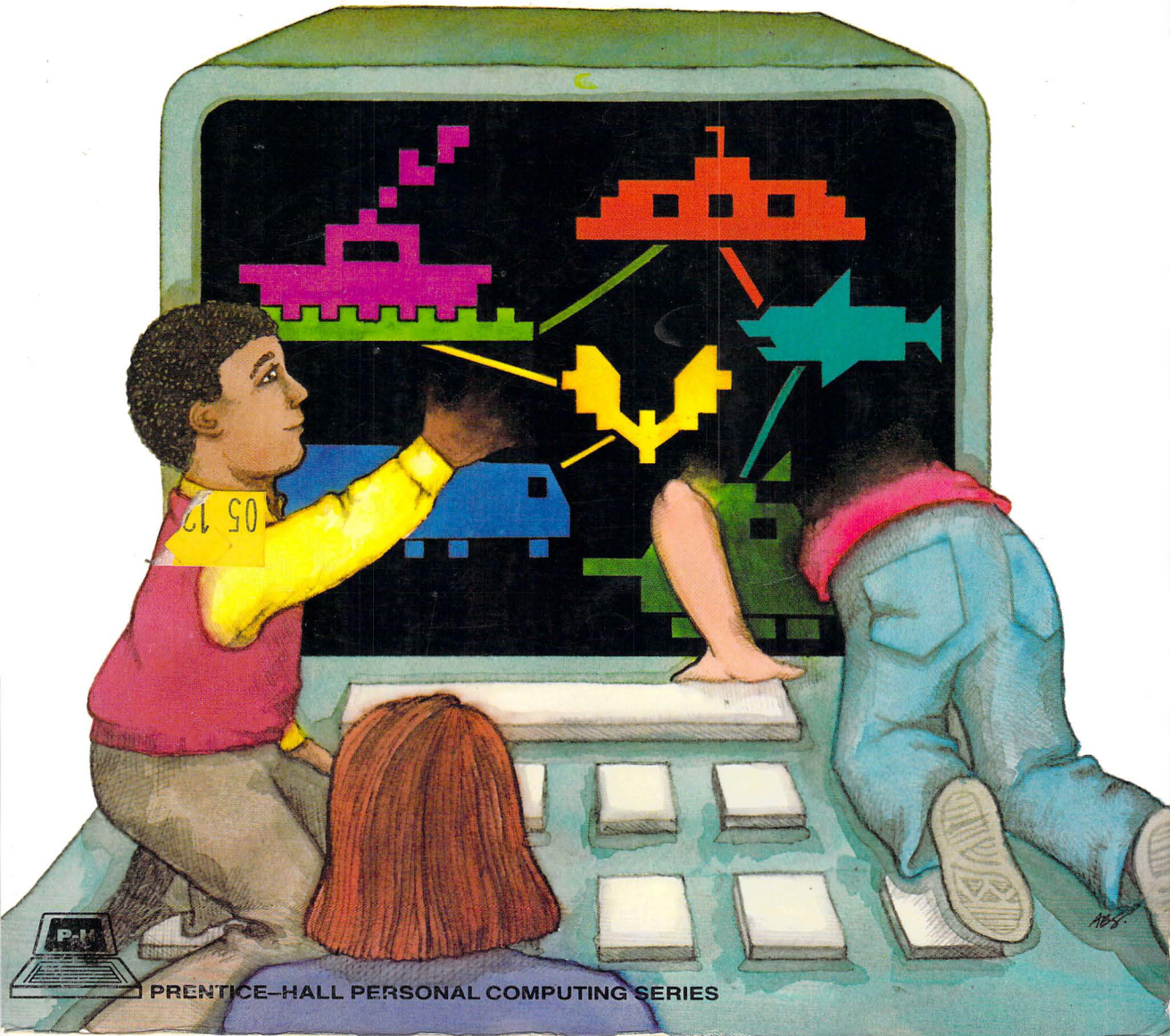


**TONY FABBRI**

**ANIMATION, GAMES, & SOUND FOR THE**

# 64 COMMODORE



PRENTICE-HALL PERSONAL COMPUTING SERIES



ANIMATION, GAMES,  
AND SOUND  
FOR THE  
COMMODORE 64

Prentice-Hall Personal Computing Series  
Lance A. Leventhal, series editor

- FABBRI, *Animation, Games, and Graphics for the Timex-1000*  
FABBRI, *Animation, Games, and Sound for the Apple III/IIe*  
FABBRI, *Animation, Games, and Sound for the Commodore 64*  
FABBRI, *Animation, Games, and Sound for the IBM PC*  
FABBRI, *Animation, Games, and Sound for the TI 99/4A*  
FABBRI, *Animation, Games, and Sound for the VIC-20*  
HARRIS & SCOFIELD, *IBM PC Conversion Handbook of BASIC*  
SCANLON, *EasyWriter II System Made Easy-er*  
SCANLON, *The IBM PC Made Easy*  
SCHNAPP & STAFFORD, *Commodore 64 Computer  
Graphics Toolbox*  
SCHNAPP & STAFFORD, *Computer Graphics for the Timex 1000  
and Sinclair ZX-81*  
SCHNAPP & STAFFORD, *VIC 20 Computer Graphics Toolbox*  
THRO, *Making Friends With Apple Writer II*



# ANIMATION, GAMES, AND SOUND FOR THE COMMODORE 64

**Tony Fabbri**

**Prentice-Hall, Inc.**

*Englewood Cliffs, New Jersey 07632*

*Library of Congress Cataloging in Publication Data*

Fabbri, Tony. (date)

Animation, games, and sound for the Commodore 64.

Includes index.

1. Computer graphics. 2. Computer animation. 3. Computer music. 4. Computer games. 5. Commodore 64 (Computer)--Programming. I. Title.

T385.F28 1984 794.8'2 84-6756

ISBN 0-13-037375-3

ISBN 0-13-037367-2 (book/disk)

Editorial/production supervision: **Karen Skrable Fortgang**

Interior design: **Kathryn Gollin Marshak**

Cover design: **Jeannette Jacobs**

Manufacturing buyer: **Gordon Osbourne**

