that will play the C Scale one note at a time. After all eight notes have been played, the program ends. How could you get the scale to play again and again? Change line 190 to:

```
190 GOTO 10
```

and the scale will play continuously.

You can instruct the computer to play any musical scale using Program 4-2 as an example. All that is needed is to change the notes in lines 40 to 170.

The values for G Major Scale are:

	High	Low
G	50	60
A	56	99
B	63	75
C	67	15
D	75	69
E	84	125
F#	94	214
G	100	121

Scales can be combined if you like. Simply POKE the scales on the same line and use a colon between the statements. For combined C and G scales on Voices 1 and 2, the program would be as shown in the following listing.

```

5 REM**COMBINED SCALES
10 POKE54296,15:POKE54277,9:POKE54278,129
15 POKE54284,9:POKE54285,129
20 H=54273:L=54272
25 H=54280:B=54279
30 POKEH,34:POKEL,75:POKER,50:POKEB,60
40 GOSUB 200
50 POKEH,38:POKEL,126:POKER,56:POKEB,99
60 GOSUB 200
70 POKEH,43:POKEL,52:POKER,63:POKEB,75
80 GOSUB 200
90 POKEH,45:POKEL,198:POKER,67:POKEB,15
100 GOSUB 200
110 POKEH,51:POKEL,97:POKER,75:POKEB,69
120 GOSUB 200
130 POKEH,57:POKEL,172:POKER,84:POKEB,125
140 GOSUB 200
150 POKEH,64:POKEL,188:POKER,94:POKEB,214
160 GOSUB 200
170 POKEH,68:POKEL,149:POKER,100:POKEB,121
180 GOSUB 200
190 END
200 POKE54276,17:POKE54283,17
210 FOR T=1TO300:NEXT
220 POKE54276,16:POKE54283,16
290 RETURN
    
```

Notice that this is the same program as for the C Scale program shown earlier with Voice 2 POKEs added.

PLAYING CHORDS ON THE COMMODORE 64

As we mentioned earlier in this chapter, you can activate more than one voice at a time. The following program illustrates how the Commodore can be made to sound like an organ.

50 THE TOOL KIT SERIES Commodore 64 Edition

```
5 REM**CHORDS!**
10 POKE54296,15:POKE54277,0:POKE54278,240
15 POKE54284,0:POKE54285,240:POKE54291,0:POKE54292,240
20 H=54273:L=54272
25 A=54280:B=54279:X=54287:Y=54286
30 POKEH,34:POKEL,75:POKEA,43:POKEB,52:POKEX,51:POKEY,97
40 GOSUB 200
50 POKEH,38:POKEL,126:POKEA,48:POKEB,127:POKEX,57:POKEY,172
60 GOSUB 200
70 POKEH,43:POKEL,52:POKEA,54:POKEB,111:POKEX,64:POKEY,188
80 GOSUB 200
90 POKEH,47:POKEL,107:POKEA,59:POKEB,190:POKEX,75:POKEY,69
100 GOSUB 200
110 POKEH,51:POKEL,97:POKEA,63:POKEB,75:POKEX,75:POKEY,69
120 GOSUB 200
130 POKEH,57:POKEL,172:POKEA,71:POKEB,12:POKEX,84:POKEY,125
140 GOSUB 200
150 POKEH,64:POKEL,188:POKEA,79:POKEB,191:POKEX,94:POKEY,214
160 GOSUB 200
170 POKEH,68:POKEL,149:POKEA,84:POKEB,125:POKEX,100:POKEY,121
180 GOSUB 200
190 POKE54296,0:END
200 POKE54276,17:POKE54283,17:POKE54290,17
210 FOR T=1TO1000:NEXT
220 POKE54276,16:POKE54283,16:POKE54290,16
290 RETURN
```

- Lines 30 through 170 represent the natural chords for C, D, E, F, G, A, B, and C.
- Subroutine at line 200 starts and stops the waveform for each chord according to the timer in line 210.

How can we add to this program to make it even more interesting? Instead of the computer having the fun of playing the chords, let's play the organ ourselves.

THE COMMODORE 64 ORGAN

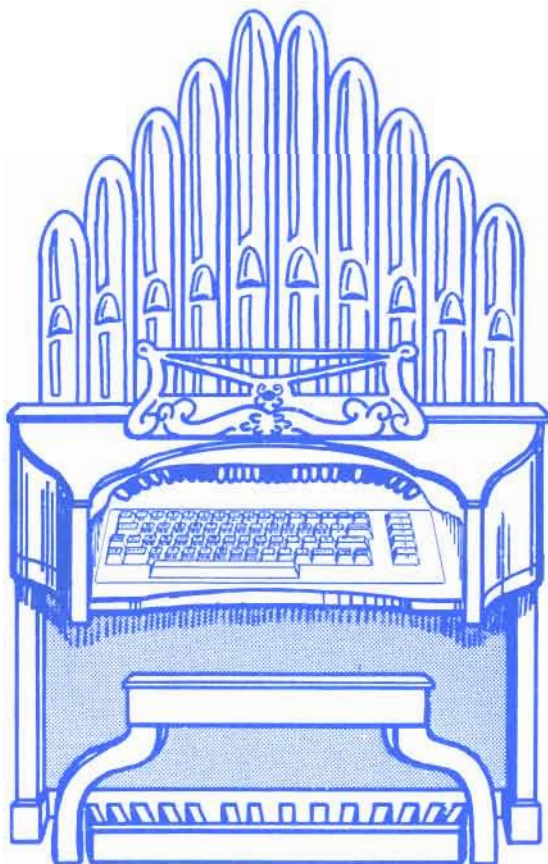
Using our chord organ program, let's make some changes so we can input the chord we want to play. These are shown in the following program.

```
1 REM**COMMODORE 64 ORGAN**
2 POKE54296,15:POKE54277,0:POKE54278,240
3 POKE54284,0:POKE54285,240:POKE54291,0:POKE54292,240
4 H=54273:L=54272
5 A=54280:B=54279:X=54287:Y=54286
10 PRINT"PRESS C,D,E,F,G,A,B, OR C2 FOR THE CHORD YOU WANT"
12 INPUT A$
14 IFA$="C" THEN GOTO 30
16 IFA$="D" THEN GOTO 50
17 IFA$="E" THEN GOTO 70
18 IFA$="E" THEN GOTO 70
20 IFA$="F" THEN GOTO 90
22 IFA$="G" THEN GOTO 110
24 IFA$="A" THEN GOTO 130
26 IFA$="B" THEN GOTO 150
28 IFA$="C2" THEN GOTO 170
30 POKEH,34:POKEL,75:POKEA,43:POKEB,52:POKEX,51:POKEY,97
40 GOSUB 200
50 POKEH,38:POKEL,126:POKEA,48:POKEB,127:POKEX,57:POKEY,172
60 GOSUB 200
```

```

70 POKEH,43:POKEL,52:POKER,54:POKEB,111:POKEX,64:POKEY,188
80 GOSUB 200
90 POKEH,47:POKEL,107:POKER,59:POKEB,190:POKEX,75:POKEY,69
100 GOSUB 200
110 POKEH,51:POKEL,97:POKER,63:POKEB,75:POKEX,75:POKEY,69
120 GOSUB 200
130 POKEH,57:POKEL,172:POKER,71:POKEB,12:POKEX,84:POKEY,125
140 GOSUB 200
150 POKEH,64:POKEL,188:POKER,79:POKEB,191:POKEX,94:POKEY,214
160 GOSUB 200
170 POKEH,68:POKEL,149:POKER,84:POKEB,125:POKEX,100:POKEY,121
200 POKE54276,17:POKE54283,17:POKE54290,17
210 FOR T=1T01000:NEXT
220 POKE54276,16:POKE54283,16:POKE54290,16
230 GOTO 10
290 RETURN

```



It is simple to make the Commodore 64 “interactive.” Lines 14 through 28 ask if a key is entered in response to the INPUT and then make the computer go to the appropriate chord routine.

52 THE TOOL KIT SERIES Commodore 64 Edition

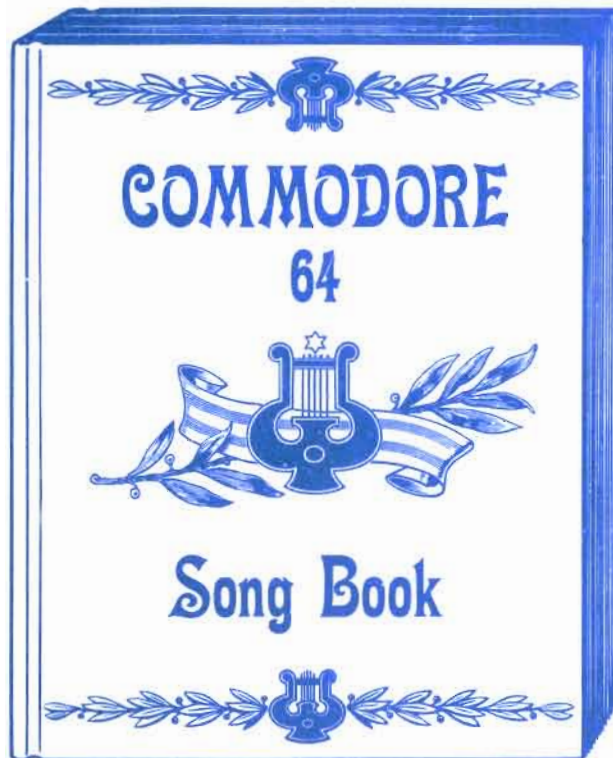
Here are some hints for playing and adapting the Commodore 64 Organ:

1. You can make the organ play without interruption by inputting your next chord choice as soon as you hear the first. The Commodore 64 thinks pretty fast and it will accept the input as soon as the preceding chord starts to play. Of course, you must press **RETURN** after each selection.
2. The organ can become a piano just by deleting Voices 2 and 3 and the values you don't need. Only one note will be played at a time. The simplicity and clarity of piano music is nice. Have some fun with this adaptation.
3. Would you like to end the program with a key input? Easy. Simply add these lines to the program:

```
29 IF A$ = "Z" THEN GOTO 300  
300 END
```

Now pressing "Z" stops the program cold for you, saving you from using **RUN/STOP** and **RESTORE**.

COMMODORE SONG BOOK



You can play any song you want by POKEing the sounds in in the proper order. We can see how even short songs would become long and dull to enter. BASIC provides an easier way to do it.

SOUND SUBROUTINES USING DATA FILES

The DATA and READ statements can save a lot of inputting time. Values can be assigned to variables by telling the computer to READ from a DATA list. The DATA statement does not tell the computer to do anything, it simply provides values in the list.

When we use the READ command and a DATA file, we have to keep some things in mind. The computer begins with the first value in the DATA list and continues through until there is no more data or until the computer finds a value that tells it to stop. Be sure you input the correct data and be sure that the notes are in the right order, because if you don't, the "64" will play some sour notes.

Let's use the READ and DATA statements to make some beautiful music with the Commodore 64. Input the following program:

```

300 REM***MUSIC WITH DATA STATEMENTS**
310 POKE54296,15:POKE54277,9:POKE54275,0:POKE54274,255:POKE54278,0
330 READH
340 IFH=-1 THEN END
350 READL
370 POKE54273,H:POKE54272,L:POKE54276,65
375 FOR T=1TO300:NEXT
380 POKE54276,64
410 GOTO 330
500 DATA34,75,34,75,51,97,51,97,57,172,57,172,51,97,45,198,45,198
510 DATA43,52,43,52,38,126,38,126,34,75,51,97,51,97,45,198,45,198
520 DATA43,52,43,52,38,126,51,97,51,97,45,198,45,198,43,52,43,52
530 DATA38,126,34,75,34,75,51,97,51,97,57,172,57,172,51,97,45,198
540 DATA45,198,43,52,43,52,38,126,38,126,34,75
590 DATA-1,-1
    
```

When you run this little beauty, you will hear an electronic rendition of TWINKLE, TWINKLE, LITTLE STAR. Save this program because we will show you some graphic subroutines to use with this song.

Here are the note sequences of some old favorites that you can try out.

Mary Had a Little Lamb

EDCDEEE DDD EGG EDCEEEDD EDC

Are You Sleeping

GGABGGABG BCD BCD DEDCB GDEDCBGGDG GDG

54 THE TOOL KIT SERIES Commodore 64 Edition

You can add many others to your own song library, from nursery songs to Carole King. We will use the DATA and READ statements more in later chapters. For now, just remember that these are great time savers.

ARCADE-TYPE SOUNDS

Earlier in this chapter, we mentioned the noise maker for the Commodore 64. All the programs in this section will be particularly useful for game programs, arcade games, and educational programs. You can use all three voices of the Commodore 64 for these sound effects.

Type the following program in:

```
10 REM**POLICE SIREN**
20 POKE54296,15:POKE54277,136:POKE54278,129
30 POKE54273,36:POKE54272,85
40 GOSUB 100
50 POKE54273,64:POKE54272,188
60 GOSUB 100
70 GOTO 20
100 POKE54276,33
110 FOR T=1TO350:NEXT
120 POKE54276,32
150 RETURN
```



This sounds like a European police siren, doesn't it? You can use this in a chase or time-sensitive game. However, one cannot listen to this siren for long without panicking.

Let's use the next routine to make the sound of a crash.

```
10 REM**CRASH**
20 POKE54277,129:POKE54278,65:POKE54273,5:POKE54272,251
30 FOR V=15 TO 0 STEP-1
40 POKE54276,129:POKE54296,V
50 FOR T=1TO50:NEXT
60 NEXT V
```

To experiment, you should change the duration time in line 40. This routine uses the volume as the changing variable while the noise actually stays constant.

If you develop any type of space game, a rocket blast-off noise can be essential. Try this.


```
10 REM**BLAST-OFF**
20 POKE54277,9:POKE54278,129:POKE54273,25:POKE54272,100
30 FOR V=15T00STEP-.25
40 POKE54296,V:POKE54276,129
50 FORT=1T050:NEXT
60 NEXT V
```



Now try this program.



```
10 REM***MACHINE GUN***
20 POKE54296,15:POKE54277,8:POKE54278,1
30 POKE54273,34:POKE54272,75
40 POKE54276,129
50 FOR T=1T050:NEXT
60 POKE54276,128
70 GOTO 20
```

Now, how about some musical reinforcement for a correct answer to a quiz.

56 THE TOOL KIT SERIES Commodore 64 Edition

```
10 POKE54296,15
20 POKE54277,9
30 FOR T=1TO50:NEXT
50 READ A
70 READ B
90 IF B=-1 THEN STOP
90 POKE54273,A:POKE54272,B
100 POKE54276,17
110 FOR T=1TO50:NEXT
120 POKE54276,16
140 GOTO20
150 DATA34,75,38,126,43,52,45,198
160 DATA51,97,57,172,64,188,68,149
170 DATA -1,-1
```

However, if you enter an answer incorrectly, your Commodore 64 might give you this message:

```
10 REM***WRONG ANSWER***
20 POKE54296,15
30 POKE54277,0:POKE54278,240
40 READA:READB:READD
90 IF B=-1 THEN STOP
100 POKE54273,A:POKE54272,B
110 POKE54276,17
120 FOR T=1TO50:NEXT
130 POKE54276,16
140 GOTO 30
150 DATA 12,143,100,8,97,500
160 DATA-1,-1,-1
```

The more you work with sound and music, the more interesting touches you can add to your programs. Keep experimenting.

PUTTING SOUND AND COLOR TOGETHER

We discussed color in Chapter 3 and now we have introduced you to some sound subroutines for the Commodore 64. Let's combine what we have learned into one program. Let's change the Commodore 64 Organ into the *Commodore 64 Color Organ*.

```
1 REM***COMMODORE 64 COLOR ORGAN**
2 POKE54296,15:POKE54277,0:POKE54278,240
3 POKE54284,0:POKE54285,240:POKE54291,0:POKE54292,240
4 H=54273:L=54272:K=53281
5 A=54280:B=54279:X=54287:Y=54286
10 PRINT"PRESS C,D,E,F,G,A,B, OR C2 FOR THE CHORD YOU WANT"
12 INPUT A$
14 IF A$="C" THEN GOTO 30
16 IF A$="D" THEN GOTO 50
17 IF A$="E" THEN GOTO 70
18 IF A$="E" THEN GOTO 70
20 IF A$="F" THEN GOTO 90
22 IF A$="G" THEN GOTO 110
24 IF A$="A" THEN GOTO 130
26 IF A$="B" THEN GOTO 150
28 IF A$="C2" THEN GOTO 170
30 POKEH,34:POKEL,75:POKEA,43:POKEB,52:POKEX,51:POKEY,97:POKEK,0
40 GOSUB 200
50 POKEH,38:POKEL,126:POKEA,48:POKEB,127:POKEX,57:POKEY,172:POKEK,1
60 GOSUB 200
70 POKEH,43:POKEL,52:POKEA,54:POKEB,111:POKEX,64:POKEY,188:POKEK,2
```

```

80 GOSUB 200
90 POKEH,47:POKEL,107:POKEA,59:POKEB,190:POKEX,75:POKEY,69:POKEK,3
100 GOSUB 200
110 POKEH,51:POKEL,97:POKEA,63:POKEB,75:POKEX,75:POKEY,69:POKEK,4
120 GOSUB 200
130 POKEH,57:POKEL,172:POKEA,71:POKEB,12:POKEX,84:POKEY,125:POKEK,5
140 GOSUB 200
150 POKEH,64:POKEL,188:POKEA,79:POKEB,191:POKEX,94:POKEY,214:POKEK,6
160 GOSUB 200
170 POKEH,68:POKEL,149:POKEA,84:POKEB,125:POKEX,100:POKEY,121:POKEK,7
200 POKE54276,17:POKE54283,17:POKE54290,17
210 FOR T=1TO1000:NEXT
220 POKE54276,16:POKE54283,16:POKE54290,16
230 GOTO 10
290 RETURN

```

With the additional POKE statements in lines 30 through 170, we have added a splash of color to our Organ. Each time a different chord is played, the screen flashes with different colors.

In this chapter, we have covered many ways to use sound and music on the Commodore 64. We encourage you to experiment and try inventing your own subroutines to use.





Graphics Subroutines

5

60 THE TOOL KIT SERIES Commodore 64 Edition

One of the main reasons you probably bought your Commodore 64 was to create computer graphics. Right? Well, in this chapter, we will show you how to create mazes, borders, graphs, playing boards for games, and a variety of fun-filled and useful characters.

CREATING GRAPHICS WITH THE PRINT COMMAND

Graphics can be created using the PRINT command. Simply put the graphics characters inside quotation marks following PRINT. Try this program:

```
10 PRINT"  _  "  
20 PRINT"|  | "  
30 PRINT"|  ●  | "  
40 PRINT"|  ^  | "  
50 PRINT" \  / "  
60 PRINT" /  \  
"
```

The program will display a friendly face when you type RUN and press **RETURN**. The completed face looks like this:



Anyone you know? For a face you will never forget, add this line:

```
70 GOTO 10
```

and the face is printed over and over.

You can place the face anywhere on the screen using the PRINT command. Try this:

```
5 PRINT"  _  "  
  |  | "  
  |  ●  | "  
  |  ^  | "  
  \  / "  
  /  \  
  "
```

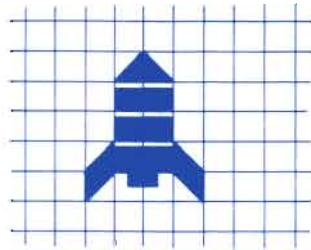
This clears the screen and starts printing 4 lines down on the screen. The **Q** is the unshifted cursor key **↑** inside quotation marks. To move the face more toward the screen's center, use the TAB function. Add TAB(10) to each of the lines (lines 10 through 60). Line 10 would look like this:

```
10 PRINT TAB(10)"——"
```

Now let's use PRINT to make our own space shuttle.

```
10 PRINT "XXXXXXXXXX"
20 PRINT "  /  \"
30 PRINT " /  \"
40 PRINT "/  \"
50 PRINT "/  \"
60 PRINT "/  \"
70 PRINT "/  \"
```

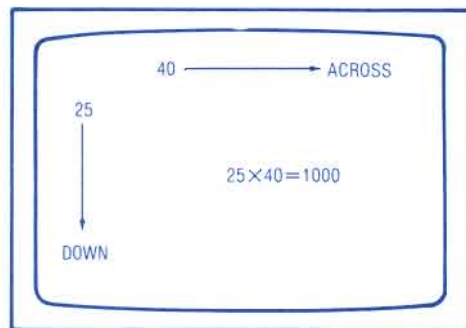
Improve on the space shuttle if you want. The graphics on the keyboard offer a lot of flexibility, especially when you learn to use the reverse of the graphics. As your graphic design becomes more complex, you may find that graph paper is helpful. Using the PRINT command, try to make this new spacecraft.



It is not easy. When you try to add color to the graphics, you will realize how tough graphics like this one can become inside the PRINT quotation marks.

CREATING GRAPHICS USING POKE


We POKEd sound and color, and we can POKE graphics as well. Just like in sound, music, and color; we POKE values into various screen memory locations to create graphics.



62 THE TOOL KIT SERIES Commodore 64 Edition

Screen Maps

Your Commodore 64 has 1000 screen memory locations (40 across times 25 down). Count the rows and columns for yourself. Each of these locations has an address. Whenever you want to POKE a character into a location, you POKE the character into its "memory address." For characters, the addresses range from 1024 to 2023 (from top left-hand corner to lower right-hand corner).

Each keyboard character has a POKE value. See Appendix B for a listing of the screen codes for each character. The POKE value for a solid ball is 81. If you want to POKE the solid ball  into the upper left-hand corner ("home" position), type in:

```
POKE 1024, 81
```

No line numbers are needed at this point.

There are 1000 memory locations for color also. The locations start at 55296 and end at 56295. You must POKE in the color with the character for it to appear. The upper left-hand corner of the color code memory map is position 38400. The colors available are coded as:

BLACK	0	PURPLE	4	ORANGE	8	MEDIUM GRAY	12
WHITE	1	GREEN	5	BROWN	9	LIGHT GREEN	13
RED	2	BLUE	6	LIGHT RED	10	LIGHT BLUE	14
CYAN	3	YELLOW	7	DARK GRAY	11	LIGHT GRAY	15

Now try this:

```
POKE 1024, 81  
POKE 55296, 5
```

A solid green ball will appear in the upper left-hand position.

The maps in Figs. 5-1 and 5-2 will show you all the locations by row and column for character and color memory. These maps are identical to those shown in the User's Guide that came with your Commodore 64.

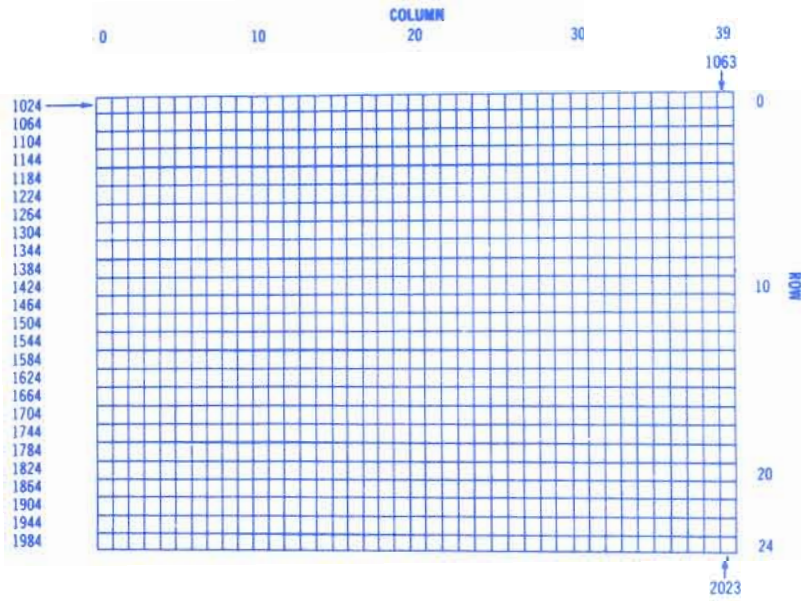


Fig. 5-1. Screen character memory map.