**Commodore 64 is a trademark of Commodore Electronics Limited.**

©1984 by **Prentice-Hall, Inc.**, Englewood Cliffs, New Jersey 07632

All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from the publisher.

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-037375-3

ISBN 0-13-037367-2 {BK/DISK} 01

Prentice-Hall International, Inc., *London*  
Prentice-Hall of Australia Pty. Limited, *Sydney*  
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*  
Prentice-Hall Canada Inc., *Toronto*  
Prentice-Hall of India Private Limited, *New Delhi*  
Prentice-Hall of Japan, Inc., *Tokyo*  
Prentice-Hall of Southeast Asia Pte. Ltd., *Singapore*  
Whitehall Books Limited, *Wellington, New Zealand*

# CONTENTS

	<b>EDITOR'S FOREWORD</b>	<b>xiii</b>
	<b>PREFACE</b>	<b>xv</b>
<b>Chapter 1</b>	<b>INTRODUCTION</b>	<b>1</b>
<b>Chapter 2</b>	<b>SIMPLE PRINTING</b>	<b>11</b>
	Program 2-1 Print Animal Names 12	
	Program 2-2 Print DOG 20 Times 13	
	Program 2-3 Printing Your Name 10 Times 14	
	Program 2-4 Your Name in Columns 14	
	Program 2-5 Your Name Filling Screen 15	
	Program 2-6 Your Name Fancy 15	
	Program 2-7 Your Name Diagonal 18	
	Program 2-8 Friend's Name Diagonal 19	
	Program 2-9 Two Names in Crossing Pattern 20	
<b>Chapter 3</b>	<b>DRAWING PICTURES</b>	<b>22</b>
	Program 3-1 Face Moving Right 22	
	Program 3-2 Creature Moving Right 24	
	Program 3-3 Two Faces Moving Right 24	
	Program 3-4 Monster Face Moving Left 25	

Program 3-5	Gorilla Moving Left	25	
Program 3-6	Car Moving Left	26	
Program 3-7	Airplane Moving Right	26	
Program 3-8	Sailboat Moving Left	26	
Program 3-9	Tank Moving Left	27	
Program 3-10	Alien Face Moving Back and Forth		27
Program 3-11	Bug Moving Back and Forth	28	
Program 3-12	Cow's Face Moving Back and Forth Three Times	29	
Program 3-13	Tank Moving Back and Forth Four Times	30	

## Chapter 4

### SOME THINGS MOVE WHILE OTHERS REMAIN STATIONARY

31

Program 4-1	Small Rocket Moving Right	31	
Program 4-2	Tank Firing Rocket	31	
Program 4-3	Airplane Firing Rockets	32	
Program 4-4	Artillery Gun Firing Rocket Right		32
Program 4-5	Arrow Shot at Target	33	
Program 4-6	Car Driving Through Town	33	
Program 4-7	Bee Flying Left to Flower	34	
Program 4-8	Bee Flying Right to Flower	35	
Program 4-9	Moving Ball, Without Erasure	35	
Program 4-10	Moving Ball, With Erasure	36	
Program 4-11	Game of Catch	37	

## Chapter 5

### ROCKETS FLY, BLAST OFF, AND CHASE ONE ANOTHER

38

Program 5-1	Rocket	38	
Program 5-2	Rocket Flying Fast	39	
Program 5-3	Enemy Rocket Chasing Two Rockets, With Smoke	39	
Program 5-4	Alien Spacecraft Flying, With Smoke		40

## Chapter 6

### CREATING MOTION PICTURES

42

Program 6-1	Moving Bug	43	
Program 6-2	Moving Bug With Blinking Eyes and Open Mouth	44	
Program 6-3	Figure Walking	44	

Program 6-4	Funny, Moving Aliens	45	
Program 6-5	Stationary Creature Making Faces		45
Program 6-6	Horse Grazing	46	
Program 6-7	Exploding Stars	47	

**Chapter 7                    POSITIONING FIGURES                    49**

Program 7-1	Moving Asterisk Using TAB	49	
Program 7-2	Asterisk Moving Left, Using Cursor Control	51	
Program 7-3	Alien Moving Right, Using Cursor Control	53	
Program 7-4	Face Moving Right, Using Cursor Control	54	
Program 7-5	Names All Over	55	
Program 7-6	Lunar Station Firing a Shell		57
Program 7-7	Lunar Station Firing a Shell, With Erasure	57	

**Chapter 8                    USING GRAPHICS CHARACTERS                    58**

Program 8-1	Simple Box	58	
Program 8-2	Graphics Rocket	60	
Program 8-3	Alien Spaceship With Smoke, Using Graphics	60	
Program 8-4	Moving Car	61	
Program 8-5	Moving Graphics Face	61	
Program 8-6	Moving, Blinking Eyes	61	
Program 8-7	Graphics Tank Moving Left		62
Program 8-8	Graphics Truck/Trailer Moving Left	63	

**Chapter 9                    PLACING PICTURES RANDOMLY                    64**

Program 9-1	Snowfall	64	
Program 9-2	Faces Appear and Disappear at Random Positions	66	
Program 9-3	Random Characters at Random Positions	68	
Program 9-4	Exploding Supernovas at Random Positions	69	
Program 9-5	Random Four-Letter-Word-Generator		71



**Chapter 10                    PERSONAL MESSAGES: NAMES AND CARDS                    73**

Program 10-1	Printing Your Name 20 Times	74
Program 10-2	Your Name Moving Right	74
Program 10-3	Input Names in Crossing Pattern	75
Program 10-4	Valentine's Day Card	76
Program 10-5	Christmas Card	76
Program 10-6	New Year's Card	77
Program 10-7	Birthday Card	77
Program 10-8	Halloween Card	78
Program 10-9	Halloween Cat	79
Program 10-10	Mother's Day Card	79
Program 10-11	Easter Card	79
Program 10-12	Easter Bunny	80

**Chapter 11                    THE PERSONAL TOUCH                    81**

Program 11-1	Personalized Story	81
Program 11-2	Homework Assignment Notice	82
Program 11-3	Legal Notice with Three Names	83
Program 11-4	Extra! Extra! Read All About It!	84
Program 11-5	Computer Interviewer	85
Program 11-6	Doctor's Assistant	86
Program 11-7	Tax Collector's Assistant	86
Program 11-8	Simple Arithmetic Quiz	87
Program 11-9	Arithmetic Quiz With Two Questions	88
Program 11-10	Geography Quiz With an Alphabetic Question	88
Program 11-11	Geography Quiz With Several Questions	88

**Chapter 12                    CONTROLLING ACTION FROM THE KEYBOARD                    89**

Program 12-1	Lunar Station Firing Missile, Under Keyboard Control	90
Program 12-2	Eye Creatures Moving Right	91
Program 12-3	Target Practice	92

**Chapter 13                    MAKING NOISE                    94**

Program 13-1	Single Note	96
Program 13-2	Rocket Sound Using White Noise	98
Program 13-3	Rocket Launch With Sound	98