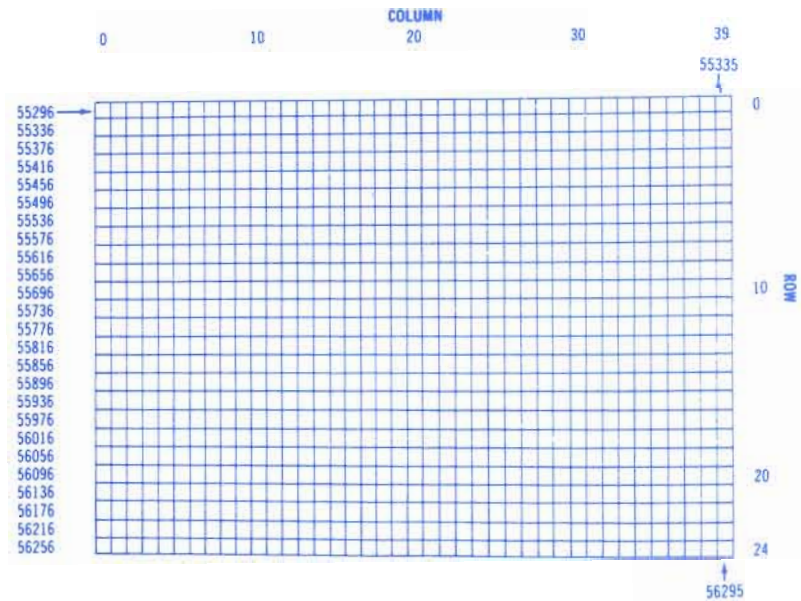


Fig. 5-2. Color code memory map.

64 THE TOOL KIT SERIES Commodore 64 Edition

Screen Border

Looking at the two memory maps, let's devise a program for creating an additional border around the screen. (We changed screens and borders using color in Chapter 3.) We want a red checkerboard border around our screen. How do we do this?

Step 1. In order to get the top border filled, we have to POKE from 1024 to 1063. The BASIC language offers us a convenient shortcut using FOR...NEXT loops. 127 is the character code for ■.

```
10 FORZ=1024T01063:POKEZ,127:POKEZ+54272,2:NEXT
```

54272 is the difference between 55296 and 1024.

Step 2. Now let's fill the border on the left side.

```
20 FORZ=1024T01984 STEP40:POKEZ,127:POKEZ+54272,2:NEXT
```

We use increments of 40 to tell the computer where to POKE the border.

Step 3. Next, let's fill the border on the right side.

```
30 FORZ=1063T02023 STEP40:POKEZ,127:POKEZ+54272,2:NEXT
```

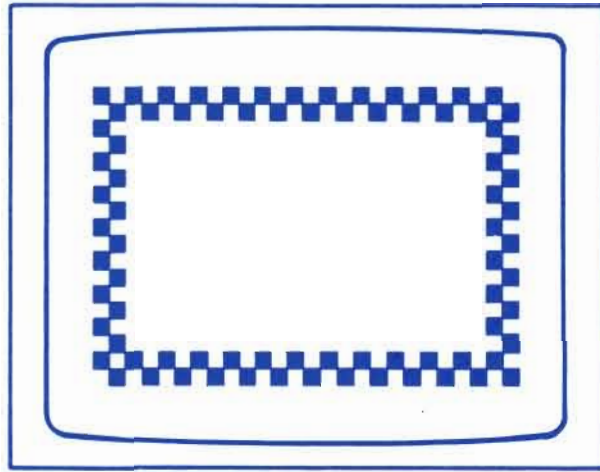
Step 4. Now, the bottom border.

```
40 FORZ=1984T02023:POKEZ,127:POKEZ+54272,2:NEXT
```

We could make our program a little shorter by changing lines 20 and 30 to become:

```
20 FORZ=1024T01984 STEP40:POKEZ,127:POKEZ+54272,2  
30 POKEZ+39,127:POKEZ+54311,2:NEXT
```

See how helpful arithmetic can be?



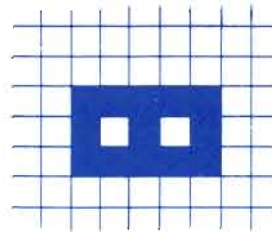
Now, change the border color or the character color. How about a solid black border?

```
POKE Z, 160 and POKE Z + 54272, 0.
```

Score Box

Using what we learned about the border, let's make a simple "score box" that you can use for games or enhancement purposes.

```
10 FORZ=1044 TO 1048:POKEZ,160:POKEZ+54272,5:NEXT
20 FORZ=1044 TO 1124STEP40:POKEZ,160:POKEZ+54272,5
30 POKEZ+2,160:POKEZ+54274,5:POKEZ+4,160:POKEZ+54276,5:NEXT
40 FORZ=1124 TO 1128:POKEZ,160:POKEZ+54272,5:NEXT
```



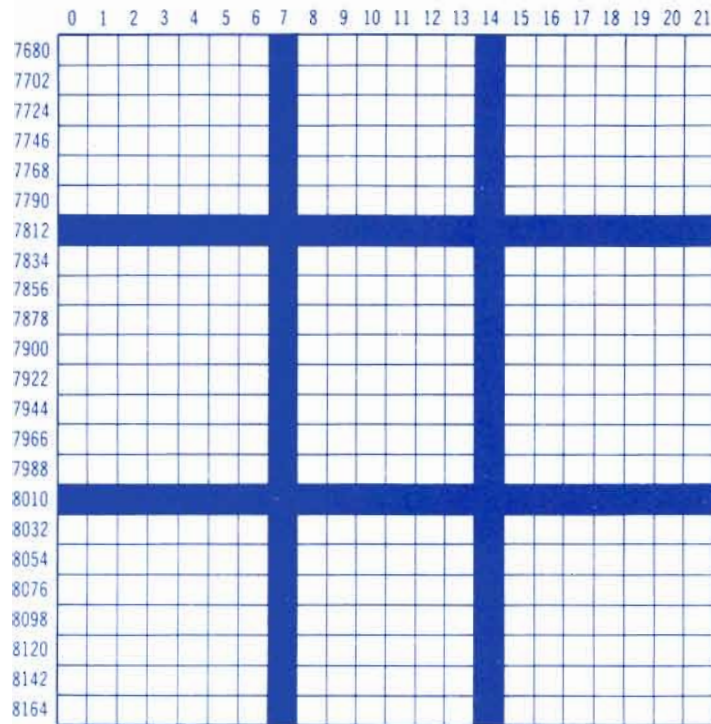
Lines 20 and 30 contain the directions for drawing the three vertical lines; Z , $Z + 2$, and $Z + 4$. Move the score box to another location on the screen. Use other graphics characters to make a better score box.

66 THE TOOL KIT SERIES Commodore 64 Edition

Tic-Tac-Toe Board

Now, let's make a Tic-Tac-Toe board. Give this a try.

```
100 PRINT"☐"  
110 FORG=1264 TO 1284:POKEG,160:POKEG+54272,0:NEXT  
120 FORG=1544 TO 1564:POKEG,160:POKEG+54272,0:NEXT  
130 FORG=1830 TO 1790 STEP 40:POKEG,160:POKEG+54272,0:NEXT  
140 FORG=1037 TO 1797 STEP 40:POKEG,160:POKEG+54272,0:NEXT
```



This program displays nine boxes that are roughly of the same size and suitable for use in games like Secret Square or Tic-Tac-Toe.

Let's fill the screen with graphics. In order to fill the screen with diamonds, let's input the following program.

```
100 PRINT"☐"  
110 FORZ=1024 TO 2023:POKEZ,90:POKEZ+54272,4:NEXT
```

We like "reversed" diamonds. In order to reverse the character, add **128** to the POKE value. For diamond, the POKE value is **90**; the reverse value is **218**.

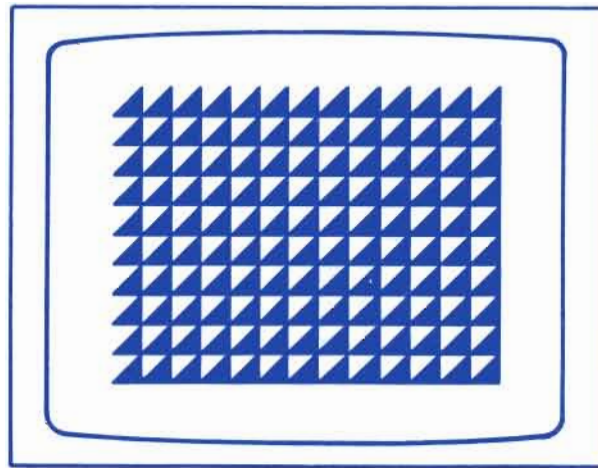
CHARACTER CODES

We encourage you to try all the graphics characters on the keyboard. Here is a little program that makes it easy to experiment. Look in the appendix for the screen codes for each character needed to use this program.

```

10 PRINT "C"
20 INPUT "WHAT CHARACTER CODE":X
30 FOR Z=1024 TO 2023:POKEZ,X:POKEZ+54272,4:NEXT
40 FOR T=1T03000:NEXT
50 GOTO 10

```



Use this program to get an idea of what a full screen of keyboard characters looks like. You may want to use this kind of display as we do in later games. The color chosen for this program was purple — you are welcome to change it.

BAR CHARTS OR GRAPHS

Bar graphs are useful because they compare information graphically. In the following program, we show you how to use bar graphs. Let's pretend you want to compare the acceleration of different cars. How long does it take for a car to reach 55 miles per hour?

68 THE TOOL KIT SERIES Commodore 64 Edition

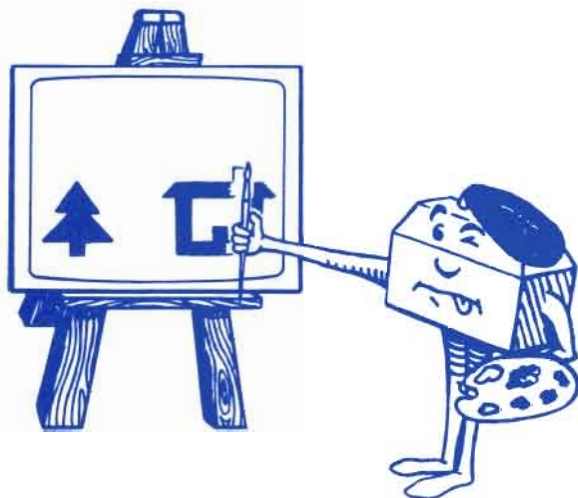
```
10 PRINT"?"
100 PRINT"HOW MANY SECONDS TO REACH 55 MPH?"
110 INPUT "CORVETTE ";A
120 INPUT "TOYOTA ";B
130 INPUT "DODGE ";C
140 INPUT "VOLKS ";D
150 INPUT "MUSTANG ";E
160 INPUT "JEEP ";F
200 PRINT"?"
210 PRINT"COR";
220 X=A:GOSUB 500
230 PRINT"TOY";
240 X=B:GOSUB 500
250 PRINT"DOD";
260 X=C:GOSUB 500
270 PRINT"VW";
280 X=D:GOSUB 500
290 PRINT"MUS";
300 X=E:GOSUB 500
310 PRINT"JEE";
320 X=F:GOSUB 500
340 END
500 FOR Y=1TOX
510 PRINT CHR$(162);
520 NEXT Y
530 PRINT
590 RETURN
```



There is a lot of information you can graph in this way. Try out different graphic characters. We like the way that the solid ball, CHR\$(113), looks in a bar chart.

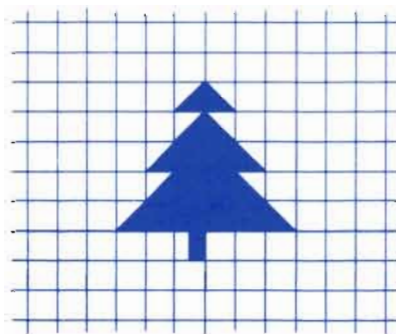
PICTURES WITH THE COMMODORE 64

You can create pictures of almost any type using the Commodore 64. In this next section, we will show you how to put several graphics together to form a scene. We will also add music and a sneak preview of animation. (Animation is covered in the next chapter in detail.)



Pine Tree

Let's draw a pine tree. Using graph paper, we have sketched a tree like this.



The program for POKEing the pine tree is:

```

5 PRINT"☐"
10 Z=1024:C=55296
25 REM**PINE TREE
30 POKEZ+690,233:POKEC+690,5:POKEZ+691,223:POKEC+691,5
40 POKEZ+730,233:POKEC+730,5:POKEZ+731,223:POKEC+731,5
50 POKEZ+769,233:POKEC+769,5:POKEZ+770,160:POKEC+770,5
60 POKEZ+771,160:POKEC+771,5:POKEZ+772,223:POKEC+772,5
70 POKEZ+809,233:POKEC+809,5:POKEZ+810,160:POKEC+810,5
80 POKEZ+811,160:POKEC+811,5:POKEZ+812,223:POKEC+812,5
90 POKEZ+848,233:POKEC+848,5:POKEZ+849,160:POKEC+849,5
100 POKEZ+850,160:POKEC+850,5:POKEZ+851,160:POKEC+851,5
110 POKEZ+852,160:POKEC+852,5
120 POKEZ+853,223:POKEC+853,5:POKEZ+890,118:POKEC+890,0

```

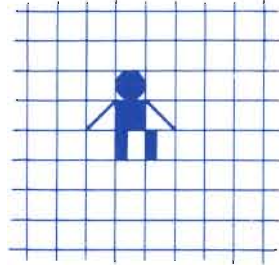
You should have a green tree with a black trunk near the lower left-hand corner of the screen. Save this program because we are going to add to it.

70 THE TOOL KIT SERIES Commodore 64 Edition

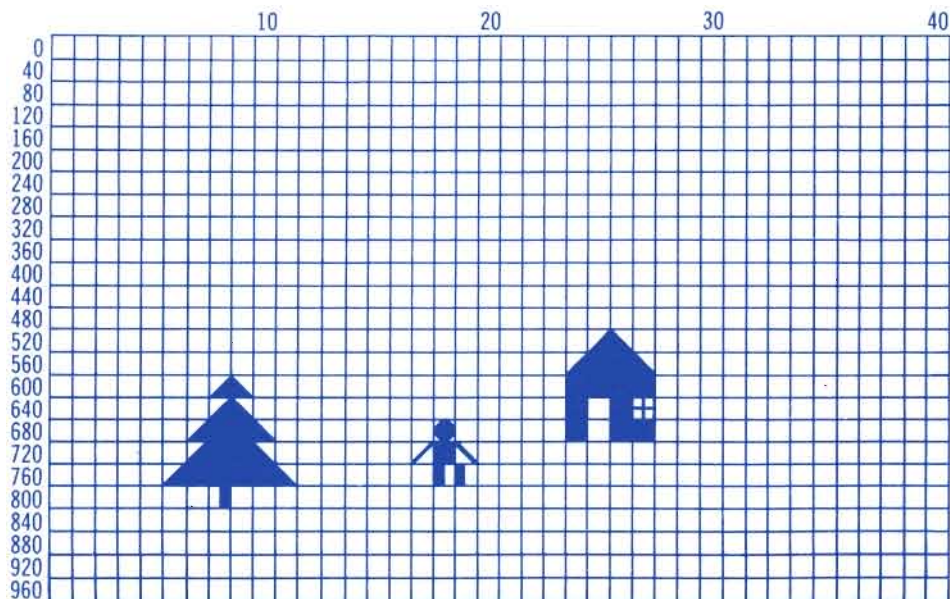
Person

Add the following lines to the program:

```
130 REM**PERSON
140 POKEZ+779,81:POKEC+779,7:POKEZ+818,78:POKEC+818,7
150 POKEZ+819,160:POKEC+819,7:POKEZ+820,77:POKEC+820,7
160 POKEZ+859,117:POKEC+859,7:POKEZ+860,117:POKEC+860,7
```



Now you have a person-type figure located near the bottom center of the screen. Since we are also going to add a house to this “scene,” let us show you how we planned the whole thing. We devised a special graph paper, showing screen and color code memory locations as variables. This is a handy tool and you’re welcome to copy it from the book for your own designing. There is an example of this graph paper given in the appendices.



House

Add the following lines to the program. This will draw a little red house:

```

165 REM**HOUSE
170 POKEZ+748,160:POKEC+748,2:POKEZ+749,219:POKEC+749,1
180 POKEZ+786,160:POKEC+786,2:POKEZ+787,160:POKEC+787,1
190 POKEZ+788,160:POKEC+788,2:POKEZ+789,160:POKEC+789,2
210 POKEZ+706,160:POKEC+706,2:POKEZ+707,160:POKEC+707,2
220 POKEZ+708,160:POKEC+708,2:POKEZ+709,160:POKEC+709,2
230 POKEZ+746,160:POKEC+746,2:POKEZ+747,160:POKEC+747,1
240 POKEZ+627,233:POKEC+627,0:POKEZ+628,223:POKEC+628,0
250 POKEZ+666,233:POKEC+666,0:POKEZ+667,160:POKEC+667,0
260 POKEZ+668,160:POKEC+668,0:POKEZ+669,223:POKEC+669,0

```

We have POKEd a tree, a person, and a house on the screen. Let's make the scene even more interesting. Add the following lines:

```

300 REM**MUSIC AND STARS
310 POKES4296,15:POKE54277,88:POKE54275,15:POKE54274,15:POKE54278,89
320 FOR S=1TO100
330 READH
340 IFH=-1 THEN END
350 READL
360 POKE1024+INT(RND(1)*560),42
370 POKES4273,H:POKES4272,L:POKE54276,65
375 FOR T=1TO100:NEXT
380 FORT=1TO25 :POKE1024+INT(RND(1)*560),32:POKE54276,64:NEXT
390 NEXTS
410 GOTO 330
500 DATA34,75,34,75,51,97,51,97,57,172,57,172,51,97,45,198,45,198
510 DATA43,52,43,52,38,126,38,126,34,75,51,97,51,97,45,198,45,198
520 DATA43,52,43,52,38,126,51,97,51,97,45,198,45,198,43,52,43,52
530 DATA38,126,34,75,34,75,51,97,51,97,57,172,57,172,51,97,45,198
540 DATA45,198,43,52,43,52,38,126,38,126,34,75
550 DATA-1,-1

```

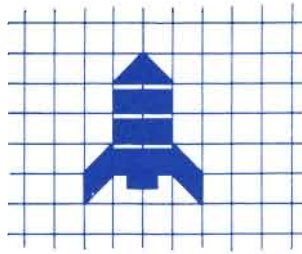
- Line 310 set the volume, attack/decay, sustain/release, and pulse rates for sound.
- Line 320 sends the program through the stars and music generation sequence for a count of 1 to 100.
- Line 360 POKES a star randomly into the first 560 screen locations.
- Line 370 plays the note read from DATA statements.
- Line 380 counts to 25 and POKES a space where the star was (so that it twinkles). The note is also cut off.

Let's start a graphics subroutines library. You probably have some of your own to include.

Spacecraft

Remember the spacecraft we asked you to create using the PRINT command earlier in this chapter? Let's create it with POKE.

72 THE TOOL KIT SERIES Commodore 64 Edition



```
100 REM**SPACECRAFT
110 PRINT"Q"
120 S=1024:C=55296
130 POKES+770,233:POKEC+770,0:POKES+771,223:POKEC+771,0
140 POKES+810,247:POKEC+810,0:POKES+811,247:POKEC+811,0
150 POKES+850,247:POKEC+850,0:POKES+851,247:POKEC+851,0
160 POKES+889,233:POKEC+889,0:POKES+890,247:POKEC+890,0
170 POKES+891,247:POKEC+891,0:POKES+892,223:POKEC+892,0
180 POKES+929,105:POKEC+929,0:POKES+930,124:POKEC+930,0
190 POKES+931,126:POKEC+931,0:POKES+932,95:POKEC+932,0
```

Dice

Now, let's draw some dice that we can use later in our "Traditional Games" chapter.

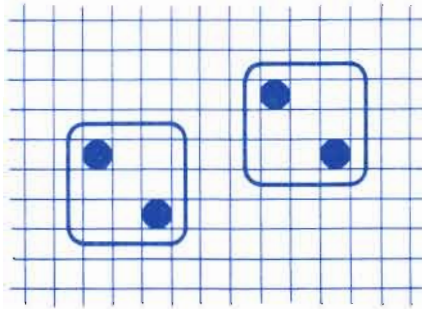
```
6000 REM *** DICE OUTLINE
6010 POKES+579,85:POKEC+579,0:POKES+580,64:POKEC+580,0
6020 POKES+581,64:POKEC+581,0:POKES+582,64:POKEC+582,0
6030 POKES+583,73:POKEC+583,0:POKES+619,66:POKEC+619,0
6040 POKES+623,66:POKEC+623,0:POKES+659,66:POKEC+659,0
6050 POKES+663,66:POKEC+663,0:POKES+699,66:POKEC+699,0
6060 POKES+703,66:POKEC+703,0:POKES+739,74:POKEC+739,0
6070 POKES+740,64:POKEC+740,0:POKES+741,64:POKEC+741,0
6080 POKES+742,64:POKEC+742,0:POKES+743,75:POKEC+743,0
6090 POKES+666,85:POKEC+666,0:POKES+667,64:POKEC+667,0
6100 POKES+668,64:POKEC+668,0:POKES+669,64:POKEC+669,0
6110 POKES+670,73:POKEC+670,0:POKES+706,66:POKEC+706,0
6120 POKES+710,66:POKEC+710,0:POKES+746,66:POKEC+746,0
6130 POKES+750,66:POKEC+750,0:POKES+736,66:POKEC+736,0
6140 POKES+790,66:POKEC+790,0:POKES+826,74:POKEC+826,0
6150 POKES+827,64:POKEC+827,0:POKES+828,64:POKEC+828,0
6160 POKES+829,64:POKEC+829,0:POKES+830,75:POKEC+830,0
```

These lines POKE the outsides of the dice. Now let's put the faces on each die.

```
550 POKES+620,81:POKEC+620,0:POKES+702,81:POKEC+702,0
```

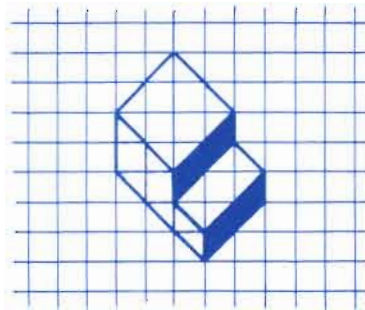
```
850 POKES+707,81:POKEC+707,0:POKES+789,81:POKEC+789,0
```

The dice should look like the following illustration. Now, change the program to show the dice with two sixes, or all the other possible values.