Program 13-4	Eye Creatures Making Noises	100
Program 13-5	Alien With Strange Sound	101
Program 13-6	Blinking, Moving Alien Making Noise	101
Program 13-7	Police Car With Flashing Lights and Siren	102
Program 13-8	Alien Spacecraft With Engine Noise	102
Program 13-9	Tank Command With Strange Noises	103

**Chapter 14      FALLING OBJECTS      105**

Program 14-1	Falling Asterisk Without Erasure	106
Program 14-2	Falling Asterisk With Erasure	106
Program 14-3	Falling Spider	108
Program 14-4	Blinking, Falling Spider	109
Program 14-5	Blinking, Falling, Noisy Spider	110
Program 14-6	Race Car Moving Down the Screen	111
Program 14-7	Race Car With Loud Mufflers	111
Program 14-8	Falling Bug	112
Program 14-9	Infestation of Noisy Falling Bugs	112
Program 14-10	Bug Moving Up and Down	113
Program 14-11	Paratropper Landing	115

**Chapter 15      BOMBING GAME      116**

Program 15-1	Falling Bomb	116
Program 15-2	Bomb Falling to Ground, With Noise	117
Program 15-3	Small Bomb Falling, With Noise	118
Program 15-4	Bombing Under Keyboard Control	119
Program 15-5	Moving Bomb Before Dropping It	120
Program 15-6	Bombs-Away Game	122
Program 15-7	Enemy Airplane Dropping Bombs	123
Program 15-8	Enemy Airplane Dropping Bombs, With Explosion	124

**Chapter 16      MOVING THINGS DIAGONALLY      126**

Program 16-1	Eye Creature Moving Right Diagonally, With Shadow	126
Program 16-2	Eye Creature Moving Diagonally, Without Shadow	127
Program 16-3	Alien Moving Right Diagonally, With Sound	128
Program 16-4	Asterisk Moving Diagonally	129

	Program 16-5	Pulsating Alien Moving Right Diagonally	130
	Program 16-6	Eye Creature Moving Left Diagonally	131
	Program 16-7	Eye Creature Moving Left Diagonally, Without Shadow	132
	Program 16-8	Alien Moving Left Diagonally	132
	Program 16-9	Alien Moving Right Diagonally, From Bottom	134
	Program 16-10	Alien Moving Left Diagonally, From Bottom	134
<b>Chapter 17</b>	<b>SPACESHIP GAME</b>		<b>135</b>
	Program 17-1	Moving the Master Spaceship	135
	Program 17-2	Master Ship Attacking With Guided Bombs	136
	Program 17-3	Spaceship Game With Speed Control	138
	Program 17-4	Spaceship Game With Speed Control and Explosions	139
<b>Chapter 18</b>	<b>CONTROLLING MOTION IN ALL DIRECTIONS</b>		<b>141</b>
	Program 18-1	Alien Moving in Four Directions	142
	Program 18-2	Rocket Moving Anywhere	145
	Program 18-3	Rocket Moving Anywhere, With Sound	147
	Program 18-4	Rocket Moving Anywhere, With Sound and Targets	147
	Program 18-5	Rocket Game With Momentum	149
<b>Chapter 19</b>	<b>TARGET PRACTICE</b>		<b>151</b>
	Program 19-1	Shell Rising	152
	Program 19-2	Shell Rising and Falling	153
	Program 19-3	Artillery Shelling a Factory	154
	Program 19-4	Artillery Shelling Factory, With Noise and Explosion	156
<b>Chapter 20</b>	<b>SIMULTANEOUS MOTION IN TWO DIRECTIONS</b>		<b>159</b>
	Program 20-1	Ship and Submarine Moving in Different Directions	160
	Program 20-2	Ship, Submarine, Ocean Waves, and Birds	161

Program 20-3	Submarine Firing Projectile	162
Program 20-4	Ocean, Birds, Ship, and Submarine, With Launch	163

**Chapter 21      MUSICAL INTERLUDE      166**

Program 21-1	Musical Quarter Notes	166
Program 21-2	Musical Scale in Octave 3	168
Program 21-3	"Old MacDonald's Farm"	172
Program 21-4	"Yankee Doodle Dandy"	174
Program 21-5	Stick Figure Conducting "Yankee Doodle Dandy"	175
Program 21-6	Dancing Stick Figure	177
Program 21-7	Stick Figure Dancing to "Yankee Doodle Dandy"	179
Program 21-8	Stick Figure Dancing to Song, With Bow and Applause	180