THREE DIMENSIONS

You can POKE in more than two dimensions on the Commodore 64. There is a routine that POKES a nice staircase for you. We planned it like this:



```

10 REM**THREE DIMENSIONS**
20 PRINT"J"
30 S=1024:C=55296
40 POKES+450,78:POKES+451,77:POKES+489,78:POKES+492,77
50 POKES+529,223:POKEC+529,7:POKES+532,233:POKEC+532,2
60 POKES+569,160:POKEC+569,7:POKES+570,223:POKEC+570,7
70 POKES+571,233:POKEC+571,2:POKES+572,105:POKEC+572,2
80 POKES+573,77:POKES+609,95:POKEC+609,7:POKES+610,160:POKEC+610,7
90 POKES+611,105:POKEC+611,2:POKES+613,233:POKEC+613,2
100 POKES+650,95:POKEC+650,7:POKES+651,223:POKEC+651,7
110 POKES+652,233:POKEC+652,2:POKES+653,105:POKEC+653,2
120 POKES+691,95:POKEC+691,7:POKES+692,105:POKEC+692,2

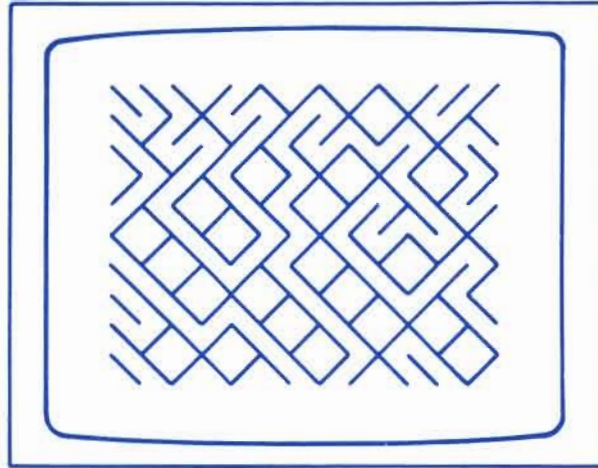
```

MAZE MAKER

Learn a little more about graphics characters by playing with the following program.

74 THE TOOL KIT SERIES Commodore 64 Edition

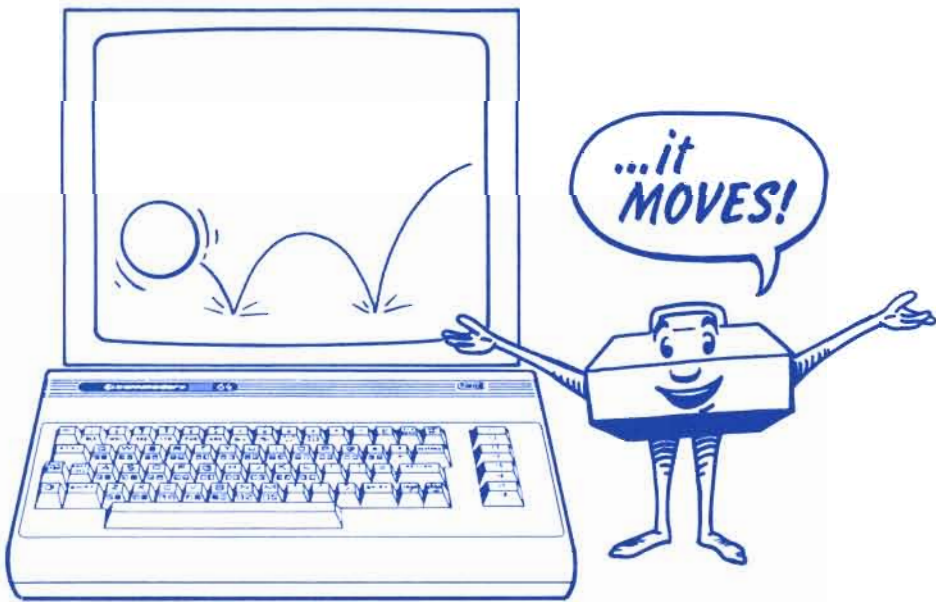
```
10 REM**MAZE MAKER**  
20 PRINT"𐄂"  
30 PRINT CHR$(109.5+RND(1));  
40 GOTO30
```



This displays a maze formed with characters 109 and 110 (l and /). They are picked randomly in the program by line 20.

The more you experiment with graphics, the better you will get! We use some of the subroutines developed in this chapter in some of the later chapters. The twinkling stars on the picture scene gave us a primer for animation. Let's animate our graphics in the next chapter.





Animation Subroutines

6

76 THE TOOL KIT SERIES Commodore 64 Edition

Animation, or the movement of figures on the screen, is a lot of fun. Although it may seem difficult, it really is not. Having mastered the graphics routines in the last chapter, you are now ready to animate.

HOW DOES THE COMPUTER ANIMATE?

Just like Walt Disney Studios. Animation is performed by plotting a character in one position, erasing it, and then plotting the character in another position. The computer completes this operation so quickly and so efficiently that all we see is the character moving along. Because of the speed, we hardly notice the erasing process. As in graphics, we can animate by using the PRINT command or by POKEing screen locations.

ANIMATION USING THE PRINT COMMAND

To illustrate this process, let's make a blinking ball.

```
10 REM**BLINKING BALL**
20 PRINT"␣"
30 PRINT"●"
40 FOR T=1TO200:NEXT
50 PRINT"␣"
60 FOR T=1TO200:NEXT
70 GOTO 30
```

Actually, we have made the ball blink on and off much like the computer blinks the "cursor" for us. If this excites you, try this fluttering heart program.

```
10 PRINT"␣"
20 PRINT"♥"
30 FOR T=1TO300:NEXT
40 PRINT"␣"
50 FOR T=1TO300:NEXT
60 GOTO 20
```

The **S** in lines 20 and 40 sets the cursor at the home position. To get the **S** inside quotation marks (for the PRINT command), you press **CLR/HOME** without pressing **SHIFT**. Since only one character can occupy a place at the same time, each new character "erases" the old one.

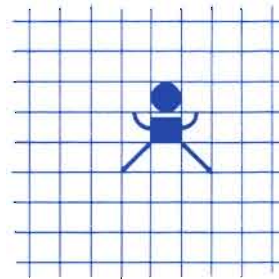
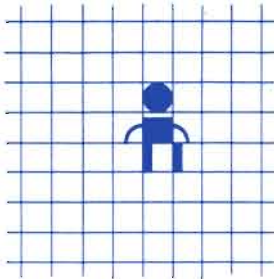
Jumping Jack

Let's use the person we made in the graphics chapter for animation. Type this into your Commodore 64.

```

5 REM**JUMPING JACK**
10 PRINT"□"
20 PRINT" ● "
30 PRINT" /  \ "
40 PRINT" | | "
50 FOR T=1TO500:NEXT
60 PRINT"□"
70 PRINT" ● "
80 PRINT" \  / "
90 PRINT" /  \ "
100 FOR T=1TO500:NEXT
110 GOTO 10

```



Jack exercises on the screen for us. Lines 30 and 80 make use of the Reverse On and Reverse Off keys to make the abdomen for Jack.

Now, let's move our "Jumping Jack" figure across the screen using the PRINT command and the cursor controls.

```

10 PRINT"□"
20 PRINT" ● ████";
25 PRINT"  ████";
30 PRINT"  ████";
40 FOR T=1TO100:NEXT
50 PRINT"  ████";
60 PRINT"□  ████";
65 PRINT"□  ████";
70 FOR T=1TO100
180 GOTO 20

```

- Line 20 prints the head and then moves the cursor left 3 spaces.
- Line 25 moves the cursor down 1 space, prints the arms and body, and then moves the cursor left 3 spaces.
- Line 30 moves the cursor down 1 space, prints the two legs, and then moves the cursor left 3 spaces.
- Lines 50 through 65 erase the figure. The last character inside the quotation marks in line 65 (█) moves the cursor one space to the right, thus moving the whole process across (horizontally) the screen. Note that this character is made by pressing the cursor key while inside " "

Now let's make our figure do some exercises while moving across the screen. Add these lines to your program.

78 THE TOOL KIT SERIES Commodore 64 Edition

```
80 PRINT " ● ■■■■";
90 PRINT "⊗ ⊗ ■■■■";
100 PRINT "⊗ \ \ ■■■■";
120 FOR T=1 TO 100:NEXT
130 PRINT " ■■■■";
140 PRINT "□ ■■■■";
150 PRINT "□ ■■■■";
160 FOR T=1 TO 100:NEXT
```

As you enter these lines, you should notice the repetition. Let's make our program shorter and more efficient by using the GOSUB command.

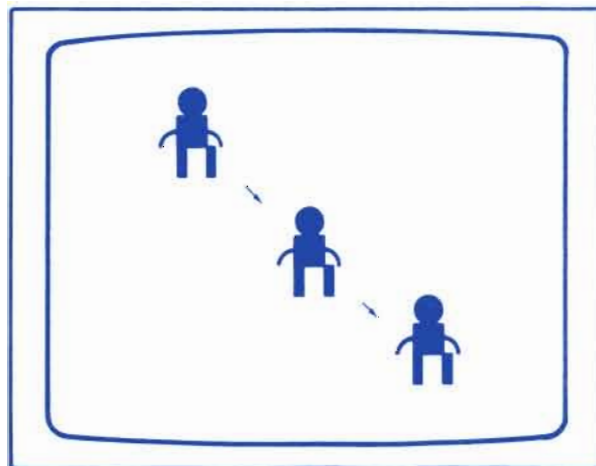
```
10 PRINT "□"
20 PRINT " ● ■■■■";
25 PRINT "⊗ ⊗ ■■■■";
30 PRINT "⊗ \ \ ■■■■";
40 GOSUB 200
80 PRINT " ● ■■■■";
90 PRINT "⊗ ⊗ ■■■■";
100 PRINT "⊗ \ \ ■■■■";
120 GOSUB 200
180 GOTO 20
200 FOR T=1 TO 200:NEXT
210 PRINT " ■■■■";
220 PRINT "□ ■■■■";
230 PRINT "□ ■■■■";
240 FOR T=1 TO 50:NEXT
250 RETURN
```

We changed the timing a little in lines 200 and 240 and the animation looks a little more realistic. How can we move the figure vertically (down the screen)? Change line 230 to:

```
230 PRINT "□ ■■■■";
```

How about diagonally across the screen? Change line 230 to:

```
230 PRINT "□ ■■■■>";
```



USING STRINGS IN ANIMATION

We can save some effort in defining the graphics we want to animate by defining them as string variables. Look at the following snake example.

```

5 REM**SNAKE**
10 PRINT "Q"
20 A$=" | | | | | | | | | |"
30 B$="          ) | | | | |"
50 PRINT A$;
80 GOSUB 100
90 GOTO 20
100 FOR T=1 TO 100:NEXT
110 PRINT B$;
140 FOR T=1 TO 50:NEXT
150 RETURN
    
```

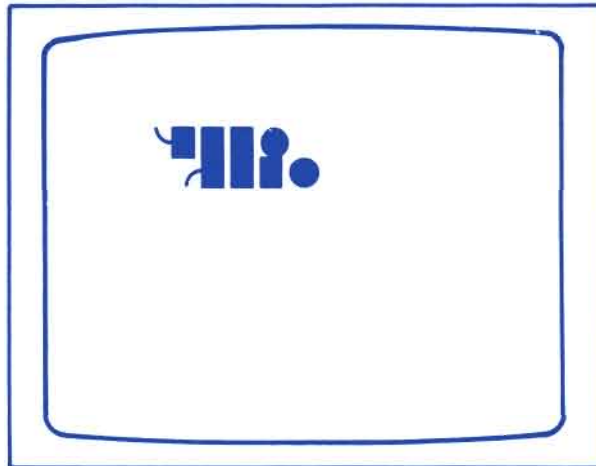
See how easy string variables are to use? Let's make the snake's tail wiggle when it moves. Add these lines.

```

40 C$=" | | | | | | | | | |"
    
```

```

60 GOSUB 100
70 PRINT C$;
    
```



Try to animate your own graphics. How about a flying saucer or a car? Make a cruise missile or a cruise ship move across the screen. We have discussed animation using the PRINT command and now will move to animation using POKE.

80 THE TOOL KIT SERIES Commodore 64 Edition

ANIMATION USING POKE

Let's move a ball around the screen using the POKE statement. Type this in.

```
5 REM**BALL MOVEMENT**
10 PRINT"□"
80 SP=1034:SC=SP+54272
100 FORX=1TO1000 STEP 40
110 POKESP+X,81
120 POKESC+X,0
130 FOR T=1TO100:NEXT
140 POKESP+X,32
150 FORT=1TO100:NEXT
160 NEXT X
200 GOTO100
```

- Line 80 sets SP and SC as the starting screen location for the ball.
- Line 100 moves the ball down the screen (40 spaces at a time and one row at a time) for a count of 1000.
- Lines 110 and 120 POKE a black ball.
- Line 140 POKES a space where the ball was, therefore "erasing" it.

You can change the program to move the ball wherever you want by changing either the starting positions or by changing line 110. Try the following. Change line 100 to:

```
100 FORX=1TO1000 STEP 2
```

The ball moves across the screen in a horizontal fashion. To add some randomness to the ball's position enter this:

```
100 X=INT(RND(1)*1000)+1
```

Moving Jack

We can move a graphic figure like our "Jack" using this method. Try this program:

```
5 REM**MOVING JACK**
10 PRINT"□"
20 SP=1034
30 FOR X=1TO900 STEP40
40 POKESP+X,81
50 POKESP+40+X,102
60 POKESP+39+X,85
70 POKESP+41+X,73
80 POKESP+80+X,116
90 POKESP+81+X,116
160 FOR T=1TO100:NEXT
170 POKESP+X,32
180 POKESP+40+X,32
```

```

190 POKESP+39+X,32
200 POKESP+41+X,32
210 POKESP+80+X,32
220 POKESP+81+X,32
230 FOR T=1T050:NEXT
240 NEXT X
250 GOT030

```

- Line 20 sets SP as the starting place for Jack.
- Line 30 moves Jack down the screen.
- Lines 40 through 90 POKE the “Jack” character.
- Lines 170 through 220 erase Jack by POKEing a blank space.

You could easily change the direction by changing the starting positions and directions in lines 80, 90, and 100.

If we want to control the ball or Jack by testing its location, we can use the following formula to do this:

$$L = 1024 + X + 40 * Y$$

where,

- L = location,
- X = row (from 0 to 39),
- Y = column (from 0 to 24).

Bouncing Ball

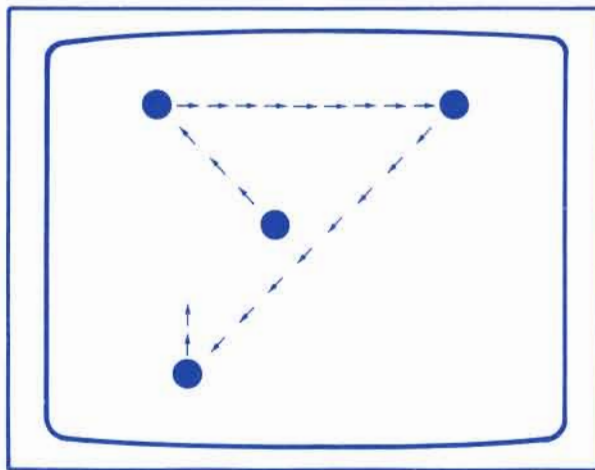
Let’s use the formula in the following program.

```

5 REM**BOUNCING BALL**
10 PRINT"□"
20 X=1:Y=1:ZX=1:ZY=1
30 POKE1024+X+40*Y,81
40 FOR T=1T050:NEXT
50 POKE1024+X+40*Y,32
60 X=X+ZX
70 IFX=0ORX=39 THENZX=-ZX
80 Y=Y+ZY
90 IFY=0ORY=24 THEN ZY=-ZY
100 GOT030

```

So you see how the X and Y variables move the ball about the screen? Lines 70 and 90 tell the ball to reverse its direction if it touches any of the screen borders. If you want to, you can make our Jumping Jack bounce around the screen, also. Pretty strange, but fun to watch.



Speeding Up Jack

We can control and change Jack's speed by "PEEKing" to see if a key has been pressed.

```

10 PRINT"JACK"
20 PRINT" ● ■■■■";
30 PRINT"X / \ ■■■■";
40 PRINT"X | | ■■■■";
50 F1=4:F3=5:X=64
60 Z=PEEK(197)
70 IFZ=X THEN S=150
80 IFZ=F1 THEN S=25
90 IFZ=F3 THEN GOTO 10
100 GOSUB300
120 GOSUB400
130 PRINT" ● ■■■■";
140 PRINT"X / \ ■■■■";
150 PRINT"X | | ■■■■";
160 GOSUB300
170 GOSUB400
180 GOTO 20
300 FOR T=1TO S:NEXT
399 RETURN
400 PRINT" ■■■■";
410 PRINT"J ■■■■";
420 PRINT"J ■■■■";
499 RETURN

```

This is our original "Jumping Jack" program with lines 50 through 90 added, as well as having line 300 broken out as its own subroutine.

- If function key (F1) is pressed, then Jack's speed is increased.
- If no key is pressed, Jack moves at his "normal" speed (T = 1 TO 150).

- If function key (F3) is pressed, then the program is reset with Jack in the original position.

Using these keys allows us control over a program while it is running. This proves very useful in game designs.

Use PEEK for the bouncing ball program or use it for POKEd Jack. Try this logic to control direction as well as speed. Use your imagination and have fun.

Space Commander

You are in control of the spacecraft because you can:

1. Increase the speed by pressing function key 1 (F1).
2. Move the craft to the left by pressing function key 3 (F3).
3. Move the craft to the right by pressing function key 5 (F5).

Key in the following program:

```

5 REM**SPACE COMMANDER**
20 PRINT "J"
70 FORX=0TO-950STEP-40
80 F1=4:F3=5:F5=6:Q=64
90 Z=PEEK(197)
100 IFZ=F1 THEN W=10
110 IFZ=F5 THEN X=X+1
120 IFZ=F3 THEN X=X-1
130 IFZ=Q THEN W=100
140 POKE1914+X,233:POKE1915+X,223:POKE1954+X,106:POKE1955+X,116
150 POKE1994+X,78:POKE1995+X,77
160 GOSUB 300
170 POKE1914+X,32:POKE1915+X,32:POKE1954+X,32:POKE1955+X,32
180 POKE1994+X,32:POKE1995+X,32
190 NEXT X
200 GOTO 70
300 FORK=1TOW:NEXT
310 RETURN

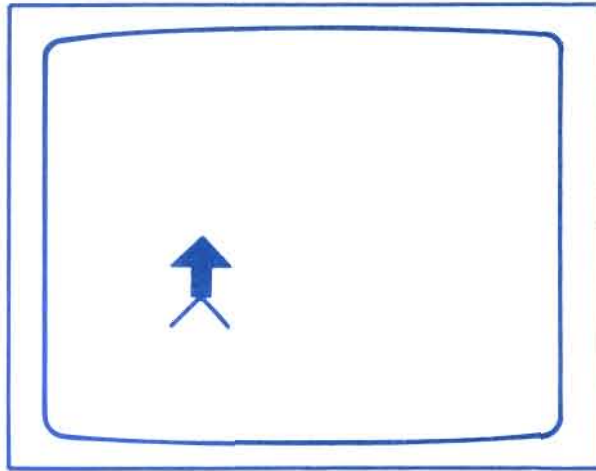
```

- Line 70 sets the movement of the spacecraft up the screen.
- Line 80 defines the function key values.
- Line 90 checks to see if a key is being pressed.
- Lines 100 through 130 define the function keys' effect on speed and side movement.
- Lines 140 and 150 POKE the spacecraft at the screen bottom.
- Lines 170 and 180 erase the spacecraft by POKEing and blank space.
- Subroutine 300 is the timing routine.

You could add a number of asteroids or enemy space stations. Add these lines to the program:


84 THE TOOL KIT SERIES Commodore 64 Edition

```
30 FORD=1T015
40 POKE1024+INT(RND(1)*600),102
50 NEXT D
```



These lines add 15 asteroids which the spacecraft can eliminate by touching them. You control the craft's movement. Can you clear the sky of the pesky asteroids? This animation routine shows you how games are conceived. In order to turn this program into a full-fledged game, we would want to add some sound and color special effects plus a scoring condition.

Duck Hunting With Bow and Arrow

This program will show you how to combine movement with a shooting effect. The hunter is at the bottom of the screen in a boat (the  graphic character). You can move the hunter to the left and right by pressing function keys 1 and 2.

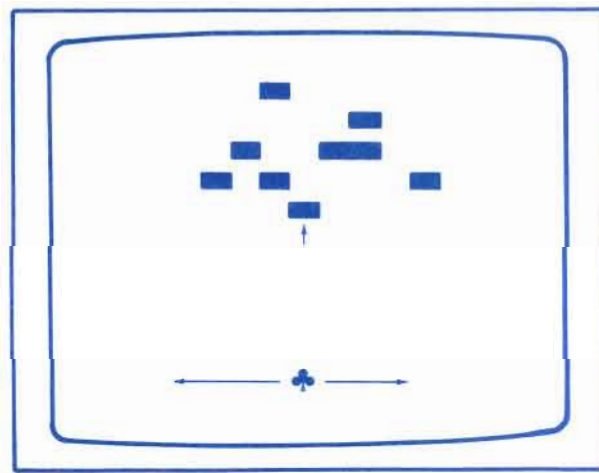
There are 20 ducks in the sky. When you want the hunter to shoot his arrow at the ducks, press the "F" key. An arrow is released and if a duck is hit, it is erased by the arrow.

```
10 REM**HUNTING**
20 F1=4:F3=5:MC=64:F0=21: SX=1024
30 PRINT"♣"
40 F0Y=1T020
50 POKE1024+INT(RND(1)*600),104
60 NEXT Y
70 Y= SX+980
80 S=0:K=Y
```

```

90 X=PEEK(197)
100 IFX=M0 THEN D=0
110 IFX=F1 THEN D=1
120 IFX=F3 THEN D=-1
130 IFX=F0 AND S=0 THEN S=S+1
140 IFS<>0 THEN GOSUB 400
150 POKEY,32
160 Y=Y+D
170 POKEY,88
180 GOTO 90
400 IF S=1 THEN K=Y-40:POKEK,30:S=2:RETURN
410 POKEK,32
415 IF S=25 THEN S=0:RETURN
420 K=K-40:S=S+1
430 POKEK,30
490 RETURN

```



- Line 20 sets the values for keys to be pressed.
- Lines 40 through 60 POKE 20 ducks randomly in the sky in the first 600 screen locations.
- Line 70 sets the starting location for the boat and hunter at 2004 (1024 + 980).
- Line 80 sets S and a variable that we use in shooting at 0.
- Line 90 asks if a key is being pressed.
- Line 100 states that if no key is pressed, then the direction is unchanged; $D = 0$.
- Line 110 states that if function key 1 is pressed, the hunter is moved to the right.
- Line 120 states that if function key 3 is pressed, hunter is moved to the left.
- Line 130 states that if the FIRE key is pressed, "F", then S is set at $S + 1$.
- Line 140 states that if S is not equal to 0, then the shooting subroutine for the arrow begins.

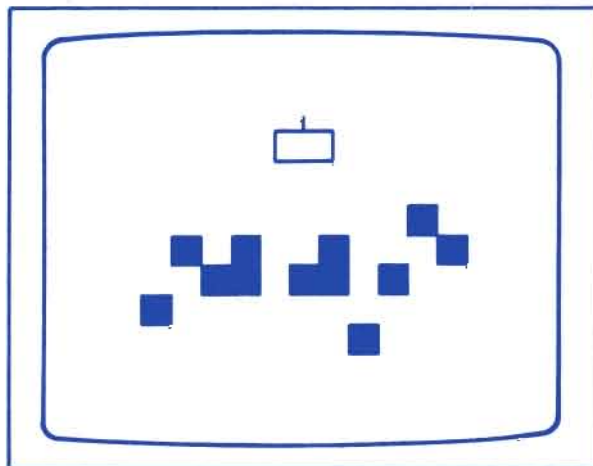
86 THE TOOL KIT SERIES Commodore 64 Edition

- Line 150 POKEs a blank space for screen location, Y.
- Line 160 adds the value of D to Y.
- Line 170 POKEs the hunter in the boat on the screen.
- Line 180 sends the program to line 90 to see if any keys are being pressed.
- Line 400 states that if S equals 1, then screen location K (which was defined as being the same as 2004 in line 80) is reduced by 40. The arrow is POKEd, S is increased in value, and the subroutine returns to the main program.
- Line 410 states that the arrow is erased.
- Line 415 says if $S = 25$ (the arrow has left the screen), the value of S is set at 0 and we are returned to the main program.
- Line 420 says for every movement of the arrow, location K is decreased by 40 and S is increased by 1.
- Line 430 causes the arrow to be POKEd on the screen.