**Chapter 22      RACE CAR GAME      181**

Program 22-1	Beginner's Racetrack	181
Program 22-2	Car Racing Around Track	183
Program 22-3	Intermediate Car Race	185
Program 22-4	Car Race, With Sound	185
Program 22-5	Car Race, With Advanced Racetrack	186

**Chapter 23      ADD A LITTLE COLOR TO YOUR LIFE      188**

Program 23-1	Colorful Alien Moving Right	188
Program 23-2	Color Names in Color	190
Program 23-3	Multicolored Alien Moving Right	190
Program 23-4	Black Horse Grazing on Green Grass	191
Program 23-5	Horse of a Different Color	192
Program 23-6	Moving Alien with Varying Colors	192
Program 23-7	Space Station Changing Color	193
Program 23-8	Multicolored Dot Moving in a Circle	194
Program 23-9	Random Art With Color	195
Program 23-10	Additional Color Names in Color	196
Program 23-11	Kaleidoscope of Screen and Border Colors	197
Program 23-12	Neon Sign With Moving Lights	198
Program 23-13	James Bond's Secret Orders	199

<b>Chapter 24</b>	<b>REVERSING THE ODDS</b>	<b>200</b>
	Program 24-1      Your Name in Reverse      200	
	Program 24-2      Your Name in Reverse, Moving Left      201	
	Program 24-3      Your Name Flashing      201	
	Program 24-4      Abandon Ship      202	
	Program 24-5      Message Flasher      202	
	Program 24-6      Blimp With Blinking Message      204	
	Program 24-7      Keep on Trucking      205	
	Program 24-8      Tanks for the Memory      205	
	Program 24-9      Little House on the Prairie      206	
	Program 24-10      Little Red Schoolhouse      207	
	Program 24-11      Maze Builder      207	
	Program 24-12      Rolling Ocean Waves      208	
	Program 24-13      Rolling Ocean Waves on the Red Sea      209	
	Program 24-14      Parting the Red Sea      209	
<b>Appendix A</b>	<b>MUSIC NOTE VALUES</b>	<b>211</b>
<b>Appendix B</b>	<b>MUSIC REVIEW</b>	<b>214</b>
<b>Appendix C</b>	<b>COMMODORE 64 DISC OPERATIONS</b>	<b>216</b>
	<b>INDEX</b>	<b>219</b>



# EDITOR'S FOREWORD

What can you say about a book with program titles like

Gorilla Moving Left  
Car Driving Through Town  
Moving Bug With Blinking Eyes and Open Mouth  
Lunar Station Firing a Shell  
Graphics Truck/Trailer Moving Left  
Exploding Supernovas at Random Positions  
Random Four-Letter-Word Generator  
Halloween Cat  
Extra! Extra! Read All About It!  
Tax Collector's Assistant  
Police Car With Flashing Lights and Siren  
Blinking, Falling, Noisy Spider  
Paratrooper Landing  
Artillery Shelling a Factory, With Noise and Explosion  
Ship, Submarine, Ocean Waves, and Birds  
Stick Figure Dancing to "Yankee Doodle Dandy"  
Black Horse Grazing on Green Grass  
James Bond's Secret Orders  
Little Red Schoolhouse  
Parting the Red Sea

Tony Fabbri's programs speak for themselves. This book is light, enjoyable, easy to read, and probably addictive as well. For those serious souls who worry about wasting valuable time, let me add that the book also teaches BASIC and good programming practices. What more can you ask? I expect that you will have as good a time with this book as I did. Please remember to tear yourself away from your computer every once in a while.

Lance A. Leventhal  
*San Diego, CA*

# PREFACE

While many books deal with computers, hardly any are just plain fun to read and apply. This book does not explain the insides of computers or describe all possible ways to use them. Rather, it lets the reader do exciting things with the computer and see action unfold on the screen. It assumes no prior knowledge of computers or programming. The reader will learn about computers and enjoy doing it.

After covering a few fundamentals, the book slowly leads the beginner through drawing and moving pictures such as aliens, rockets, animals, insects, tanks, cars, and spacecraft. Then we introduce sound, and the pictures come to life. Spaceships, airplanes, tanks, and strange creatures make noises as they move, turn, fire, blink, and attack. We then describe how to control the action from the keyboard. The reader now is able to create his or her own simple arcade games. Finally, we explain how to put music and color in games to produce even more exciting adventures.