The use of the S variable allows the program to alternate movement from the hunter to the arrow. Since no two characters can move on the Commodore 64 at the same time using POKE, we alternate their movement with the S variable in order to make animation more realistic. We could make our graphics more elaborate and colorful and could also add sound to develop this into a challenging game of marksmanship.

Mars Landing

Let's practice landing a flying saucer on the surface of Mars. You control the saucer by using function keys 1, 3, and 5.




```

5 REM**MARS LANDING**
10 PRINT" ":POKE53281,0
20 GOSUB 400
30 X=0
40 FOR Y=0 TO 23 STEP 1
50 F1=4:F3=5:F5=6:Z0=64
60 K=PEEK(197)
70 IFK=F1 THEN X=X+1
80 IFK=Z0 THEN Q=40
90 IFK=F3 THEN X=X-1
100 IFK=F5 THEN Q=220
110 IF PEEK(1084+X+40*Y)=102 THEN GOSUB 800
120 IF PEEK(1085+X+40*Y)=102 THEN GOSUB 800
130 IF PEEK(1086+X+40*Y)=102 THEN GOSUB 800
140 POKE1044+X+40*Y,112:POKE1045+X+40*Y,113:POKE1046+X+40*Y,110
150 POKE1084+X+40*Y,109:POKE1085+X+40*Y,67:POKE1086+X+40*Y,125
160 GOSUB 300
170 POKE1044+X+40*Y,32:POKE1045+X+40*Y,32:POKE1046+X+40*Y,32
180 POKE1084+X+40*Y,32:POKE1085+X+40*Y,32:POKE1086+X+40*Y,32
190 NEXT Y
200 GOTO 10
300 FOR T=1 TO Q:NEXT
399 RETURN
400 FOR K=1 TO 250
410 M=1024+INT(RND(1)*990)
420 IFM>1864 THEN POKEM,102
430 NEXT K
499 RETURN
800 REM**HIT MARS SURFACE
810 POKES4296,15:POKE54273,61:POKE54272,126:POKE54277,9
820 POKE54276,65:POKE54275,15:POKE54274,150
830 POKE53281,2
840 FORT=1 TO 100:NEXT
850 POKE54276,64
860 POKE53281,0
899 RETURN

```

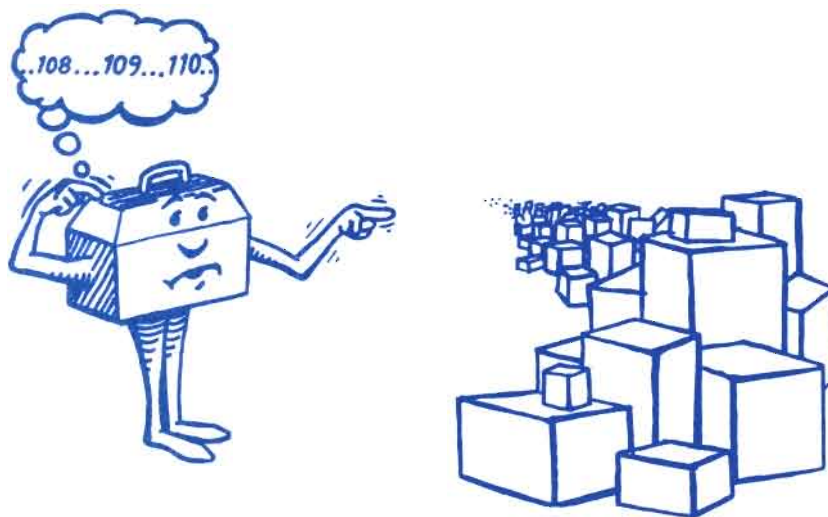
- Line 10 clears the screen and POKEs the background color black.
- Line 30 sets X at 0. X is the column variable.
- Line 40 sets Y from 0 to 23, the number of rows we want the saucer to move down the screen.
- Line 50 sets values for the function keys and if no key is pressed (20).
- Line 60 asks if a key is being pressed.
- Line 70 states that if function key 1 is pressed, the saucer is moved one space to the right.
- Line 80 states that if no key is pressed, the saucer's speed is set at the normal rate ($T = 1$ to 40).
- Line 90 states that if function key 3 is pressed, the saucer is moved one space to the left.
- Line 100 states that if function key 5 is pressed, the saucer is slowed down.
- Lines 110 through 130 test to see if the lower part of the saucer touches the Mars surface. If it does, then the program is sent to Subroutine 800.
- Lines 140 and 150 POKE the saucer graphic.
- Lines 170 and 180 erase the saucer by POKeing spaces.
- Subroutine 300 is the timing subroutine.

88 THE TOOL KIT SERIES Commodore 64 Edition

- Subroutine 400 POKEs the surface obstacles of Mars in the lower part of the screen.
- Subroutine 800 flashes the screen red and beeps for every Mars surface obstacle hit.

Use your own imagination to develop animation routines for the Commodore 64. You can set almost any graphics character to animation, using color and sound to make the sight more exciting. By PEEKing where one character is in relation to another, we can learn the basis for arcade game development.





Calculating Subroutines: Facts, Figures, and Conversions

7

90 THE TOOL KIT SERIES Commodore 64 Edition

Your Commodore 64 can easily become a helpful calculator and problem-solver. We think it is smart to use your computer in this way, and you'll be learning how to develop subroutines for fun math games and quizzes in the next chapter.

Don't be anxious about math, because our gentle approach to it will show you how easy arithmetic and algebra is on the Commodore 64. Let's start by reviewing some simple arithmetic operations.

SIMPLE CALCULATIONS

All you have to do to perform simple calculations on the Commodore 64 is use the PRINT statement. Try it.

PRINT 2 + 2	(and press RETURN)
PRINT 5 - 2	(and press RETURN)
PRINT 5*2	(The multiplication sign is the asterisk *.)
PRINT 6/2	
PRINT 4 ↑ 2	(The ↑ is used for exponents. Thus, 4 ↑ 2 is the same as 4 ² .)
PRINT 10*(2/5)	(The parentheses show the computer which operation to perform first.)

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

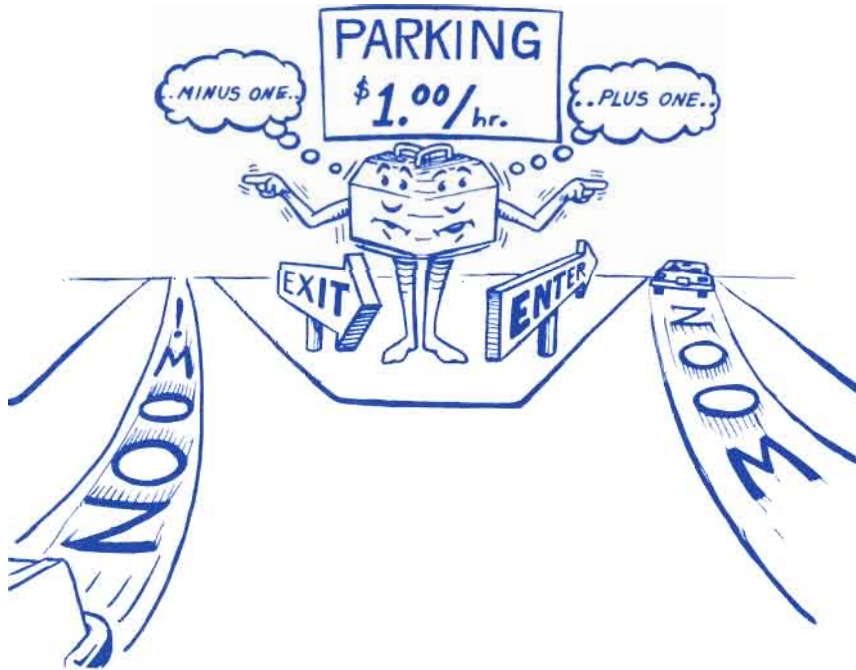
exponent	↑ (highest priority)
multiplication	*
division	/
addition	+
subtraction	- (lowest priority)

Let's look at some examples to see how you and the Commodore 64 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 20*15  
  
300
```



There are 300 cars.

In the same full 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 this line:

```
PRINT 300 - (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 arrive at the same rate? How many would there be in 4 hours?

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

92 THE TOOL KIT SERIES Commodore 64 Edition

There are other numeric functions your Commodore 64 can perform. We encourage you to learn about square roots, logarithms, cosines and the like in the manual that came with the 64.

CALCULATING SUBROUTINES

We're going to show you how to solve problems by using short programs, or subroutines, on the Commodore 64.

Gallons and Liters



First, let's look at this problem:

If one gallon is the same as .2642 liters, how many gallons are 60 liters?

We can solve this problem by multiplying 60 liters times .2642 to get the correct number of gallons. Using the PRINT statement, this would be:

```
PRINT 60*.2642
```

If we want to convert many different amounts of liters into gallons, we can write a program.

L = liters
G = gallons

```
10 INPUT L  
20 LET G=.2642*L  
30 PRINT G
```

- Line 10 asks you to input a value for L.
- Line 20 defines G as a function of L or in relation to L.
- Line 30 tells the computer to print the value of G.

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

Let's make the program more "friendly." Change line 10 and 30 to read:

```
10 INPUT "# OF LITERS";L
30 PRINT L" LITERS IS THE SAME AS "G" GALLONS"
```

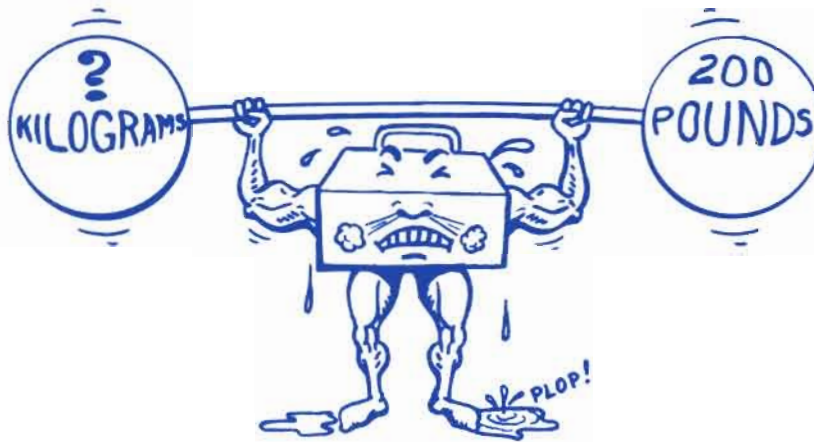
Now the program asks you to input the number of liters and it displays for you in plain English the liters-to-gallons conversion.

Kilometers and Miles

PROBLEM 2:

Write a subroutine for converting kilometers from miles. For every kilometer, there are 0.621 miles.

Pounds, Ounces, and Kilograms



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

1 pound = 2.205 kilograms

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

```
5 PRINT"?"
10 PRINT "FIRST POUNDS, THEN OUNCES"
20 INPUT P,OZ
30 LET PZ=P+OZ/16
40 KG=PZ/2.205
50 PRINT PZ" POUNDS ARE "KG" KILOGRAMS"
```

94 THE TOOL KIT SERIES Commodore 64 Edition

- Line 5 starts the program with a clear screen.
- Line 10 instructs you to input first pounds, and then ounces.
- Line 20 asks you to give the values for P and OZ.
- Line 30 defines PZ in relation to pounds and ounces (P and OZ).
- Line 40 defines KG in relation to PZ (pounds).
- Line 50 tells the computer to print pounds and kilograms for the values you have assigned.

Inches and Centimeters

We know that 1 inch equals 2.54 centimeters. Let's write the simple subroutine for solving centimeters:

```
10 INPUT "HOW MANY INCHES";I
20 LET CM=2.54*I
30 PRINT I" INCHES IS APPROXIMATELY "CM"CENTIMETERS"
```

PROBLEM 3:

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

Temperature

Converting from Fahrenheit to Celsius is not easy, but this subroutine should help:

```
10 INPUT "DEGREES FAHRENHEIT";F
20 LET C=.55556*(F-32)
30 PRINT F" DEGREES FAHRENHEIT IS "C" DEGREES CELSIUS"
40 GOTO10
```

- Line 40 tells the computer to go back to line 10 and prompt you for another Fahrenheit value. This line turns the program into a continuous loop. The way to stop it is to key in **RUN/STOP** and **RESTORE** at the same time.

Dollars and Deutschmarks

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


```

10 PRINT "C"
20 PRINT "DOLLARS", "MARKS"
30 D=0
40 PRINT D, 2.55*D
50 D=D+5
60 IF D<30 THEN 40
70 STOP
    
```

- Line 10 starts the program on a clear screen.
- Line 20 tells the computer to divide the screen in half (because of the comma) and put dollars on one side, marks on the other.
- Line 30 starts D at 0.
- Line 40 tells the computer to put the value of D on the left and 2.55*D on the right.
- Line 50 tells the computer to add 5 to its value.
- Line 60 says that as long as D is less than 30, print D and 2.55 times D.
- Line 70 tells the program to stop.

DOLLARS	MARKS
0	0
5	12.75
10	25.5
15	38.25
20	51
25	63.75

Type the program into the computer and RUN it. Do you like the table format? What happens when you change line 60 to **IF D < 50 THEN 40**? How about changing the increment in line 50 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 it? This program shows you how easy it is to make money when you have some. The standard formula for return is:

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

where,

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

```
10 PRINT"☺"  
20 PRINT"AMOUNT OF MONEY TO INVEST":INPUT M  
30 PRINT"WHAT IS THE INTEREST RATE PER YEAR":INPUT I  
40 PRINT "HOW MANY YEARS":INPUT Y  
50 LET R=M*(1 +I/100)^Y  
60 PRINT "YOUR RETURN ON "M" INVESTED IS "R
```

None of the lines used in this subroutine are new except for lines 20 and 30, which combine the PRINT and INPUT statements using the colon.

This is a fun subroutine to play with. We suggest you add another line so you don't have to type in NEW after each run.

```
70 GOTO 20
```

These continuous loops are easy to make. What would happen if you had typed in

```
70 GOTO 10
```

Profit

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

$$\text{Profit} = \text{Total Revenues} - \text{Total Costs}$$

Now let's do a "profitable" subroutine.

```
10 PRINT " "
20 PRINT "WHAT WERE YOUR TOTAL REVENUES": INPUT R
30 PRINT "WHAT WERE YOUR TOTAL COSTS": INPUT C
40 LET P=R-C
50 PRINT P " IS YOUR TOTAL PROFIT"
```

Easy enough. 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 performance of various businesses can be compared regardless of their size. We have to add the following lines to get this ratio.

```
60 LET PP=F/R*100
70 PRINT PP " IS YOUR PROFIT AS A PERCENTAGE OF REVENUES"
```

- Line 60 defines PP (Profit as a % of Revenue) as Profit divided by Revenue times 100.
- Line 70 simply prints the value of PP with a message. Now RUN the whole program.

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:

$$\text{Current Ratio} = \text{Current Assets} / \text{Current Liabilities}$$

PROBLEM 6:

Write a subroutine for figuring current ratio.

Loan Analysis

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

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.

```
10 PRINT "Q"
20 PRINT "WHAT ARE YOUR CURRENT ASSETS": INPUT A
30 PRINT "WHAT ARE YOUR CURRENT LIABILITIES": INPUT L
40 LET R=A/L
50 IF R>1 THEN 100
60 IF R<=1 THEN 200
100 PRINT "YOUR CURRENT RATIO IS "R" YOUR LOAN IS ACCEPTED"
105 STOP
200 PRINT "YOUR CURRENT RATIO IS "R" YOUR LOAN IS DENIED"
205 STOP
```

- Line 50 tells the computer that if the current ratio (R) is greater than 1, it is to go to line 100.
- Line 100 prints the current ratio and tells you that the loan is accepted.
- Line 60 tells the computer that if R is less than or equal to 1, it is to go to line 200.
- Line 200 prints the current ratio and tells you that the loan is denied.

PROBLEM 7:

Using what you have learned in this Loan Analysis case, go back to the Profit subroutine and add lines to the program so that:

- If the percentage of profit is more than 20%, "the company is in good health."
- If the percentage of profit is less than 20%, "the company is performing poorly."

ANSWERS TO CHAPTER PROBLEMS

Problem 1

Parking Lot

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

Problem 2

Kilometers and Miles

```
10 INPUT "MILES";M
20 LET KM=0.621*M
30 PRINT M " MILES EQUAL"KM" KILOMETERS"
```

Problem 3

Feet, Inches, and Centimeters

```
10 PRINT "FIRST FEET, THEN INCHES"
20 INPUT F,I
30 LET FI=I+(F*12)
40 LET CM=2.54*FI
50 PRINT FI" INCHES IS APPROXIMATELY "CM" CENTIMETERS"
```

Problem 4

Exchange Rate

```
10 INPUT "EXCHANGE RATE";EX
20 PRINT"DOLLARS", "MARKS"
30 D=0
40 PRINT D,EX*D
50 D=D+5
60 IF D<30 THEN 40
70 STOP
```

Problem 5

Temperature Table

```
10 PRINT "C"
20 PRINT "FAHRENHEIT", "CELSIUS"
30 F=0
40 PRINT F,.55556*(F-32)
50 F=F+20
60 IF F<=220 THEN 40
70 STOP
```

Problem 6

Current Ratio

```
10 PRINT "C"
20 PRINT "WHAT ARE YOUR CURRENT ASSETS":INPUT A
30 PRINT "WHAT ARE YOUR CURRENT LIABILITIES":INPUT L
40 LET R=A/L
50 PRINT R " IS YOUR CURRENT RATIO"
```

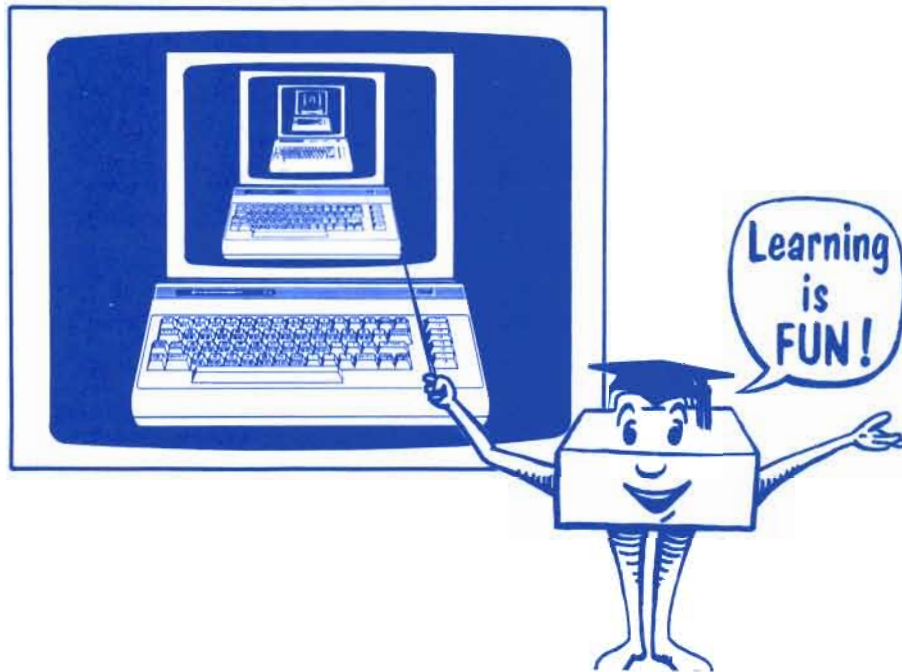
100 THE TOOL KIT SERIES Commodore 64 Edition

Problem 7

Profit Analysis

```
70 IF PP>=20 THEN 100
80 IF PP<20 THEN 200
100 PRINT "YOUR PERCENTAGE PROFIT IS "PP" AND YOUR COMPANY IS IN GOOD HEALTH"
105 STOP
200 PRINT "YOUR PERCENTAGE PROFIT IS "PP" AND THE COMPANY IS PERFORMING POORLY"
205 STOP
```





Educational Games

8

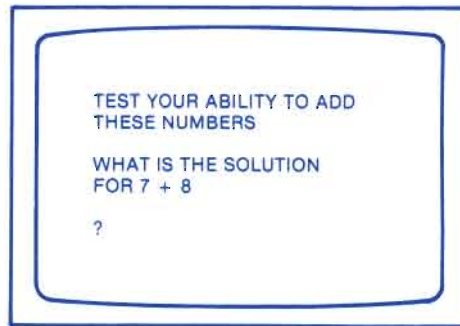
102 THE TOOL KIT SERIES Commodore 64 Edition

The educational games in this chapter are designed so that you can have fun while learning.

We will explain the structure of each program and we encourage you to improve and enhance it with graphics, color, and sound, or whatever 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 their parents about all kinds of things, from States and Capitals to Riddles; we hope you have as good a time with these games as we do.

ARITHMETIC TESTER



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.

```
10 PRINT "☺"
20 PRINT "TEST YOUR ABILITY TO ADD THESE NUMBERS"
30 Q=0
40 N=0
50 A=INT(RND(0)*11)
60 B=INT(RND(0)*11)
70 IF Q=10 THEN GOTO 240
80 Q=Q+1
90 PRINT "☺"
100 PRINT "WHAT IS THE SOLUTION FOR "A" PLUS "B
110 PRINT
120 INPUT X
130 PRINT
140 IF X=A+B THEN 190
150 PRINT "WRONG"
160 PRINT A+B" IS THE CORRECT ANSWER"
170 FOR T=1TO1000:NEXT T
180 GOTO 220
190 PRINT "CORRECT"
200 FOR T=1TO1000:NEXT T
210 N=N+1
220 PRINT
230 GOTO 50
240 GOTO 250
250 PRINT N" QUESTIONS RIGHT OF "Q
```



```
260 J=N/Q
270 IF J=1 THEN GOSUB 3000
280 IF J<1 AND J>=.7 THEN GOSUB 4000
290 IF J<.7 THEN GOSUB 5000
300 END
3000 PRINT "VERY GOOD WORK"
3500 RETURN
4000 PRINT "GOOD WORK, BUT MORE PRACTICE WILL HELP"
4500 RETURN
5000 PRINT "YOU NEED TO PRACTICE ALOT"
5500 RETURN
```

- Line 30 starts the number of questions asked at 0.
- Line 40 starts the number of correct questions at 0.
- Lines 50 and 60 state A and B are integers (or whole numbers) randomly selected from 1 to 10.
- Line 70 tells the program to go to line 240 if 10 questions have been asked.
- Line 80 adds 1 to the number of questions asked.
- Line 100 prints A + B in question form.
- Line 120 asks you to input the answer.
- Line 140 checks to see if the answer is correct. Otherwise, the program moves to the “wrong” answer part of the program.
- Line 160 prints the correct answer.
- Line 190 tells you that your answer is correct. Remember that line 140 did the checking.
- Line 210 counts the number correct for all the questions asked.
- Line 250 prints the score.
- Line 260 turns the score into a ratio of the number correct out of the number asked.
- Lines 270 through 290 tell the computer which subroutine to go to, depending upon the percentage answered correctly.

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 Subroutines 3000-5000.

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

104 THE TOOL KIT SERIES Commodore 64 Edition

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

1. **LEVEL** — Change the level of difficulty by changing lines 50 and 60. To make the quiz harder, change the statement to read:

```
INT(RND(1) * 100) + 1
```

for both A and B. Now you will have addition problems of numbers 1 to 100.

2. **LENGTH** — You can make the quiz longer by changing the number of questions asked before the program ends. Adapt line 70 to:

```
IF Q = 20 THEN GOTO 240
```

Now you have 20 questions. You could change it to anything — even 1000 or more.