All the discussions and early programs are short. The reader will find it easy to extend the examples to create new games. A significant fringe benefit of this fun is that the reader will actually be learning the skills needed for practical applications in business, education, engineering, management, and science.

This book uses the popular Commodore 64 computer. No special graphics equipment is necessary, since we draw all pictures with ordinary PRINT statements. The reader can simply enter a program and watch the action on the screen.

We hope this book will provide many hours of enjoyment for its readers, both as a learning tool and as a reference. We also hope it encourages readers to build a comfortable relationship with the computer, thus freeing their imaginations to use it creatively. This is the key to obtaining the greatest benefit from today's amazing personal computers.

Tony Fabbri



# 1

## INTRODUCTION

The Commodore 64 is a popular, inexpensive personal computer. Although it is no bigger than a loaf of bread, it is as powerful as computers of the 1950s and 1960s that occupied entire rooms and cost thousands of dollars. Together with a television set (preferably color) and a cassette recorder or disk drive, it forms a complete computer system. Among its many uses are:

1. Word processing (electronic writing of letters, notices, reports, and books without erasures, misspellings, and typing errors).
2. Business and financial calculations, such as figuring interest rates, loan payments, rates of return, and cost of capital.
3. Computerized lessons in arithmetic, spelling, social studies, and language.
4. Space, adventure, sports, gambling, word, mathematical, and skill games.
5. Record keeping for collectors, hobbyists, investors, property owners, and officers in local clubs and organizations.
6. Preparing a home budget.
7. Analyzing and tracking investments such as stocks, bonds, real estate, and commodities. The Commodore 64 can even obtain current quotations and financial news over the telephone.
8. Balancing checkbooks, helping with tax forms, calculating percentages and discounts, and performing other home financial tasks.
9. Performing engineering calculations to determine thrust, inertia, stress, energy use, distances, and flow rates.
10. Controlling things such as lights, motors, generators, household appliances, and laboratory or shop equipment.
11. Creating art, music, and sound effects.



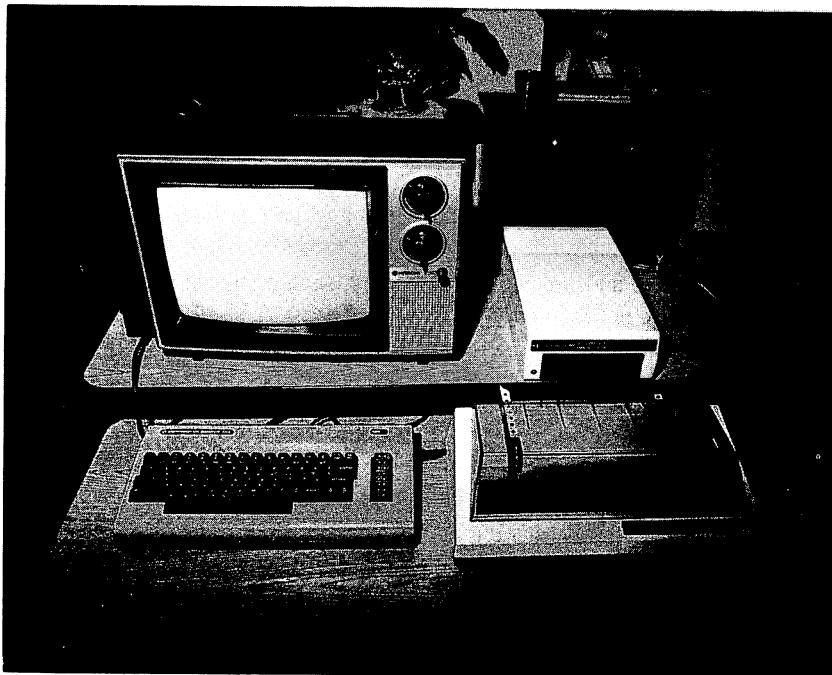
12. Electronic filing of names and addresses, receipts, grades and attendance records, recipes, and appointments.

These are only a few of the Commodore 64's many uses. Hundreds of thousands of people have purchased Commodores for business, educational, recreational, or personal use.

## **COMPONENTS OF A COMMODORE 64**

A standard Commodore 64 (see Fig. 1-1 for a photograph) consists of two pieces of equipment (we call this the *hardware*):

1. The computer itself—a rectangular, beige plastic box with a sloping top. Near the front are dark brown keys arranged like a typewriter. On the right are a few extra dark gray keys. Near the top right-hand corner is a small red power indicator.
2. A television set that serves as an output device or display unit. We call this the *video display*.