3. **GRADING** — You can set your own grades or competency levels by changing lines 270-290. If you would rather have “very good work” mean .9 (or 90%) and higher, change the lines to read:

```
270 IF J >= .9 THEN GOSUB 3000  
280 IF J < .9 AND J >= .7 THEN GOSUB 4000  
290 IF J < .7 THEN GOSUB 5000
```

4. **OPERATION** — You can change this from addition to any other arithmetic operation you want, such as subtraction, division, and multiplication. How? For subtraction, change (A + B) to (A - B) in lines 140 and 160. Change the PRINT statement in line 20 to read SUBTRACT and in line 100 to read MINUS, and you’re done.

MULTIPLICATION TABLES

Using the same program structure as “Arithmetic Tester” and using READ-DATA statements, this program tests your ability to multiply the numbers provided in DATA statements in the program. The number of questions asked is the same as the amount of data. When your answers are all correct, the computer smiles for you.

```
10 PRINT "Q"  
20 PRINT "THIS PROGRAM WILL HELP YOU LEARN YOUR MULTIPLICATION TABLES--HAVE FUN"  
30 Q=0  
40 N=0  
50 READ A  
60 IF A=99 THEN 240  
70 READ B  
80 Q=Q+1
```

```

90 PRINT "J"
100 PRINT "WHAT IS THE ANSWER FOR "A" TIMES "B"
110 PRINT
120 INPUT X
130 PRINT
140 IF X=A*B THEN 190
150 PRINT "WRONG"
160 PRINT A*B" IS THE RIGHT ANSWER"
170 FOR T=1TO1000:NEXT T
180 GOTO 220
190 PRINT "CORRECT"
200 FOR T=1TO1000:NEXT T
210 N=N+1
220 PRINT
230 GOTO 50
240 GOTO250
250 PRINTN" QUESTIONS RIGHT OUT OF "Q
260 J=N/Q
270 IF J=1THEN GOSUB 3000
280 IF J<1 AND J>=.6 THEN GOSUB 4000
290 IF J<.6 THEN GOSUB 5000
500 DATA11,3,12,6,15,9,14,4,10,11,13,12,15,6,11,12,13,5,7,14
700 DATA99
1000 END
3000 PRINT "EXCELLENT WORK"
3010 C=54272
3020 POKE1724,87:POKE1724+C,7:POKE1726,87:POKE1726+C,7
3030 POKE1765,81:POKE1765+C,7:POKE1804,77:POKE1804+C,7
3040 POKE1805,100:POKE1805+C,7:POKE1806,78:POKE1806+C,7
3050 RETURN
4000 PRINT"GOOD WORK. PRACTICE MAKES PERFECT"
4050 RETURN
5000 PRINT "POOR WORK. PRACTICE MORE"
5050 RETURN

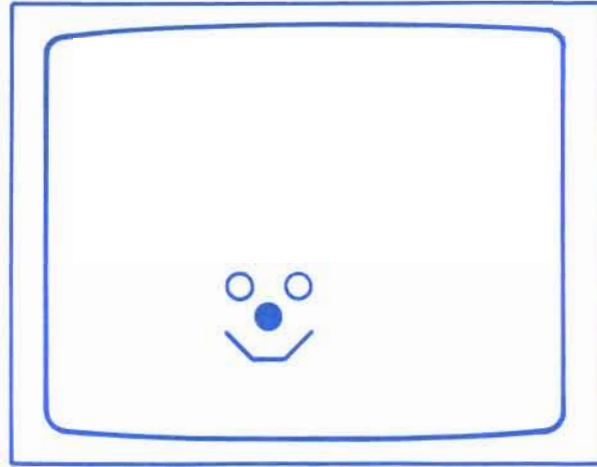
```



- Line 50 tells the computer to read the DATA in A (11, 12, 15, 14, etc.). These are numeric variables.
- Line 60 tells the computer to go to the end of the program when all the DATA are used.
- Line 70 reads DATA in B (3, 6, 9, etc.).

Suggested Program Changes and “Dressing”

Add sound, color, and more graphics to the subroutines. We have used a smiling face to encourage you to be creative. Use a frowning face for Subrou-



tine 5000. Can you make the eye wink? Use the Random Color program in the “Commodore Color” chapter for good scores. You want music? Look at what you learned in the “Sound and Music Subroutines” of Chapter 4 for some fun routines. Change the programs in these ways to suit you:

- Increase or decrease the level of difficulty of the problems by changing the DATA statements.
- Add more data to make the program as long as you wish.
- Change the scoring any way you wish. Can you think of a way to use a “point system” rather than saying “the number correct out of the number asked?”
- Change the operation to addition, division, or subtraction.

SHAPE RECOGNITION GAME

This game was written to test our children’s shape recognition capabilities. The player decides if the shape drawn on the screen is a circle, square, triangle, or rectangle.

```
10 REM *** SHAPE RECOGNITION GAME
20 GOSUB 1000
30 Q= INT(RND(1)*4)+1
50 IF Q=1 THEN GOSUB 400
60 IF Q=2 THEN GOSUB 500
70 IF Q=3 THEN GOSUB 600
```

```

80 IF Q=4 THEN GOSUB 700
120 PRINT:PRINT"IS THIS A CIRCLE, SQUARE, TRIANGLE OR      RECTANGLE?":PRINT
130 INPUT X$
150 IF Q=1 AND X$="CIRCLE" THEN GOSUB 800
160 IF Q=1 AND X$<>"CIRCLE" THEN GOSUB 900
170 IF Q=2 AND X$="SQUARE" THEN GOSUB 800
180 IF Q=2 AND X$<>"SQUARE" THEN GOSUB 900
190 IF Q=3 AND X$="TRIANGLE" THEN GOSUB 800
200 IF Q=3 AND X$<>"TRIANGLE" THEN GOSUB 900
210 IF Q=4 AND X$="RECTANGLE" THEN GOSUB 800
220 IF Q=4 AND X$<>"RECTANGLE" THEN GOSUB 900
250 GOTO 30
400 REM**CIRCLE SUBROUTINE
410 PRINT"○○○○○○○∩"
499 RETURN
500 REM**SQUARE SUBROUTINE
510 PRINT"□○○○○○○□"
599 RETURN
600 REM**TRIANGLE SUBROUTINE
610 PRINT"△○○○○○○∧"
620 PRINT"∨□□□□□"
699 RETURN
700 REM**RECTANGLE SUBROUTINE
710 PRINT"□○○○○○○□"
799 RETURN
800 REM** CORRECT ANSWER SUBROUTINE
850 PRINT"○○○○○○YES! YOU ARE RIGHT!"
890 FOR T=1TO1000:NEXT
899 RETURN
900 REM** WRONG ANSWER SUBROUTINE
910 PRINT "○○○○○○WRONG☹"
915 IF Q=1 THEN PRINT"THAT WAS A CIRCLE"
920 IF Q=2 THEN PRINT"THAT WAS A SQUARE"
930 IF Q=3 THEN PRINT "THAT WAS A TRIANGLE"
940 IF Q=4 THEN PRINT "THAT WAS A RECTANGLE"
950 FOR T=1TO1500:NEXT
990 GOTO 50
999 RETURN
1000 REM**OPENING FRAME
1005 PRINT"○○○○○○WELCOME TO SHAPE RECOGNITION"
1010 FOR T=1 TO 10:PRINT"○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○○":NEXT
1050 FOR T=1TO2000:NEXT
1099 RETURN

```

- Line 30 defines Q as a random integer from 1 to 4.
- Lines 50 through 80 send the program to the shape POKEing subroutines, depending upon what value Q is.
- Line 130 asks for the shape.
- Lines 150 through 220 send the program to the right and wrong answer subroutines, based on the input.

You could simplify this game for a child by defining the input necessary as the first letter of each shape. For example, use "T" for Triangle, "C" for Circle, and so on. We, personally, have also taped a drawing of the shape on the appropriate keys when using this game for toddlers.

RHYMING RECOGNITION GAME

This game uses DATA statements to test the player's ability to match words that rhyme.

108 THE TOOL KIT SERIES Commodore 64 Edition

```
5 REM- RHYMING RECOGNITION
100 Q=0:N=0
110 READ A$
120 IF A$="FINISH" THEN GOTO 300
125 READ B$:READ C$:READ D$:READ E$
130 Q=Q+1
140 PRINT "XXXXX"A$
150 PRINT "XXXXXXXXXX"B$
160 PRINT"XXXXXXXXXX"C$
170 PRINT"XXXXXXXXXX"D$
180 PRINT"XXXXX"
190 INPUT X$
200 IF X$=E$ THEN GOSUB 5000
210 IF X$<>E$ THEN GOSUB 6000
240 GOTO 110
300 END
1000 DATA CAT,PUT,SAT,SIT,SAT,LOOK,TAKE,LAKE,BOOK,BOOK
1005 DATA LOVE,LEAVE,DOVE,LIVE,DOVE
1010 DATA WALK,WAKE,TALK,TAKE,TALK,MAN,CAN,MINE,AND,CAN
1020 DATA END,SEND,AND,BIN,SEND
1030 DATA FINISH
5000 REM CORRECT ANSWER ROUTINE
5100 PRINT "YOU ARE RIGHT!"
5200 POKE54296,15
5250 POKE54277,9
5300 POKE54273,33:POKE54272,135
5350 POKE54276,17
5400 FOR T=1TO1000:NEXT
5450 POKE54276,16
5990 RETURN
6000 REM WRONG ANSWER ROUTINE
6100 PRINT "YOU ARE WRONG!"
6350 POKE54296,15
6400 POKE54277,0:POKE54278,240
6450 POKE54273,12:POKE54272,143
6500 POKE54276,17
6550 FOR T=1TO1000:NEXT
6600 POKE54276,16
6700 GOTO 140
6900 RETURN
```

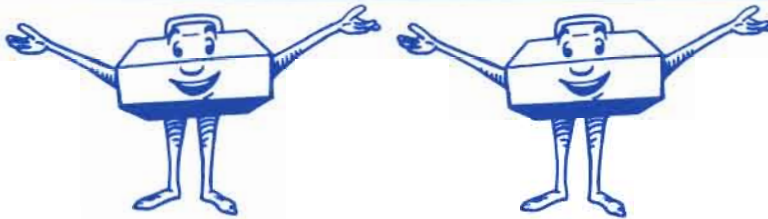
- Line 140 prints the word with which one of the three choices will rhyme.
- Lines 150 through 170 print the choices for the key word.
- Line 200 states that if the answer inputted is the same as the correct answer (E\$), then the correct answer subroutine is called.
- Line 210 states that if the inputted answer is different than the correct answer (E\$), then the wrong answer subroutine is called.

This program could be enhanced with color and graphics. A scoring subroutine could be included. Lines 100 and 130 set this up (Q as the number of questions, N as the number correct). The number of words tested can be increased by simply adding more DATA statements.

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. You have to use your imagination to answer the riddle! For example:

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



110 THE TOOL KIT SERIES Commodore 64 Edition

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 keep asking you *forever* using the hint until you finally get the riddle — just like your friends do when they tell you a riddle!

```
10 READ A$
20 IF A$="END" THEN STOP
30 READB$:READC$:READD$:READE$:READF$:READG$:READH$:READI$:READJ$:READK$
40 READL$:READM$:READN$
100 PRINT "███"
110 PRINT
120 PRINT "SOLVE THIS RIDDLE"
130 PRINT
200 PRINTA$:PRINTB$:PRINTC$:PRINTD$:PRINTE$:PRINTF$
205 PRINTG$:PRINTH$:PRINTI$:PRINTJ$
210 PRINT
220 PRINT"███"
230 PRINT
235 INPUT X$
240 IF X$ = K$ THEN GOTO 400
250 PRINT "███THERE IS A HINT - TRY AGAIN"
260 PRINT
270 PRINTL$:PRINTM$:PRINTN$
280 INPUT Y$
290 IF Y$ = K$ THEN GOTO 400
300 GOTO 250
400 PRINT "███YOU ARE RIGHT, THE ANSWER TO THE RIDDLE WAS "K$
410 FOR T=1 TO 2000:NEXT T
420 GOTO 10
1000 DATA FIRST 1/2 OF WHATEVER,MIDDLE 3/5 OF CHASE,FIRST 1/2 OF FIVEFOLD
1020 DATA LAST 1/2 OF BUCKEYES,MIDDLE 3/5 OF HANDY,LAST 5/7OF ARRESTS
1030 DATA LAST 1/3 OF BEACON,MIDDLE 1/9 OF CRABAPPLE,FIRST 5/9 OF WATERFALL
1040 DATA LAST 3/7 OF GRABBED,THE MISSISSIPPI RIVER,MIDDLE 3/5 OF OTHER
1050 DATA FIRST 11/13 OF MISSISSIPPIAN,LAST 5/6 OF DRIVER
1060 DATA FIRST 1/2 OF WHATEVER,LAST 1/2 OF BABYFOOD,MIDDLE 1/3 OF UNIQNE
1070 DATA FIRST 5/7 OF BRAVERY,LAST 3/6 OF GENTLEPEOPLE
1080 DATA MIDDLE 3/7 OF CREATOR,.,.,.
1090 DATA HERO SUB,FIRST 4/7 OF HEROINE,LAST 3/5 OF GOSUB,
```

- Line 20 defines the stop program condition.
- Lines 10, 30, and 40 read all the string DATA for the riddles.
- Line 100 provides a green stripe across the screen top.
- Lines 200 and 205 print the riddle “code.”
- Line 220 provides a green stripe across the screen bottom.
- Line 235 asks you for your input — the answer to the riddle.
- Line 240 defines the correct answer condition.
- Line 250 through 280 gives you a hint if you gave the wrong answer.
- Line 290 compares your input to the correct riddle answer (K\$).
- Line 400 acknowledges that you have solved the riddle correctly.

Suggested Program Changes and “Dressing”

Add color, sound, and more elaborate graphics. If you don’t like the green riddle box we have provided in the program, change it!

This program is persistent about asking for the answer. It won’t stop try-

ing. If you would rather have the answer to the riddle appear after failing to correctly answer (using the hint), here is how:

Change line 300 to:

```
300 PRINT "WRONG AGAIN. THE ANSWER IS" K#
```

and add line 310:

```
310 GOTO 10
```

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 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 you are allowed only 88 characters (or four lines on the screen) in DATA statements for each statement.

PAINTER'S PALETTE



112 THE TOOL KIT SERIES Commodore 64 Edition

Welcome to art class at Commodore school. We think you will enjoy creating your own pictures and graphic designs on the screen using all eight Commodore colors and a brush.

The brush movement is controlled by the keyboard:

A = UP < = LEFT
Z = DOWN > = RIGHT

The brush color is changed using the color keys at the top of the keyboard.

If the brush touches the picture frame, it is returned to the screen center in white. Note that turning the brush to white allows you to erase.

```
5 REM - PRINTER'S PALETTE
10 PRINT "THIS PROGRAM TURNS YOUR SCREEN INTO A COMMODORE PALETTE"
20 PRINT "YOUR BRUSH IS THE BLACK BALL(●) WHICH APPEARS IN THE SCREEN CENTER"
30 PRINT "YOU CONTROL THE BRUSH BY USING THESE KEYS:"
40 PRINT "...A= UP"
50 PRINT "...Z= DOWN"
60 PRINT "...<= LEFT"
70 PRINT "...>= RIGHT"
80 FOR T=1 TO 3000: NEXT T
90 PRINT "USE THE COLOR KEYS TO CHANGE THE BRUSH COLOR"
100 PRINT "WARNING-IF YOU PAINT THE PICTURE FRAME"
110 PRINT "A WHITE BRUSH RETURNS TO THE CENTER OF THE SCREEN"
115 FOR R=1024 TO 2023: POKE R, 81: POKE R+54272, 1: NEXT R
120 GOSUB 500
130 A=1524: B=55796: K=0
140 X=PEEK(197)
150 IF X=10 THEN D=-40
160 IF X=12 THEN D=40
170 IF X=47 THEN D=-1
180 IF X=44 THEN D=1
190 IF X=64 THEN D=0
200 IF X=56 THEN K=0
210 IF X=59 THEN K=1
220 IF X=8 THEN K=2
230 IF X=11 THEN K=3
240 IF X=16 THEN K=4
250 IF X=19 THEN K=5
260 IF X=24 THEN K=6
270 IF X=27 THEN K=7
290 A=A+D
300 B=B+D
305 IF PEEK(A)=160 THEN A=1524: B=55796: K=1
310 POKE A, 81: POKE B, K
320 FOR W=1 TO 50: NEXT W
330 GOTO 140
500 PRINT "!" : POKE 53281, 1
505 FOR P=1024 TO 1063: POKE P, 160: POKE P+54272, 0: NEXT P
510 FOR P=1024 TO 1984 STEP 40: POKE P, 160: POKE P+54272, 0: NEXT P
520 FOR P=1063 TO 2023 STEP 40: POKE P, 160: POKE P+54272, 0: NEXT P
530 FOR P=1984 TO 2023: POKE P, 160: POKE P+54272, 0: NEXT P
590 RETURN
```


- Lines 10 through 110 give you directions for painting.
- Line 130 lets A and B be defined as screen and color code locations (near the center of the screen map). K is set at 0 (color code for black) initially.
- Line 140 checks to see if a key is being pressed.
- Lines 150 through 270 tell the program that if a key is pressed, values

for D (distance) and K (color) are assigned. All the codes for memory box 197 are listed in the appendices.

- Line 290 states that A equals A (1524 to start with) plus D (distance depending upon what key is pressed).
- Line 305 states that if the paint-brush character location touches a PEEK value of 160 (the picture frame), the brush is returned to the screen-map center, in color white (K = 1).
- Line 310 states that the brush is moved and color is changed based on which keys are pressed.
- Line 320 slows the brush movement.
- Subroutine 590 POKEs a solid black picture frame.

Suggested Program Changes and “Dressing”

This program obviously uses graphics and color extensively. Some ideas for changes include:

1. Make the picture frame a different color or use a different screen character. See the POKE screen code table in the appendix. Try code 127, the , for a checkerboard effect.
2. Make the paint brush different by using different characters:

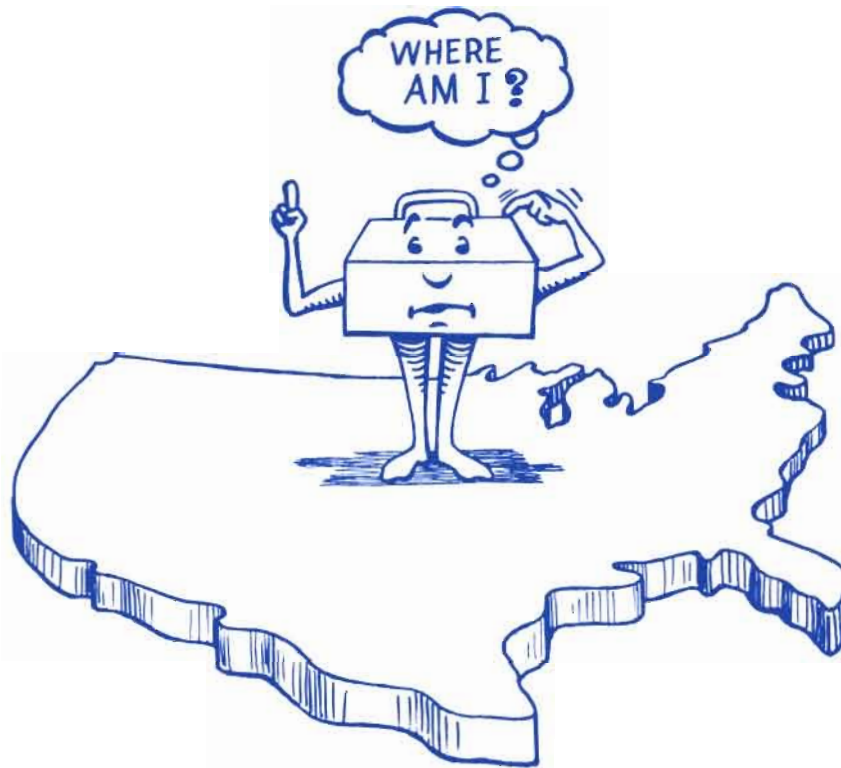
Solid block	160
Diamond	90
Spade	65
3. Slow down or speed up the movement of the brush by changing (or deleting) line 320.

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 the 50 states and are then scored and evaluated at the end of the game. Both color and sound greet you in the scoring subroutines of this game as well as when you enter a correct or incorrect response.

The structure of the overall program is similar to Arithmetic Tester and Multiplication Tables. It uses a powerful BASIC tool — arrays. With the DIM statement, in this program, you are allowed to dimension the data in two ways, one for States and one for Capitals. An important factor in these types of



educational quizzes is that you are able to have the questions chosen randomly from the data.

```

1 REM ---STATES AND CAPITALS---
5 Q=0:N=0
6 GOSUB 6000
10 DIM A$(50),B$(50)
15 RESTORE
20 FOR Z=1 TO 50:READ A$(Z),B$(Z):NEXT Z
30 Z=INT(RND(1)*50)+1
40 IF Q=20 THEN GOTO 200
50 Q=Q+1
60 PRINT "WHAT IS THE CAPITAL OF "A$(Z)
70 PRINT
90 INPUT X$
100 PRINT
110 IF X$=B$(Z) THEN GOSUB 1500
120 IF X$<>B$(Z) THEN GOSUB 2000
130 GOTO 15
200 GOTO 210
210 PRINT N " QUESTIONS RIGHT OUT OF A TOTAL OF "Q
220 J=N/Q
230 IF J>=.9 THEN GOSUB 3000
240 IF J<.9 AND J>=.7 THEN GOSUB 4000
250 IF J<.7 THEN GOSUB 5000
500 DATA ALABAMA,MONTGOMERY,NEW YORK,ALBANY,NEVADA,CARSON CITY,OHIO,COLUMBUS
550 DATA IOWA,DES MOINES,SOUTH CAROLINA,COLUMBIA,OREGON,SALEM,TEXAS,AUSTIN
600 DATA KENTUCKY,FRANKFORT,NEW JERSEY,TRENTON,MASSACHUSETTS,BOSTON
650 DATA ARIZONA,PHOENIX,TENNESSEE,NASHVILLE,WYOMING,CHEYENNE
    
```


116 THE TOOL KIT SERIES Commodore 64 Edition

- Line 15 restores the DATA after each question is asked.
- Line 20 reads the DATA in the array, as expressed in the subscript (Z), from 1 to 50.
- Line 30 sets Z as a random integer from 1 to 50.
- Line 40 tells the program to go to the scoring/ending section of the game when 20 questions have been asked.
- Line 110 compares the input to the answer and sends the program to a correct answer subroutine if they are equal.
- Line 120 sends the program to the wrong answer subroutine if the input is not equal to the correct answer.
- Lines 500 through 1250 are the DATA statements for the array. The format is just the same for READ-DATA statements.
- Subroutine 1500 plays a nice musical scale for a correct answer.
- Subroutine 2000 plays a low descending scale for a wrong answer.
- Subroutine 3000 rewards you with a color routine for answering correctly 90% or more of the questions.
- Subroutine 4000 changes the screen color and an “OK” message appears for this satisfactory level of competence — 70 to 90% correct.
- Subroutine 5000 turns the screen black with a message suggesting a need for more study of states and capitals.

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. Use more data or less data.
2. Change the scoring levels.
3. Add a title opener to the program. A map of the United States would be appropriate. *Warning:* This will take some serious POKEing about to accomplish.

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

Countries and Capitals
Continents and Countries
Inventors and Objects
Historical Events and Dates
Mythological Characters

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 will 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 using dice, cards, paper and pencil, or game boards. In this chapter, we will look at four such games and change them so that they can be played with your Commodore 64. Using these games as models, you should be able to create or adapt your own traditional games. Let's begin with a very easy one.

GUESSING 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 might want to try. Regardless of what is guessed, the program will be about the same. The computer picks a number, a color, or whatever by using the Random Number function. You try to guess the computer's pick using the Input statement. The computer checks to see if you guessed correctly or incorrectly and lets you know the outcome.



```

5 REM*** GUESSING GAME ***
10 TRY=0:C=53281
20 PRINT"?"
30 X=INT(RND(1)*8)+1
40 PRINT"00000PICK A COLOR USING THE COMMODORE COLOR KEYS"
50 PRINT:PRINT:INPUT Y$
60 TRY=TRY+1
70 PRINT"THAT WAS TRY NUMBER ";TRY
80 IF X=VAL(Y$) THEN GOSUB 100
90 IF X<>VAL(Y$) THEN GOSUB 200
100 PRINT"YOU GUESSED CORRECTLY. NOW WATCH THE COLOR"
105 FOR T=1 TO 1000:NEXT T
110 IF Y$="1" THEN POKEC,0
115 IF Y$="2" THEN POKEC,1
120 IF Y$="3" THEN POKEC,2
125 IF Y$="4" THEN POKEC,3
130 IF Y$="5" THEN POKEC,4
135 IF Y$="6" THEN POKEC,5
140 IF Y$="7" THEN POKEC,6
145 IF Y$="8" THEN POKEC,7
150 FOR T=1 TO 1000:NEXT T
155 POKEC,6
160 PRINT "IT TOOK YOU";TRY;" TRIES"
    
```

```

170 FOR T=1TO1500:NEXT T
180 GOTO 10
190 RETURN
200 PRINT "SORRY, TRY AGAIN"
210 FOR T=1TO1000:NEXT T
220 GOTO 40
230 RETURN

```

Although the game is a simple one, it has the components for some very playable guessing games. The computer selects a random number from 1 to 8 (line 30). You select one of the color keys (actually, keys 1 to 8). If the value of your input, VAL(Y\$), is equal to the random number generated, then the program moves to the subroutine at line 100. If the number was not equal to the "value of Y\$", then the program moves to the subroutine at line 200.

You can change the game to play other guessing games, such as Sounds or Numbers. If you want to guess a number from 1 to 100, you should change the following lines:

```

30 X = INT (RND (1)*100) + 1
50 INPUT Y
80 IF X=Y THEN GOSUB
90 IF X>Y THEN GOSUB
95 IF X<Y THEN GOSUB

```

We leave the rest to you. Try this number-guessing game on your own.

TIC-TAC-TOE

Let's try another favorite game, Tic-Tac-Toe. Instead of paper and pencil, you play on the display screen. Enter the program and we will show you how to play. This game requires two players.

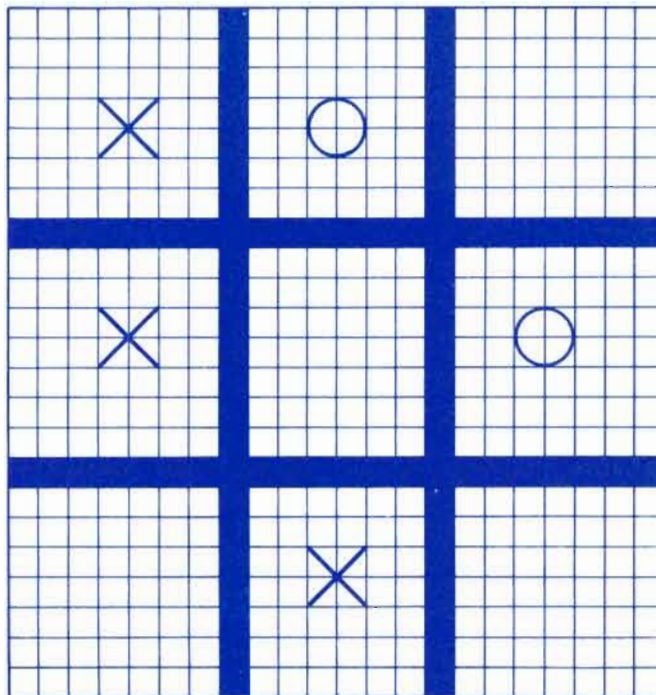
```

5 REM *** TIC TAC TOE ***
10 PRINT":XXXXXXXXXXPLAY TIC TAC TOE ON THE KEYBOARD"
20 PRINT":XXXXXXXXXSTOP THE GAME - PRESS THE SPACE BAR"
30 FOR Z=1024TO2023:POKEZ,214:POKEZ+54272,5:NEXT
60 PRINT":]"
70 S=1024:C=55296
80 FOR G=1264 TO 1284:POKEG,160:POKEG+54272,0:NEXT
90 FOR G=1544 TO 1564:POKEG,160:POKEG+54272,0:NEXT
100 FOR G=1030 TO 1790 STEP40:POKEG,160:POKEG+54272,0:NEXT
110 FOR G=1037 TO 1797 STEP40:POKEG,160:POKEG+54272,0:NEXT
120 POKES,1:POKES+1,61:POKES+2,87:POKEC,0:POKEC+1,0:POKEC+2,0
130 POKES+40,2:POKES+41,61:POKES+42,86:POKEC+40,0:POKEC+41,0:POKEC+42,0
140 POKES+7,3:POKES+8,61:POKES+9,87:POKEC+7,0:POKEC+8,0:POKEC+9,0
150 POKES+47,4:POKES+48,61:POKES+49,86:POKEC+47,0:POKEC+48,0:POKEC+49,0
160 POKES+14,5:POKES+15,61:POKES+16,87:POKEC+14,0:POKEC+15,0:POKEC+16,0
170 POKES+54,6:POKES+55,61:POKES+56,86:POKEC+54,0:POKEC+55,0:POKEC+56,0
180 POKES+280,7:POKES+281,61:POKES+282,87:POKEC+280,0:POKEC+281,0:POKEC+282,0
190 POKES+320,8:POKES+321,61:POKES+322,86:POKEC+320,0:POKEC+321,0:POKEC+322,0
200 POKES+287,9:POKES+288,61:POKES+289,87:POKEC+287,0:POKEC+288,0:POKEC+289,0
210 POKES+327,10:POKES+328,61:POKES+329,86:POKEC+327,0:POKEC+328,0:POKEC+329,0
220 POKES+296,11:POKES+297,61:POKES+298,87:POKEC+296,0:POKEC+297,0:POKEC+298,0
230 POKES+336,12:POKES+337,61:POKES+338,86:POKEC+336,0:POKEC+337,0:POKEC+338,0
240 POKES+560,13:POKES+561,61:POKES+562,87:POKEC+560,0:POKEC+561,0:POKEC+562,0

```

122 THE TOOL KIT SERIES Commodore 64 Edition

```
250 POKES+600,14:POKES+601,61:POKES+602,86:POKEC+600,0:POKEC+601,0:POKEC+602,0
260 POKES+567,15:POKES+568,61:POKES+569,87:POKEC+567,0:POKEC+568,0:POKEC+569,0
270 POKES+607,16:POKES+608,61:POKES+609,86:POKEC+607,0:POKEC+608,0:POKEC+609,0
280 POKES+574,17:POKES+575,61:POKES+576,87:POKEC+574,0:POKEC+575,0:POKEC+576,0
290 POKES+614,18:POKES+615,61:POKES+616,86:POKEC+614,0:POKEC+615,0:POKEC+616,0
300 R$=CHR$(65):B$=CHR$(66):C$=CHR$(67):D$=CHR$(68):E$=CHR$(69)
310 F$=CHR$(70):G$=CHR$(71):H$=CHR$(72):I$=CHR$(73):J$=CHR$(74):K$=CHR$(75)
320 L$=CHR$(76):M$=CHR$(77):N$=CHR$(78):O$=CHR$(79):P$=CHR$(80)
330 Q$=CHR$(81):R$=CHR$(82):S$=CHR$(82)
350 GET X$:IFX$=""THEN GOTO350
380 IF X$=R$ THEN POKES+123,15:POKEC+123,1
390 IF X$=B$ THEN POKES+123,86:POKEC+123,1
400 IF X$=C$ THEN POKES+130,15:POKEC+130,1
410 IFX$=D$ THEN POKES+130,86:POKEC+130,1
420 IFX$=E$ THEN POKES+137,15:POKEC+137,1
430 IFX$=F$ THEN POKES+137,86:POKEC+137,1
440 IFX$=G$ THEN POKES+403,15:POKEC+403,1
450 IFX$=H$ THEN POKES+403,86:POKEC+403,1
460 IFX$=I$ THEN POKES+410,15:POKEC+410,1
470 IFX$=J$ THEN POKES+410,86:POKEC+410,1
480 IFX$=K$ THEN POKES+417,15:POKEC+417,1
490 IFX$=L$ THEN POKES+417,86:POKEC+417,1
500 IFX$=M$ THEN POKES+683,15:POKEC+683,1
510 IFX$=N$ THEN POKES+683,86:POKEC+683,1
520 IFX$=O$ THEN POKES+690,15:POKEC+690,1
530 IFX$=P$ THEN POKES+690,86:POKEC+690,1
540 IFX$=Q$ THEN POKES+697,15:POKEC+697,1
550 IFX$=R$ THEN POKES+697,86:POKEC+697,1
560 IFX$=S$ THEN GOTO 10
570 GOTO 350
```

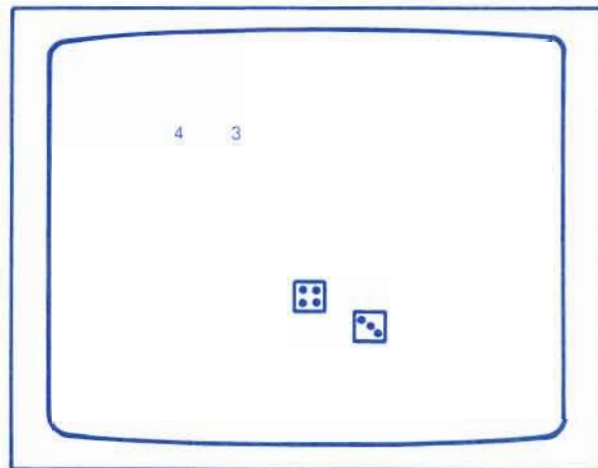


- Lines 10 through 30 set up the instructions and the introductory frame.
- Lines 80 through 110 set up the Tic-Tac-Toe “grid.”
- Lines 120 through 290 tell you what letters represent an “X” and an “O” for each of the boxes.
- Lines 300 through 330 define the input you can make on the keyboard.
- Line 350 asks you to input from the keyboard.
- Lines 410 through 550 plot either an “X” or an “O” in one of the nine Tic-Tac-Toe blocks.
- Line 560 tells the program to start over with a clean slate if the space bar is pressed (CHR\$32).

The game can be simplified by reducing the grid size, including the “X” and “O” instructions in a PRINT command but, then, the actual ease of play would be less.

TED AND DAVE’S CASINO

How about a little gambling for you older folks? At your command, the Commodore 64 will roll the dice. You begin play with \$10.00. If you roll a 7 or an 11, you win \$2.00. If you roll less than a 4 or if you roll a 12, you lose \$2.00. If any other number comes up, the house takes \$1.00 for the roll. Enter the program, play with it, and adapt it to suit yourself.



124 THE TOOL KIT SERIES Commodore 64 Edition

```
10 REM *** TED AND DAVE'S CASINO ***
20 GOSUB 5000
30 M=M+10
40 S=1024:C=55296
80 PRINT "C"
100 INPUT "PRESS RETURN TO ROLL THE DICE!";X$
110 GOSUB 6000
200 LD=INT(RND(1)*6)+1
210 IFLD=1 THEN GOSUB 500
220 IFLD=2 THEN GOSUB 550
230 IFLD=3 THEN GOSUB 600
240 IFLD=4 THEN GOSUB 650
250 IFLD=5 THEN GOSUB 700
260 IFLD=6 THEN GOSUB 750
300 RD=INT(RND(1)*6)+1
310 IFRD=1 THEN GOSUB 800
320 IFRD=2 THEN GOSUB 850
330 IFRD=3 THEN GOSUB 900
340 IFRD=4 THEN GOSUB 950
350 IFRD=5 THEN GOSUB 1000
360 IFRD=6 THEN GOSUB 1050
370 PRINT LD,RD
380 IFLD+RD=7 OR LD+RD=11 THEN M=M+2
410 IFLD+RD<4 THEN M=M-2
420 IFLD+RD=4ORLD+RD=5ORLD+RD=6ORLD+RD=8ORLD+RD=9ORLD+RD=10 THEN M=M-1
430 PRINT:PRINT"YOU NOW HAVE "M" DOLLARS"
440 IFM<1 THEN GOSUB 3000
450 IFM>20 THEN GOSUB 4000
460 FOR T=1TO2000:NEXT
470 GOTO 80
500 POKES+661,81:POKEC+661,0
540 RETURN
550 POKES+620,81:POKEC+620,0:POKES+702,81:POKEC+702,0
590 RETURN
600 GOSUB 500:GOSUB 550
640 RETURN
650 GOSUB 550
660 POKES+622,81:POKEC+622,0:POKES+700,81:POKEC+700,0
690 RETURN
700 GOSUB 500:GOSUB 650
740 RETURN
750 GOSUB 650
760 POKES+621,81:POKEC+621,0:POKES+701,81:POKEC+701,0
790 RETURN
800 POKES+748,81:POKEC+748,0
840 RETURN
850 POKES+707,81:POKEC+707,0:POKES+789,81:POKEC+789,0
890 RETURN
900 GOSUB 800:GOSUB 850
940 RETURN
950 GOSUB 850
960 POKES+709,81:POKEC+709,0:POKES+787,81:POKEC+787,0
990 RETURN
1000 GOSUB 800:GOSUB 950
1040 RETURN
1050 GOSUB 950
1060 POKES+708,81:POKEC+708,0:POKES+788,81:POKEC+788,0
1090 RETURN
3000 REM ** LOSING SUBROUTINE
3010 POKE54296,15
3020 POKE54277,9
3030 FOR T=1TO50:NEXT
3040 READ A
3050 READ B
3060 IF B=-1 THEN GOTO 3150
3070 POKE54273,A:POKE54272,B
3080 POKE54276,17
3090 FOR T=1TO50:NEXT
3100 POKE54276,16
3110 GOTO 3020
3120 DATA 68,149,64,188,57,172,51,97,45,198,43,52,38,126,34,75
```

```

3130 DATA-1,-1
3150 PRINT"YOU NOW YOU HAVE LOST ALL YOUR MONEY. CASINO IS CLOSED"
3160 END
3190 END
3199 RETURN
4000 REM ** WINNING SUBROUTINE
4010 PRINT"YOU YOU HAVE WON TOO MUCH MONEY. TED AND DAVE'S CASINO IS CLOSED."
4020 FOR T=1T03000:NEXT
4190 END
4199 RETURN
5000 REM ** OPENING SUBROUTINE
5010 PRINT"YOU WELCOME TO TED AND DAVE'S CASINO"
5020 PRINT"YOU YOU HAVE 10 DOLLARS TO PLAY WITH - TRY YOUR LUCK"
5030 PRINT"ROLL 7 OR 11 AND YOU WIN $2"
5040 PRINT"ROLL LESS THAN 4 OR ROLL A 12 AND YOU LOSE $2"
5050 PRINT"ANY OTHER ROLL AND YOU LOSE $1"
5060 FOR T=1T02500:NEXT
5070 FOR Z=1024 TO 2023:POKEZ,193:POKEZ+54272,5:NEXT Z
5199 RETURN
6000 REM *** DICE OUTLINE
6010 POKES+579,85:POKEC+579,0:POKES+580,64:POKEC+580,0
6020 POKES+581,64:POKEC+581,0:POKES+582,64:POKEC+582,0
6030 POKES+583,73:POKEC+583,0:POKES+619,66:POKEC+619,0
6040 POKES+623,66:POKEC+623,0:POKES+659,66:POKEC+659,0
6050 POKES+663,66:POKEC+663,0:POKES+699,66:POKEC+699,0
6060 POKES+703,66:POKEC+703,0:POKES+739,74:POKEC+739,0
6070 POKES+740,64:POKEC+740,0:POKES+741,64:POKEC+741,0
6080 POKES+742,64:POKEC+742,0:POKES+743,75:POKEC+743,0
6090 POKES+666,85:POKEC+666,0:POKES+667,64:POKEC+667,0
6100 POKES+668,64:POKEC+668,0:POKES+669,64:POKEC+669,0
6110 POKES+670,73:POKEC+670,0:POKES+706,66:POKEC+706,0
6120 POKES+710,66:POKEC+710,0:POKES+746,66:POKEC+746,0
6130 POKES+750,66:POKEC+750,0:POKES+786,66:POKEC+786,0
6140 POKES+790,66:POKEC+790,0:POKES+826,74:POKEC+826,0
6150 POKES+827,64:POKEC+827,0:POKES+828,64:POKEC+828,0
6160 POKES+829,64:POKEC+829,0:POKES+830,75:POKEC+830,0
6999 RETURN

```

- Line 30 gives you ten dollars to bet with.
- Lines 200 and 300 “Roll the dice.” They define the left die (LD) and right die (RD) as integers from 1 to 6.
- Lines 210 through 260 and 310 through 360 send the program to the appropriate subroutines to plot the dice faces.
- Lines 380 through 420 define the scoring conditions.
- Subroutines 500 through 1050 POKE the faces of each die on the screen depending upon the value of LD and RD.
- Subroutines 3000 plays a descending scale for a losing player and ends the game.
- Subroutine 4000 ends the game when the player has won twenty dollars or more.
- Subroutine 5000 opens the game with instructions and some graphics.
- Subroutine 6000 plots the outline of the dice.

Some of the ways to adapt the game are listed below:

1. Include sound each time the dice are rolled.
2. Change the win/lose conditions for the game, such as

IF LD=RD THEN M=M + 2

126 THE TOOL KIT SERIES Commodore 64 Edition

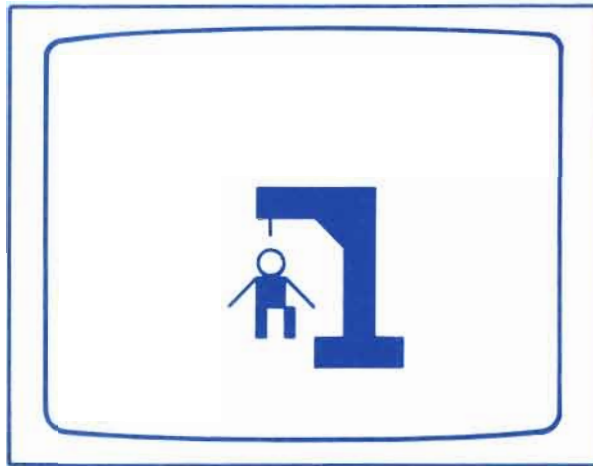
which gives the player two dollars for rolling doubles.

3. Add more color and sound for signaling the end of the game.

When you are tired of losing your shirt gambling, try the next game, a friendly little game of HANGMAN.

HANGMAN

The Commodore plays you in HANGMAN. Guess the secret word in the least number of tries and you avoid "hanging." When you guess the word correctly, a person appears with a "THANKS!" message. If you don't guess the word correctly, he is hung.



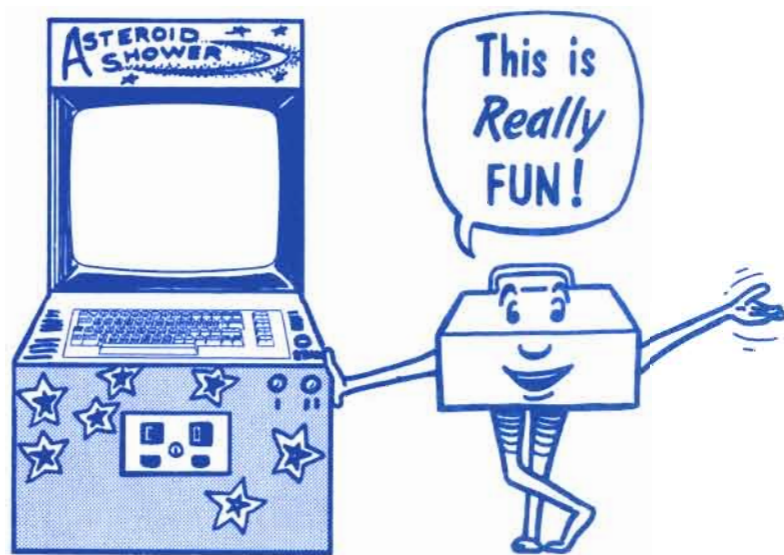
```
10 REM *** HANGMAN
20 PRINT "XXXXXXXXXWELCOME"
30 PRINT "XXXXXXXXX"
40 PRINT "XXXXXXXXXHANGMAN"
50 GOSUB1000:GOSUB1200
60 FORT=1TO3000:NEXT
70 X=INT(RND(1)*10)+1
80 FORA=1TOX
90 READ A$
100 NEXT A
110 N=LEN(A$)
120 FOR A=1TON
130 L(A)=ASC(MID$(A$,A,1))
140 O(A)=L(A)
150 NEXT A
160 T=N+4
170 PRINT"XGUESS THIS WORD IN "T" TRIES"
180 FOR S=1TOT
190 GOSUB 450
200 IFC=N THEN 340
210 PRINT T+1-"S" TRIES TO GO!"
220 INPUT X$
230 X=ASC(X$)
```


value in the array O(A) is set at 0. This allows the correct letter to be printed amid the dashes. S is reduced by one, meaning that the number of tries is *not* reduced when a correct letter is guessed.

- Lines 290 through 320 set up the losing condition for HANGMAN.
- Lines 340 through 380 set up the winning condition for HANGMAN.
- Line 450 sets the number of correct letters at 0 (zero).
- Line 470 checks to see if any letters in A\$ have been selected. If each character in the array L(Z) equals 0(Z), then a star is printed.
- Line 480 checks to see if any letters have been selected in A\$. If a correct letter has been selected, L(Z) is no longer equal to 0(Z) and the program prints the correct letter in the right space. The number that is correct increases by one for each correct letter.
- Lines 600 through 690 set up the losing condition for HANGMAN.
- Subroutine 1000 draws the person on the screen.
- Subroutine 1200 draws the gallows and POKEs a losing sound routine as well.

We hope you have enjoyed the games in this chapter. Review how the answer conditions were treated differently in the CASINO and HANGMAN games. Try some of your own favorites and use our games as models. We use some of the logic contained here for the arcade-type games in our next chapter. Let's play some ARCADE GAMES now.





Arcade Games

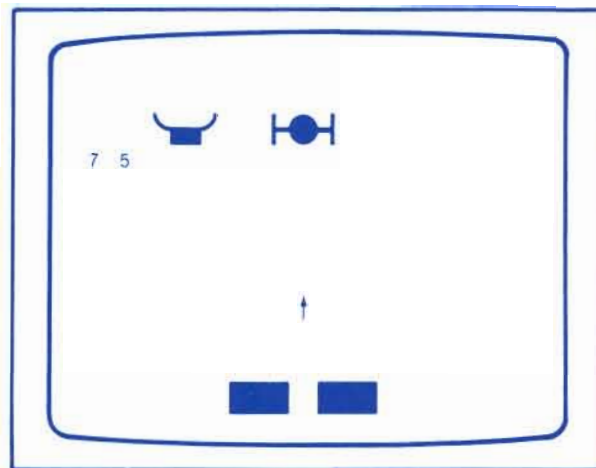
10

130 THE TOOL KIT SERIES Commodore 64 Edition

Creating arcade games can be a lot of fun, but can be extremely frustrating if you are not careful. Begin with simple games and improve them. We wrote the games in this chapter by planning the simple game logic (car driving up screen, creature moving through maze, etc.) and then adding color, sound, scoring, and other enhancements. We hope you enjoy these arcade games.