**Figure 1-1** Picture of Commodore 64 Computer System. (Courtesy of Commodore Electronics Limited.)

You can buy many accessories for the Commodore 64; among the most popular are

1. Cassette recorder. This lets you play cassettes into the computer and record them from it.
2. Disk drive. This is a small box with a rectangular slit in the front. It serves the same purpose as a cassette recorder but instead of a cassette tape, it uses a thin, flexible disk about the size of a 45-rpm record. A disk drive is more expensive than a cassette recorder, but it is faster, more reliable, and more convenient to use.
3. Printer. Printers are relatively expensive, but they give you output on paper (so-called *hard copy*) that you can save for later use.

## **COMPUTER**

First, let us discuss the computer. The power connection is on the right side near the back. You should connect the Commodore 64 to a wall socket, using the power-supply unit. Just in front of the power connection is a small on/off switch. You can turn the computer off by pushing down on this switch, disconnecting the power supply, or by pulling the power supply's plug out of the socket. You shouldn't do either of these very often, but pulling the plug out of the wall socket is the more reliable approach. Unfortunately, the computer's power light is so dim in normal room lighting that you usually cannot tell if it is on or off.

The keys on the front allow you to enter things into the computer. The only fundamental difference from a typewriter is that the entries appear on the screen rather than on paper.

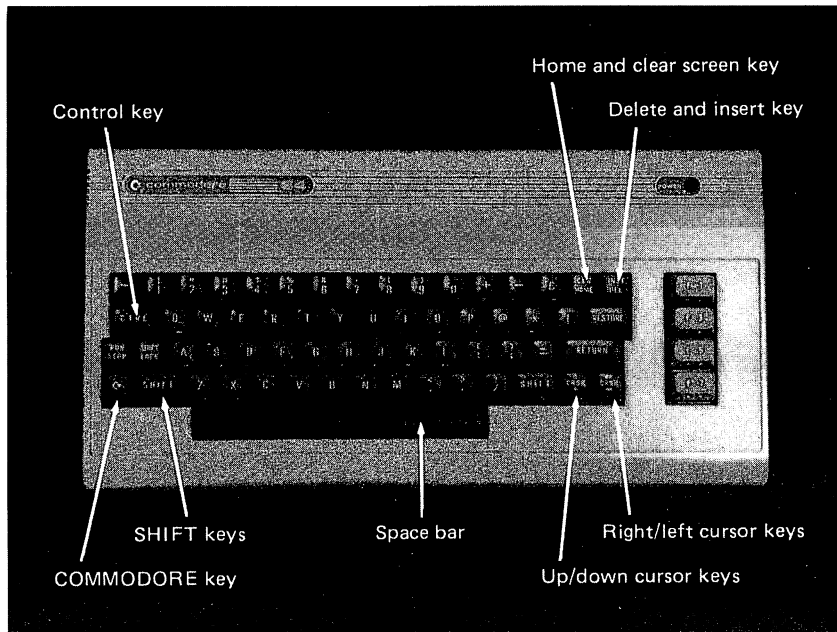
That is all you can see of the computer. The electrical parts are inside the case and are organized much like the nervous system of a person. They are

1. A brain, generally called the *central processing unit* or *CPU*.
2. Memory. Its only special feature is that it forgets everything when you turn the computer off. That is why you don't want to turn the power off very often. When you quit for the day, you should save anything you may want to use later on a cassette or diskette.
3. Connections to the outside world, generally called *interfaces*. These are like the body's nerves and muscles.

## **TELEVISION DISPLAY**

The television set serves as the computer's output device. The screen can hold 25 lines of 40 characters (a character is any letter, number, or symbol). We can think of the screen as a grid (see Fig. 1-2) consisting of 25 lines or rows and 40 columns. We will refer to the rows as 1 through 25, starting with 1 at the top, and to the columns as 1 through 40, starting with 1 at the top left. You will find it handy later to have copies of the grid in Fig. 1-2, since we will use it to draw pictures and to position the characters.