SAUCER BLASTER

Two spacecraft appear on the screen; the left being the escort ship and the right being the enemy saucer. You shoot your missile from the green "silo" by pressing function key 1 (F1). If you hit the enemy saucer, you earn 25 points. If you accidentally hit the friendly escort ship, you lose 15 points.



```
10 REM *** SAUCER.BLASTER ***
20 GOSUB 1000
30 PRINT "J"
50 Y=0: X=0: ZX=1: C=2004: R=0
60 POKE2002,160: POKE2003,160: POKE2005,160: POKE2006,160
70 POKE56274,5: POKE56275,5: POKE56277,5: POKE56278,5
75 S=0
80 PRINT"0000"R
90 K=PEEK(197)
100 IFK=4 AND S=0 THEN S=1
110 IFSC>0 THEN GOSUB 500
120 POKE1193+X+40*Y,32: POKE1194+X+40*Y,32: POKE1194+X+40*Y,32
125 POKE55465+X+40*Y,0: POKE55466+X+40*Y,0: POKE55467+X+40*Y,0
130 POKE1203+X+40*Y,32: POKE1204+X+40*Y,32: POKE1205+X+40*Y,32
135 POKE55475+X+40*Y,0: POKE55476+X+40*Y,0: POKE55477+X+40*Y,0
150 X=X+ZX
160 IFX=-9 OR X=18 THEN ZX=-ZX
180 POKE1193+X+40*Y,74: POKE1194+X+40*Y,98: POKE1195+X+40*Y,75
185 POKE55465+X+40*Y,7: POKE55466+X+40*Y,7: POKE55467+X+40*Y,7
190 POKE1203+X+40*Y,107: POKE1204+X+40*Y,81: POKE1205+X+40*Y,115
```


132 THE TOOL KIT SERIES Commodore 64 Edition

- Subroutine 900 subtracts 15 points from the score and plays a “punishment” sound routine for hitting the escort ship.
- Subroutine 1000 opens the game with some color and directions.

Saucer Blaster has all the elements of a good arcade game — graphics, animation, color, sound, and a scoring mechanism. Change some of the lines in this program to improve it. One of the best ways to test your understanding of this game’s logic is to change it to a horizontal game. Have the ships move up and down the screen with the missile firing across. Or, change the missile into bombs being dropped down the screen.

POTHOLE DERBY

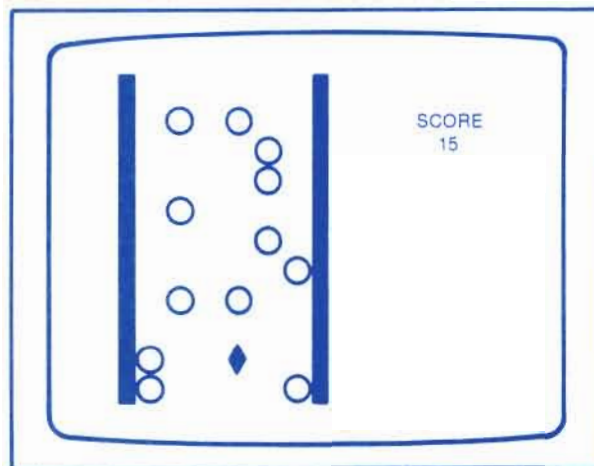
The object of this game is to drive the car (◆) up the road avoiding the potholes and the side of the road. You can control the car by use of the function keys:

- Press F1 The car moves left.
- Press F3 The car moves right.

You are given three laps on this road. The score is computed by:

- + 20 For reaching the top of the road.
- 5 For hitting a pothole.
- 10 For running off the side of the road.

In scoring, 60 is a perfect score.



```
5 REM *** POTHOLE DERBY ***
10 S=0:GOSUB000
15 FOR L=1TO3
20 PRINT"␣"
30 GOSUB700:GOSUB500:GOSUB600
```

```

40 A=0:F1=4:F3=5:F0=64
60 FORB=0TO-23STEP-1
70 K=PEEK(197)
80 IF K=F1 THEN A=A-1
90 IF K=F3 THEN A=A+1
100 IF PEEK(2004+A+40*B)=118 THEN GOSUB 400
110 IF PEEK(2004+A+40*B)=117 THEN GOSUB 400
120 IF PEEK(2004+A+40*B)=87 THEN GOSUB 300
140 IF B=-23 THEN S=S+20
180 POKE2004+A+40*B,90
190 POKE56276+A+40*B,2
200 FOR T=1TO100:NEXT
210 POKE2004+A+40*B,32
260 NEXT B
270 NEXT L
290 PRINT"#####:YOUR FINAL"
292 PRINT"#####SCORE IS"
294 PRINT"#####S"
299 END:POKE54296,0
300 REM**HIT POT HOLE
305 POKE54296,15:POKE54273,34:POKE54272,75
310 POKE54277,9
315 POKE54276,129
320 FORT=1TO80:NEXT
330 POKE54276,128
380 S=S-5
399 RETURN
400 REM**HIT SIDE OF ROAD
410 POKE54296,15
420 POKE54276,129
430 POKE54273,54:POKE54272,111
440 POKE54277,9:POKE54278,24
450 FORT=1TO75:NEXT
460 POKE54276,128
480 S=S-10
499 RETURN
500 REM**ROAD BORDER
510 FORZ=1039TO1999STEP40:POKEZ,118:POKEZ+54272,6:POKEZ+8,117:POKEZ+54280,6
520 NEXT Z
550 RETURN
600 REM**POT HOLES
610 FOR V=1TO100
620 J=INT(RND(0)*40)
630 K=INT(RND(0)*22)
640 G=1024+J+40*K
650 IF J>15ANDJ<23THENPOKEG,87:POKEG+54272,4
660 NEXT V
699 RETURN
700 PRINT"#####SCORE"
710 PRINT"#####S"
720 RETURN
800 REM**OPENING FRAME
810 POKE53281,1
820 PRINT"#####WELCOME TO##"
830 PRINT"#####POT HOLE DERBY##"
840 FOR T=1TO1000:NEXT
850 PRINT"#####PRESS F1 TO MOVE LEFT"
860 PRINT"#####PRESS F3 TO MOVE RIGHT"
870 FORU=1024TO2023:POKEU,214:POKEU+54272,4:NEXT
899 RETURN

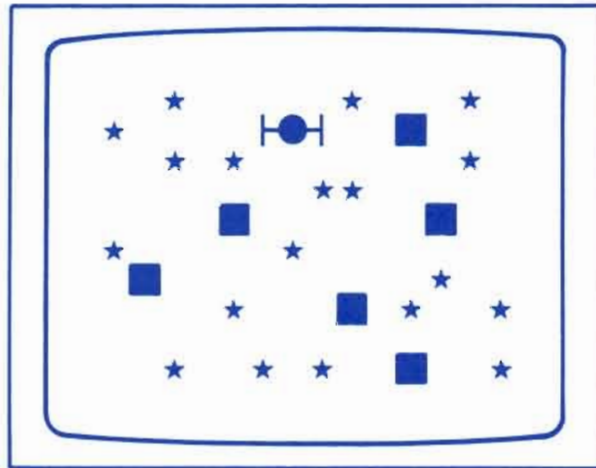
```

You can increase the difficulty of the game easily by changing the number of potholes in line 610. Change line 15 if you want more laps or fewer laps.

- Line 15 sets up the 3 laps for the pothole derby.
- Line 60 starts movement of the car up the screen.
- Line 70 asks if a key is being pressed.

134 THE TOOL KIT SERIES Commodore 64 Edition

- Line 80 checks to see if function key 1 (F1) is pressed, and then the car moves to the left.
- Line 90 checks to see if function key 3 (F3) is pressed, and then the car moves to the right.
- Line 100 tells the program if the car has hit the left-hand side of the road.
- Line 110 tells the program if the car has hit the right-hand side of the road.
- Line 120 tells the program if the car has hit a pothole.
- Line 140 checks to see if the car has reached the screen top.
- Lines 180 and 190 POKE the car into position.
- Line 210 erases the car.
- Lines 290 through 299 end the game with a final score.
- Subroutine 300 subtracts 5 from the score and plays a sound.
- Subroutine 400 subtracts 10 from the score and plays a sound for hitting the roadside.
- Subroutine 600 plots potholes in the road.
- Subroutine 700 keeps track of the score as the game progresses.
- Subroutine 800 welcomes you to POTHOLE DERBY with directions and some graphics.



ASTEROID SHOWER

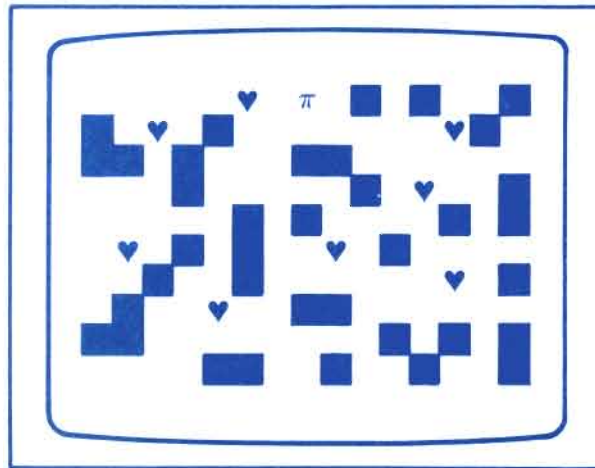
Using the same program logic as in Pothole Derby, we can create an entirely different arcade game, Asteroid Shower. You are a space ship cleaning space of asteroids. Each time you remove an asteroid, you earn 10 points.

You only have the "time warps" to fear. If you accidentally touch them, you lose 20 points. You have four turns to clean the skies of all the asteroids.

136 THE TOOL KIT SERIES Commodore 64 Edition

- Line 70 gives the player 4 turns.
- Line 90 directs movement down the screen.
- Lines 120 through 150 move the space ship, depending upon what function key is pressed.
- Lines 160 through 180 look to see if an asteroid is hit by the space ship.
- Lines 190 through 210 look to see if a warp is hit by the space ship.
- Line 220 POKES the space ship.
- Line 240 erases the space ship.
- Line 250 moves the ship down Step 1-Y.
- Line 260 sends the program to the next turn (4 turns).
- Subroutine 500 POKEs 90 asteroids on the screen.
- Subroutine 600 POKEs 10 time warps on the screen.
- Subroutine 800 adds 10 to the score for taking an asteroid.
- Subroutine 900 subtracts 20 points from the score for hitting a time warp.

You can easily add new ideas to Asteroid Shower. POKE a space station on the screen. If the ship lands on the space station for refueling, additional points are earned. Hide invisible black holes that take points away from the score. Use your imagination and have fun.



RAT TRAP

The green rat is trapped in a maze. The object of the game is to score points by capturing the prizes (hearts) while not hitting the maze walls. You earn 35 points for capturing a prize; 10 points are lost for hitting the wall. The rat has two chances to move through the maze. You control the rat with the following keys:

F1 To move up.
 F2 To move down.
 < To move left.
 > To move right.

```

5 REM *** RAT TRAP ***
10 PRINT "R":POKE53281,1
20 GOSUB1000:S=0
30 PRINT "R"
40 GOSUB 500:GOSUB 600
50 FOR A=1TO2
60 X=0
65 FOR Y=0TO22STEP1
70 K=PEEK(197)
80 IFK=47 THENX=X-1
90 IFK=44 THENX=X+1
100 IFK=4 THENY=Y-2
110 IFK=5 THENY=Y+1
120 IF PEEK(1034+X+40*Y)=160 THEN GOSUB 700
130 IF PEEK(1034+X+40*Y)=83 THEN GOSUB 800
160 POKE1034+X+40*Y,222:POKE1034+54272+X+40*Y,5
170 FORT=1TO500:NEXT
180 POKE1034+X+40*Y,32
190 NEXT Y
210 NEXT A
220 PRINT "R:#####FINAL SCORE:##"
230 PRINT "#####S"
250 END
500 REM**MAZE
510 FOR W=1TO120
520 V=1104+INT(RND(1)*820)
530 POKEV,160:POKEV+54272,0
540 NEXT W
599 RETURN
600 REM**PRIZES
610 FORP=1TO20
620 V=1104+INT(RND(1)*820)
630 POKEV,83:POKEV+54272,7
640 NEXT P
699 RETURN
700 REM**HIT MAZE WALL
710 POKE54277,9:POKE54278,40:POKE54296,15:POKE54273,25:POKE54272,0
720 POKE54276,129
730 FORT=1TO500:NEXT
740 POKE54276,128
780 S=S-10
790 PRINT "R:R"
799 RETURN
800 REM**CAPTURE TREASURE
810 POKE54277,9:POKE54278,40:POKE54296,15:POKE54273,150:POKE54272,0
820 POKE54276,33
830 FORT=1TO500:NEXT
840 POKE54276,32
880 S=S+35
890 PRINT "R:R"
899 RETURN
1000 REM**OPENING FRAME
1010 GOSUB 500:GOSUB 600
1020 PRINT "#####RAT TRAP"
1030 FOR T=1TO1500:NEXT
1040 PRINT "#####KEY F1 TO MOVE UP"
1050 PRINT "#####KEY F2 TO MOVE DOWN"
1060 PRINT "#####KEY < TO MOVE LEFT"
1070 PRINT "#####KEY > TO MOVE RIGHT"
1080 FOR T=1TO2500:NEXT
1090 RETURN

```

138 THE TOOL KIT SERIES Commodore 64 Edition

- Line 50 gives the rat two trips through the maze.
- Line 65 gives a downward movement to the game.
- Lines 80 through 110 ask if a movement-control key is being pressed.
- Line 120 asks if the maze wall is hit by the rat.
- Line 130 asks if the prize is captured.
- Line 160 POKEs the rat in its starting position.
- Line 180 erases the rat.
- Line 220 through 250 end the game with a final score.
- Subroutine 500 POKEs the maze walls.
- Subroutine 600 POKEs the yellow prizes on the screen.
- Subroutine 700 subtracts 10 points from the score for hitting the maze wall.
- Subroutine 800 adds 35 points to the score for capturing a prize.

You can customize this game further if you like. Hide a cat in the maze to chase the rat. Increase the difficulty of the game by adding more maze walls.

After working with the programs in this chapter, we hope you have learned how to successfully put all the Commodore 64 tools together to make some enjoyable games. Use our subroutines and programs as examples to develop your knowledge of BASIC and to construct your own educational and recreational programs.



Appendices

Musical Notes



Table A-1. Musical Notes in the Fourth and Fifth Octaves






































NOTE	FIFTH OCTAVE		FOURTH OCTAVE	
	HIGH FREQ	LOW FREQ	HIGH FREQ	LOW FREQ
C	34	75	16	195
C#	36	85	17	195
D	38	126	18	209
D#	40	200	19	239
E	43	52	21	31
F	45	198	22	96
F#	48	127	23	181
G	51	97	25	30
G#	54	111	26	156
A	57	172	28	49
A#	61	126	29	223
B	64	188	31	165

Screen Values for POKE

B

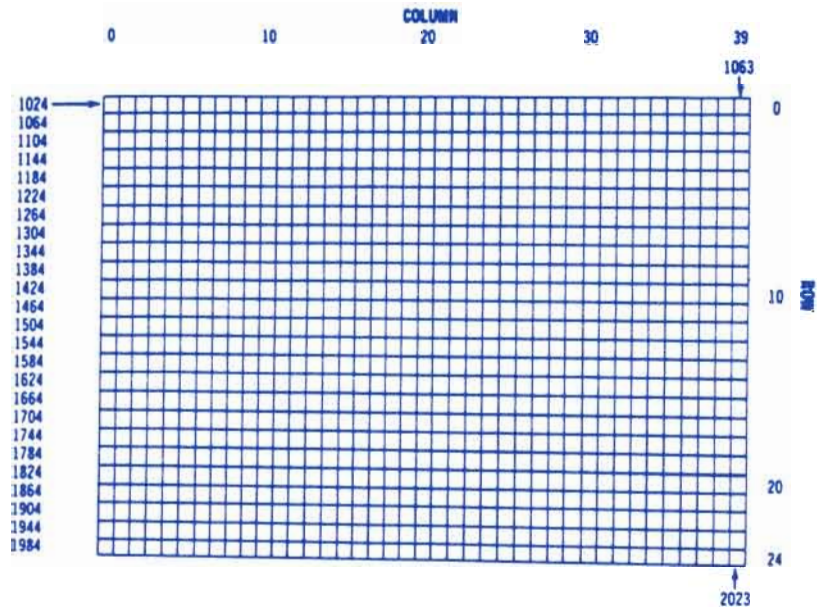
SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE
@		0	←		31	>		62
A	a	1	SPACE		32	?		63
B	b	2	!		33			64
C	c	3	"		34		A	65
D	d	4	#		35		B	66
E	e	5	\$		36		C	67
F	f	6	%		37		D	68
G	g	7	&		38		E	69
H	h	8	,		39		F	70
I	i	9	(40		G	71
J	j	10)		41		H	72
K	k	11	.		42		I	73
L	l	12	+		43		J	74
M	m	13	,		44		K	75
N	n	14	-		45		L	76
O	o	15	.		46		M	77
P	p	16	/		47		N	78
Q	q	17	0		48		O	79
R	r	18	1		49		P	80
S	s	19	2		50		Q	81
T	t	20	3		51		R	82
U	u	21	4		52		S	83
V	v	22	5		53		T	84
W	w	23	6		54		U	85
X	x	24	7		55		V	86
Y	y	25	8		56		W	87
Z	z	26	9		57		X	88
[27	:		58		Y	89
£		28	;		59		Z	90
]		29	<		60			91
↑		30	=		61			92

142 THE TOOL KIT SERIES Commodore 64 Edition

SET 1	SET 2	POKE	SET 1	SET 2	POKE	SET 1	SET 2	POKE
		93			105			117
		94			106			118
		95			107			119
SPACE		96			108			120
		97			109			121
		98			110		<input checked="" type="checkbox"/>	122
		99			111			123
		100			112			124
		101			113			125
		102			114			126
		103			115			127
		104			116			

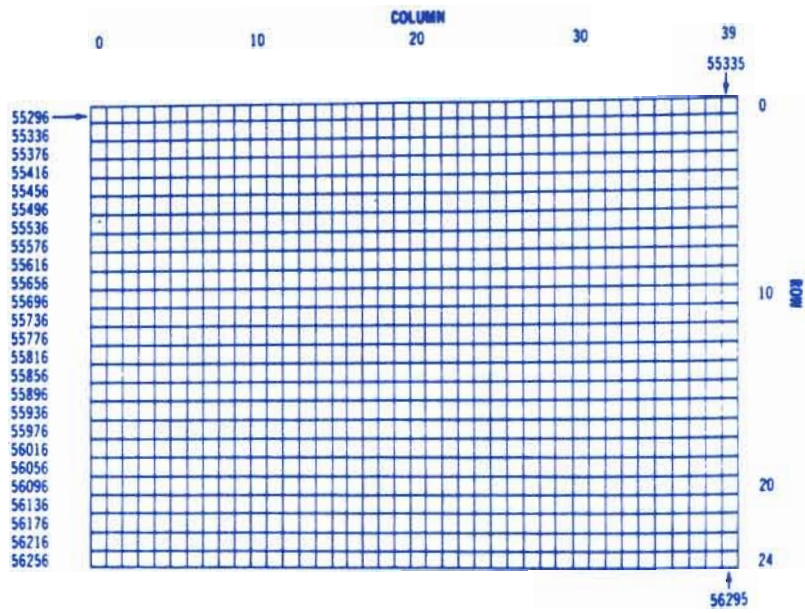
Codes from 128-255 are reversed images of codes 0-127.

Memory Maps



SCREEN CHARACTER MEMORY MAP

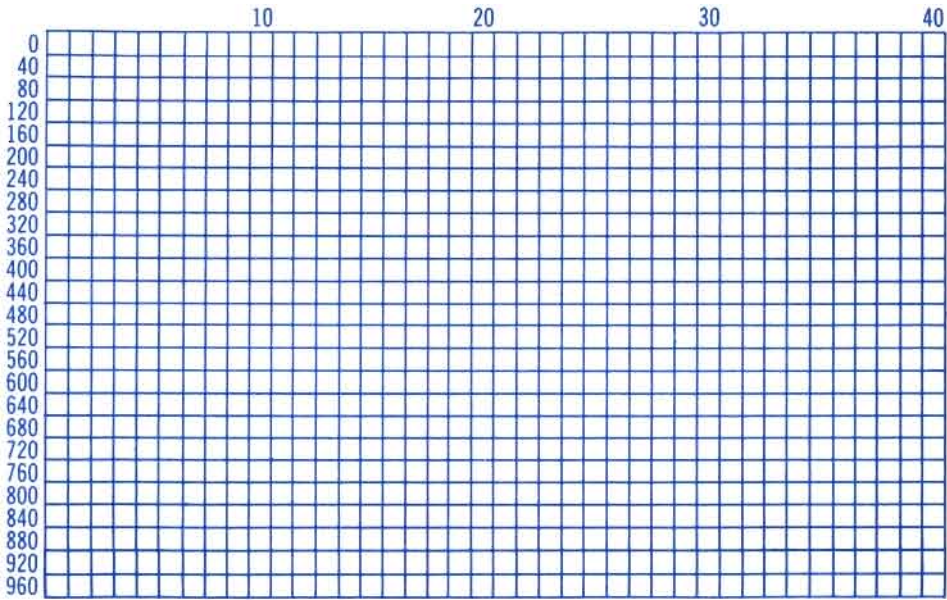
144 THE TOOL KIT SERIES Commodore 64 Edition



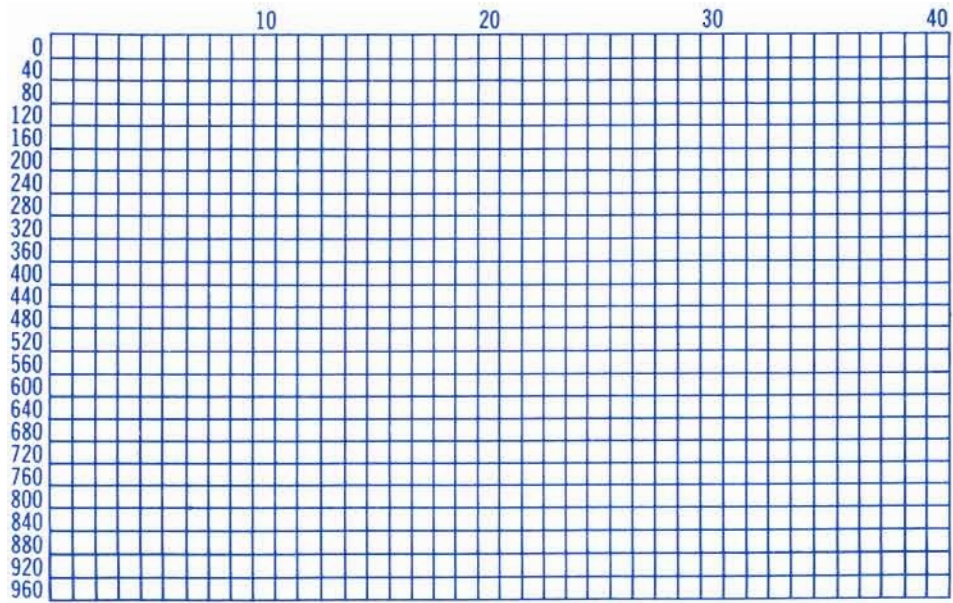
COLOR CODE MEMORY MAP

Sample Memory Map Graph Paper

D



146 THE TOOL KIT SERIES Commodore 64 Edition



ASCII and CHR\$ Codes



CODE	PRINTS	CODE	PRINTS	CODE	PRINTS	CODE	PRINTS
124		141		158		175	
125		142		159		176	
126		143		160		177	
127		144		161		178	
128		145		162		179	
129	Orange	146		163		180	
130		147		164		181	
131		148		165		182	
132		149	Brown	166		183	
133	f1	150	Lt. Red	167		184	
134	f3	151	Grey 1	168		185	
135	f5	152	Grey 2	169		186	
136	f7	153	Lt. Green	170		187	
137	f2	154	Lt. Blue	171		188	
138	f4	155	Grey 3	172		189	
139	f6	156		173		190	
140	f8	157		174		191	

Codes 192-223 same as 96-127
 Codes 224-254 same as 160-190
 Code 255 same as 126

148 THE TOOL KIT SERIES Commodore 64 Edition

CODE	PRINTS	CODE	PRINTS	CODE	PRINTS	CODE	PRINTS
0		31		62	>	93	}
1		32		63	?	94	↑
2		33	!	64	@	95	↑
3		34	"	65	A	96	
4		35	#	66	B	97	
5		36	\$	67	C	98	
6		37	%	68	D	99	
7		38	&	69	E	100	
8	DISABLES	39	.	70	F	101	
9	ENABLES	40	(71	G	102	
10		41)	72	H	103	
11		42	*	73	I	104	
12		43	+	74	J	105	
13		44	,	75	K	106	
14		45	-	76	L	107	
15		46	.	77	M	108	
16		47	/	78	N	109	
17		48	0	79	O	110	
18		49	1	80	P	111	
19		50	2	81	Q	112	
20		51	3	82	R	113	
21		52	4	83	S	114	
22		53	5	84	T	115	
23		54	6	85	U	116	
24		55	7	86	V	117	
25		56	8	87	W	118	
26		57	9	88	X	119	
27		58	:	89	Y	120	
28		59	;	90	Z	121	
29		60	<	91	[122	
30		61	=	92	£	123	

Common Error Messages

F

You will find errors inevitable. We have included here a list of the most common errors you may make using this book. For a more complete list of error messages, consult your User's Guide or the *Commodore 64 Programmer's Reference Guide*.

Error Message	Probable Cause
BAD SUBSCRIPT	You have dimensioned an array improperly. Look at the range in the DIM statement and check the number of variables of elements in the array.
BREAK	You stopped the program. Type CONT or RUN and the program will start again.
EXTRA IGNORED	You typed in too much data in response to an INPUT statement.
ILLEGAL QUANTITY	The number you have used as part of a statement or function is meaningless to the computer.
NEXT WITHOUT FOR	You have used FOR. .NEXT loops incorrectly. Check to see if the variable names match properly.
OUT OF DATA	You asked the program to READ data, but there was none left.
OUT OF MEMORY	1. You have depleted the computer's memory available to you. 2. The program you have written has too many subroutines (GOSUB statement) or too many FOR. .NEXT loops.
REDIM'D ARRAY	You have used the DIM statement wrong or you have used an array variable name before the variable was dimensioned.
REDO FROM START	You typed in letters when the computer asked you to input a number.

150 THE TOOL KIT SERIES Commodore 64 Edition

RETURN WITHOUT GOSUB	You have used a RETURN statement without a GOSUB command.
?SYNTAX ERROR	You have made a simple error in syntax—punctuation or spelling.
UNDEF'D STATEMENT	You sent the program (using GOTO or GOSUB) to a nonexistent line number.

Saving and Loading Programs



You can save the programs in this book or in any that you write by using the Datassette recorder or the Commodore VIC-1541 disk drive.

USING THE DATASSETTE RECORDER

In order to save a program on the Commodore 64, follow these two steps:

1. Use the SAVE command and the program name (which can be up to 16 characters in length). For example type:

SAVE "HANGMAN" and press **RETURN**

2. The screen will ask you to "Press play and record on tape" and you should press the play and record buttons at the same time. While saving, the Commodore 64's screen turns blank. You will know when it has finished saving; the word "READY" and a blinking cursor appear on the screen.

In order to load a program you have saved, follow the following three steps:

1. Rewind the tape and type

LOAD "HANGMAN" and press **RETURN**

2. The computer asks you to press play; you should press the Play button on the datassette recorder now.
3. The Commodore 64's screen turns blank. When the program is found, it displays

FOUND HANGMAN

At this point, you must press the Commodore key (**C**) to load the program. After the program has been loaded, "READING" and that familiar blinking cursor will appear.

USING THE DISK DRIVE

You must "FORMAT" a diskette before using it. The NEW command you use erases anything on the diskette and sets it up with markers and timing. This is how you do it:

```
PRINT#15, "NEW 0 : the disk name, ID"
```

```
PRINT#15
```

Is the command channel for communicating with the disk.

```
NEW 0
```

Clears out the directory on the diskette.

```
the disk name .
```

Identifies what is on the diskette.

```
ID
```

Is a 2-character identification name.

The FORMAT example is:

```
PRINT#15 "NEW 0:ARCADE GAMES, 10" and press RETURN
```

In formatting the diskette in this way, we have erased any programs, named it "Arcade games," and given it an ID number of 10. *NOTE:* You only format a diskette once. If you had to reformat the diskette each time, all the programs would be erased.

You must "INITIALIZE" a diskette every time you use it. This is simple to do. Type in:

```
PRINT#15, "INITIALIZE" and press RETURN
```

Remember that every time you put the diskette in the disk drive, you must initialize it.

In order to save a program on the diskette, follow this example. Type in:

```
SAVE "HANGMAN", 8 and press RETURN
```

"Hangman" is the name of the program and the 8 tells the computer to save the program on diskette. The computer will tell you that it has done so and the "READY" prompt will let you know when you can continue.

In order to load a program from a diskette, follow this example:

```
LOAD "HANGMAN", 8 and press RETURN
```

The computer will tell you it is searching for "HANGMAN" (or whatever your program name is), and it will tell you when it is loading, *and* it will tell you when you can use the computer again with the "READY" prompt.

There are a number of advanced features that can be utilized when using the Datasette Recorder or the Disk Drive. Instructions for their use are contained in the reference manuals that accompany these devices and we encourage you to take full advantage of their capabilities.

PEEK Values



When you want your program to see (PEEK) if a key on the keyboard is being pressed, use these codes in your program.

1	56	O	38
2	59	P	41
3	8	Q	62
4	11	R	17
5	16	S	13
6	19	T	22
7	24	U	30
8	27	V	31
9	32	W	9
0	35	X	23
+	40	Y	25
-	43	Z	12
£	48	*	49
CLR/HOME	51	↑	54
INST/DEL	0	[:	45
A	10]:	50
B	28	=	53
C	20	<,	47
D	18	>.	44
E	14	?/	55
F	21	RETURN	1
G	26	CRSR↑	7
H	29	CRSR←	2
I	33	F1	4
J	34	F3	5
K	37	F5	6
L	42	F7	3
M	36	-	57
N	39		
		No key pressed	64

Index

A

ABS(X), 23
 Accessories, 11–12
 Animate, how does the computer?, 76
 Animation, 39
 subroutines, 75–88
 Bouncing Ball, 81–82
 Duck Hunting With Bow and Arrow,
 84–86
 Mars Landing, 86–88
 Moving Jack, 80–81
 Space Commander, 83–84
 Speeding Up Jack, 82–83
 using
 POKE, 80–88
 strings in, 79
 the PRINT command, 76–78
 Arcade
 games
 Asteroid Shower, 134–136
 Pothole Derby, 132–134
 Rat Trap, 136–138
 Saucer Blaster, 130–132
 type sounds, 54–56
 Arguments, 23, 25, 26
 Arithmetic Tester game, 102–104
 ASCII
 and CHR\$ codes, 147–148
 code number, 24
 ASC(X\$), 24
 ATN(X), 24

B

Bar charts or graphs, 67–68
 BASIC, 15, 17–28
 COMMODORE, 18
 how do you write programs in?, 11
 Interpreter, 11
 Best way to start learning how to use the
 Commodore 64, 12
 Bit, 10
 Binary system, 10, 11
 Borders and screens, 31–33
 Bouncing Ball subroutine, 81–82
 Bow and Arrow, Duck Hunting With,
 84–86
 Branching statement, 20, 21
 Building blocks of Commodore BASIC, 15,
 17–28
 Bytes, 10, 24

C

Calculations, simple, 90–92
 Calculating subroutines, 89–100
 Current Ratio, 97, 99
 Dollars and Deutschmarks, 94–95
 Gallons and Liters, 92–93
 Inches and Centimeters, 94, 98
 Investing, 96–97
 Kilometers and Miles, 93–99
 Loan Analysis, 98
 Pounds, Ounces, and Kilograms, 93–94
 Profit, 97, 100
 Temperature, 93, 99
 Cassette tape, 23
 Central Processing Unit, 10
 Character codes, 67
 CHR\$ codes, 147–148
 CHR\$(X), 24
 CLOSE, 18
 CLR, 18
 CMD, 18
 Code(s)
 ASCII and CHR\$, 147–148
 binary, 10
 character, 67
 Color, 22, 29–39
 and sound together, putting, 56–57
 choice, 35–37
 code memory maps, 144
 in PRINT statements, 33–34
 keys, 30
 using the, 13
 random, 38
 subroutines, 29–39
 values, 32
 variable, 22
 Coloring, keyboard, 30–31
 Commands and statements, 11, 18–23, 28
 Commodore
 BASIC, 18
 64
 color, 29–39
 Organ, 50–52
 pictures with the, 68–74
 playing chords on, 49–50
 song book, 52–53
 CONT, 18
 Conversions, facts, figures, and, 89–100
 COS(X), 24
 CPU, 10

156 THE TOOL KIT SERIES Commodore 64 Edition

Creating graphics
 using POKE, 61–66
 with the PRINT command, 60–61
Current Ratio subroutine, 97, 99
Cursor keys, using the, 13

D

Data files, sound subroutines using,
 53–54
DATA statements, 19, 22
Datassette recorder, using the, 151
DEF FN, 19
Dice program, 72–73
DIM, 19
Direct mode, using the, 14–15
Disk drive
 Commodore, 12
 using the, 152
Diskette, 11, 12
Dollars and Deutschmarks subroutine,
 94–95
Duck Hunting With Bow and Arrow
 subroutine, 84–86

E

Editing on the Commodore 64, 13–14
Educational games
 Arithmetic Tester, 102–104
 Multiplication Tables, 104–106
 Painter's Palette, 111–113
 Rhyming Recognition, 107–108
 Riddle Fractions, 108–111
 Shape Recognition, 106–107
 States and Capitals, 113–116
Encyclopedia, sound, 45–46
END, 19
Error messages, common, 149–150
EXP(X), 24

F

Facts, figures, and conversions, 89–100
Floppy disk, 23
FOR/TO/STEP, 19
Frequency values of musical notes, 48
FRE(X), 24
Function(s), 23–26, 28

G

Gallons and Liters subroutine, 92–93
Games
 arcade, 129–138

Games—cont.
 educational, 101–117
 traditional, 119–128
GET statement, 19, 37, 38
G Major scale, 49
GOSUB, 19–20
GOTO, 20
Graph(s)
 or barcharts, 67–68
 paper memory maps, sample, 145–146
Graphics, 22, 39
 characters, using the, 12–13
 subroutine(s), 59–74
 Dice, 72
 House, 71
 Maze Maker, 73–74
 Person, 70
 Pine Tree, 69
 Score Box, 65
 Screen Border, 64–65
 Spacecraft, 71–72
 Three Dimensions, 73
 Tic-Tac-Toe, 121–123
Tic-Tac-Toe board, 66
 using POKE, creating, 61–66
 with the PRINT command, creating,
 60–61

H

House subroutine, 71
How does the computer animate?, 76
How do you write programs in BASIC?, 11

I

IF THEN, 20
Inches and Centimeters subroutine, 94
INPUT statement, 19, 20, 37, 38
Inputting
 procedures and hints, 26–28
 rules and suggestions, 27
Integer variables, 27, 28
Interpreter, BASIC, 11
INT(X), 24–25
Investing subroutine, 96–97

J

Jumping Jack subroutine, 76–78

K

Keyboard, 12
 coloring, 30–31

Kilometers and Miles subroutine, 93

L

Learning how to use the Commodore 64,
12
LEFT\$(X\$,X), 25
LEN(X\$), 25
LET statement, 20
LIST, 20–21
LOAD command, 14, 21
Loading programs, 151–152
Loan Analysis subroutine, 98
LOG function, 24
LOG(X), 25

M

Map(s)
color code memory, 63
screen, 62–63
character memory, 63
Mars Landing subroutine, 86–88
Maze Maker subroutine, 73–74
Memory, 10–11, 24
maps, 63
color code, 144
graph paper, 145–146
screen character, 143
random-access, 10–11
read-only, 10–11
storage, 11, 21, 23
Microprocessor, 6510, 10
MID\$ (\$,S,X), 25
Moving Jack subroutine, 80–81
Multiplication Tables game, 104–106
Music
lesson subroutines, 47–49
subroutine, 41–57
Musical notes, 140
frequency value of, 48

N

NEW, 21
NEXT statement, 19, 21
Noise Maker subroutine, 45
Numeric
functions, 23
values for a string, 26
variable, 27, 28

O

Octaves, 140
ON GOSUB, 21
ON GOTO, 21
OPEN, 21
Organ, Commodore 64, 50–52

P

Painter's Palette game, 111–113
Painting the screen, 31
Parking Lot subroutine, 90–92
PEEK values, 153
PEEK(X), 25
Person subroutine, 70
Pictures with the Commodore 64, 68–74
Pine Tree subroutine, 69
Playing chords on the Commodore 64,
49–50
POKE, 22, 69–72, 141–142
animation using, 80–88
creating graphics using, 61–66
POKEing, 31–32
more than one voice, 46–47
sound, 42–44
POS(X), 25
Pounds, Ounces, and Kilograms
subroutine, 93–94
PRINT, 22
command
animation using the, 76–78
creating graphics with the, 60–61
statement(s), 26
color in, 33–34
Profit subroutine, 97, 100
Putting sound and color together, 56–57

R

RAM, 10–11, 12
Random
access memory, 10–11
see also RAM
color, 38
function, 26
number, 26
READ, 22
Read-only memory, 10–11
see also, ROM
REM, 22
RESTORE, 22

158 THE TOOL KIT SERIES Commodore 64 Edition

RETURN statement, 19, 21, 22
Reverse mode, 13
Rhyming Recognition game, 107–108
Riddle Fractions game, 108–111
RIGHT\$(X\$, @), 26
RND(X), 26
ROM, 10–11, 12
RUN, 22

S

SAVE command, 23
Saving and loading programs, 151–152
Scale, G Major, 49
Score Box subroutine, 65
Screen(s)
 and borders, 31–33
 border, 64–65
 Border subroutine, 64–65
 character memory maps, 143
 maps, 62–63
 painting the, 31
 values, 141–142
SGN(X), 26
Shape Recognition game, 106–107
Simple calculations, 90–92
SIN(X), 26
6510 microprocessor, 10
Song book, Commodore, 52–53
Sound(s), 22, 39
 and color together, putting, 56–57
 arcade-type, 54–56
 encyclopedia, 45–46
 POKEing, 42–44
 subroutines, 41–57
 using data files, 53–54
 variables, 45
Space Commander subroutine, 83–84
Spacecraft program, 71–72
SPC(X), 26
Speeding Up Jack subroutine, 82–83
SQR(X), 26
Statements and commands, 18–23, 28
States and Capitals game, 113–116
STEP command, 19
STOP, 23
STR\$(X), 26
String(s)
 functions, 23
 variable, 21, 24, 27–28
String variable, 23, 24

String variable—cont.
 in animation, using, 79
Subroutines
 animation, 75–88
 calculating, 89–100
 color, 35
 graphics, 59–74
 music, 41–57
 lesson, 47–49
 Noise Maker, 45
 sound, 41–57
 using data files, sound, 53–54

T

TAB function, 26
TAB(X), 26
TAN(X), 26
Tape, 11, 12
Temperature subroutine, 94
Three Dimensions subroutine, 73
Tic-Tac-Toe board, 66
Traditional games
 Guessing Game, 120–121
 Hangman, 126–128
 Ted and Dave's Casino, 123–126
 Tic-Tac-Toe, 121–123

U

Using
 strings in animation, 79
 the color keys, 13
 the Commodore
 direct mode, 14–15
 graphics characters, 12
 the cursor keys, 13
 the Datasette recorder, 151
 the disk drive, 152
 the reverse function, 13

V

VAL(X\$), 26
Variable(s), 19, 27–28
 color, 22
 loudness, 22
 sound, 45
 string, 23, 24
VERIFY, 23

W

Writing programs in BASIC, 11



More Books for Commodore 64 Owners!

COMMODORE 64 PROGRAMMER'S REFERENCE GUIDE

A creative programmer's working tool and reference source, packed with information on all aspects of Commodore BASIC programming. By Commodore Computer. 486 pages, 5½ x 8½, comb-bound. ISBN 0-672-22056-3. © 1982.

No. 22056\$19.95

LEARN BASIC PROGRAMMING IN 14 DAYS ON YOUR COMMODORE 64

Consists of 14 chapters intended to be covered at the rate of one per day or one per sitting. Especially good for those aged 9 thru 19, but works fine for adults, too. Gil Schechter. 208 pages, 5½ x 8½, comb-bound. ISBN 0-872-22279-5. © 1984.

No. 22279\$10.95

MOSTLY BASIC: APPLICATIONS FOR YOUR COMMODORE 64, Book 1

Brings you 38 chapters filled with fun-and-serious BASIC programs that help you save money on energy usage, bar-chart your business sales, dial the telephone, learn a foreign language, and more. By Howard Berenbon. 192 pages, 8½ x 11, comb-bound. ISBN 0-672-22355-4. © 1984.

No. 22355\$12.95

MOSTLY BASIC: APPLICATIONS FOR YOUR COMMODORE 64, Book 2

This second collection of Commodore 64 BASIC programs includes dungeons, memory challengers, a student grader, phone directory, monthly budget, ESP tests, and more. By Howard Berenbon. 264 pages, 8½ x 11, comb-bound. ISBN 0-672-22356-2. © 1984.

No. 22356\$13.95

COMMODORE 64 GRAPHICS AND SOUNDS

Helps you quickly learn and use the Commodore 64's powerful graphic and sound capabilities in a number of spectacular routines that add pizzazz to any program. By Tim Knight. 112 pages, 5½ x 8½, softbound. ISBN 0-672-22278-7. © 1984.

No. 22278\$9.95

COMMODORE 64 BASIC PROGRAMS

Generously illustrated collection of thoroughly documented and described, fun-and-practical programs for the powerful Commodore 64. By Knight and LaBatt.

BOOK: 176 pages, 5½ x 8½, softbound. ISBN 0-672-22171-3. © 1983.

No. 22171\$9.95

BOOK/TAPE: ISBN 0-672-26171-5.

No. 26171\$16.95

COMMODORE SOFTWARE ENCYCLOPEDIA (3rd Edition)

Completely updated, highly comprehensive directory of software and more for the Commodore 64, VIC-20, PET, and other Commodore computers. By Commodore Computer. 896 pages, 8½ x 11, softbound. ISBN 0-672-22091-1. © 1983.

No. 22091\$19.95

COMMODORE 64 USER'S GUIDE

Shows you how to set up, program, and operate your 64 (same book that comes packed with the computer itself). By Commodore Computer. 166 pages, 5½ x 8½, comb-bound. ISBN 0-672-22010-5. © 1982.

No. 22010\$12.95

USER'S GUIDE TO MICROCOMPUTER BUZZWORDS

A handy quick-reference for those people who don't care what happens inside a microcomputer, yet find they must communicate with others who do. Many illustrations. By David H. Dasenbrock. 110 pages, 5½ x 8½, softbound. ISBN 0-672-22049-0. © 1983.

No. 22049\$9.95

COMPUTER LANGUAGE REFERENCE GUIDE (2nd Edition)

If you know at least one programming language, this newly updated reference can help you understand eight more, including C and FORTH. By Harry L. Helms, Jr. 192 pages, 5½ x 8½, softbound. ISBN 0-672-21823-2. © 1984.

No. 21823\$9.95

USING COMPUTER INFORMATION SERVICES

Shows you how to use your microcomputer to communicate with the national computer networks and their wide range of services. By Sturtz and Williams. 240 pages, 5½ x 8½, softbound. ISBN 0-672-21997-2. © 1983.

No. 21997\$12.95

ELECTRONICALLY SPEAKING: COMPUTER SPEECH

GENERATION

Teaches you the basics of generating synthetic speech with a microcomputer. Includes techniques, a synthesizer overview, advice on problems, and more. By John P. Cater. 232 pages, 5½ x 8½, softbound. ISBN 0-672-21947-6. © 1982.

No. 21947\$14.95

EXPERIMENTS IN ARTIFICIAL INTELLIGENCE FOR SMALL COMPUTERS

You'll conduct interesting and exciting experiments in artificial intelligence, such as reasoning, creativity, problem-solving, verbal communication, game playing, and more. By John Krutch. 112 pages, 5½ x 8½, softbound. ISBN 0-672-21785-6. © 1981.

No. 21785\$9.95

HOW TO MAINTAIN AND SERVICE YOUR SMALL COMPUTER

Shows you some easy maintenance and operating procedures, plus how to diagnose what's wrong, how to identify the faulty part, and how to make many simple, money-saving repairs yourself. By John G. Stephenson and Bob Cahill. 224 pages, 8½ x 11, softbound. ISBN 0-672-22016-4. © 1983.

No. 22016\$17.95

These and other Sams Books and Software products are available from better retailers worldwide, or directly from Sams. Call 800-428-SAMS or 317-298-5566 to order, or to get the name of a Sams retailer near you. Ask for your free Sams Books and Software Catalog!

Prices good in USA only. Prices and page counts subject to change without notice.

Sams Books cover a wide range of technical topics. We are always looking for more information from you, our readers, as to which additional topics need coverage. Please fill out this questionnaire and return it to us with your suggestions. They will be appreciated.

Please check the areas of interest:

1. CURRENT TECHNOLOGIES

- Electronics
- Circuit Design
- Computers
 - Business Applications
 - Fundamentals
 - Languages _____
Specify _____
 - Machine Specific: _____

- Microprocessors
- Networking
- Servicing/Repair
- _____
Other

2. NEW TECHNOLOGIES

- Fiber Optics
- Robotics
- Security Electronics
- Speech Synthesis
- Telecommunications
 - Cellular
 - Satellite
- Video
- Other _____

3. Do you own operate a personal computer? Model _____

4. Have you bought other Sams Books? Please list: _____

5. OCCUPATION

- Business Professional _____
Specify _____
- Educator
- Engineer _____
- Hobbyist _____
Specify
- Programmer
- Retailer
- Student
- _____
Other

6. EDUCATION

- High School Graduate
- Tech School Graduate
- College Graduate
- Post-graduate degree

COMMENTS _____

(OPTIONAL)

NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

Book Markers

SAMS™

Book Markk™

SAMS™



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS

PERMIT NO. 1076

INDIANAPOLIS, IND.

POSTAGE WILL BE PAID BY ADDRESSEE

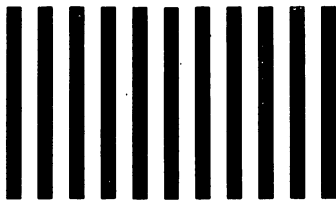
HOWARD W. SAMS & CO., INC.

4300 WEST 62ND STREET

P.O. Box 7092

Indianapolis, IN 46206

ATTENTION: Public Relations Department



The Tool Kit Series Commodore 64™ Edition

To become computer literate and to keep your Commodore 64 computer purring along properly, you must have the correct tools. The "tools" in this book are short 5- to 15- line subroutines that combine color, sound, and graphics to form a variety of educational programs and computer games.

- Forget the structured method of learning programming. Forget commands, statements, and data structures.
- Forget those frustrating evenings with your User's Manual.
- Learn to look at programs in terms of their working parts—their subroutines—and learn how to write simple programs.
- Learn how to use color, sound and music, graphics, animation, and computational subroutines as "tools" to build programs.
- Discover the modular form of programming. Learn what each subroutine will do, how it can be changed, and what the variables control.

Don't spend long frustrating hours poring over programming books and manuals. Follow the "Tool Kit" approach to programming and learn to design your own games and quizzes. Keep your Commodore 64 computer humming happily.

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

\$9.95/22314

ISBN: 0-672-22314-7