**Figure 1-3** Commodore 64 Keyboard. (Courtesy of Commodore Electronics Limited.)

## KEYBOARD

The keyboard (see Fig. 1-3) resembles a portable typewriter. As on a typewriter, you can use the SHIFT keys (one on each side in the row above the space bar) to type capital letters and uppercase symbols. You can even press SHIFT LOCK to produce these symbols without pressing SHIFT each time. However, you can see right away that this is not a normal typewriter keyboard. Among its special features are:

1. Extra keys such as CTRL (leftmost column), INST/DEL (top right-hand corner, CLR/HOME (just left of INST/DEL), 2 CRSR keys with arrows on them (bottom right), and  (the COMMODORE key, bottom left).
2. Two strange markings on the front of many keys, including all letter keys and some symbol keys. Most of these markings look like hieroglyphics, mystic insignia, or parts of cattle brands. Only a few, such as the playing-card-suit symbols on the A, S, Z, and X keys, are immediately recognizable.
3. Abbreviations on the front of the number keys in the top row. RED on the front of the 3 key is a sure clue to what these mean; you can probably guess that BLK is black, WHT white, PUR purple, GRN green, BLU blue, and YEL yellow. In case you are not an artist, CYN is cyan, or greenish-blue.

Moving right in the top row, we have RVS ON (on the front of the 9 key) and RVS OFF (on the front of the 0 key). RVS means “reverse,” which tells the computer to exchange its typing and background colors (thus producing dark blue on a light blue background, for example, instead of the normal light blue on dark blue). What does all this mean? Well, let us describe it little by little. We will not tell you everything at once—just the high points. You can learn the rest later as you do things.

The most important features of the Commodore 64 keyboard are the following:

1. When you turn the power on, the computer starts in the graphics mode. In this mode, pressing letter keys produces capital letters, while pressing nonletter keys produces lowercase symbols (such as /, :, and 3). To enter a strange-looking symbol on the front of the keys, you must press the key together with either SHIFT or the COMMODORE key, much as if you were entering a capital letter on a typewriter. Pressing COMMODORE gives you the left-hand symbol (we call these *left-hand graphics*), while pressing SHIFT gives you the right-hand symbol (*right-hand graphics*). Note that you cannot type lowercase letters when the computer is in the graphics mode.

The name *graphics* should tell you that the odd-looking symbols are building blocks for pictures, charts, lines, graphs, and forms. You may compare them to the pieces in a child’s building set or to the standard shapes and sizes of lumber, metal, pipe, and wire used in construction.

2. To enter lowercase letters, you must put the computer in the text mode by pressing SHIFT and COMMODORE simultaneously. Now the keyboard works like a normal typewriter. Pressing a key produces the lowercase letter or symbol, while pressing SHIFT at the same time produces the capital letter or uppercase symbol. Pressing COMMODORE and a key simultaneously produces left-hand graphics. In the text mode, you cannot type right-hand graphics; we have essentially traded them for lowercase letters. To return to the graphics mode, simply press SHIFT and COMMODORE simultaneously again.

You may compare switching keyboard modes to switching a radio between the AM and FM bands. To tune in a station, you must have the radio in the right band as well as at the right frequency.

Press some keys, including letters, numbers, and other symbols. Now press SHIFT and COMMODORE together. Some characters change instantaneously. Lowercase letters become capital letters and capital letters become right-hand graphics, or vice versa. Numbers, symbols with no graphics on their keys (e.g., :, =, or /), and left-hand graphics do not change. The effect is as if someone had suddenly put a prism or mirror between you and the screen.

Press SHIFT and COMMODORE together again. The characters change back to their original form.

Note that the Commodore 64 has no mode indicator. Of course, you can always tell which mode it is in by pressing a letter key with SHIFT LOCK up (inactive). If you get a lowercase letter, the computer is in text mode; if you get an uppercase letter, the computer is in graphics mode.

3. You can change the cursors’s color by holding CTRL down and pressing a color key (1 through 8). Pressing CTRL and BLU (7) makes the cursor disappear, since it is now dark blue against a dark blue background. When you type after changing the cursor’s color, characters appear in the new color. Of course, entering dark blue characters is like



writing in invisible ink. You should also note that black and purple characters are difficult to read. In fact, you may want to change the printing color to white (2), cyan (4), or yellow (8) to get better contrast than the normal light blue produces. To return to light blue, press STOP/RUN and RESTORE (just above RETURN) simultaneously.

4. Holding CTRL down and pressing RVS ON (9) reverses the character and screen (or background) colors. Now, when you type, characters appear in reverse—that is, dark blue against the color you have chosen for the cursor. They look like X rays or photographic negatives. Pressing RVS OFF (0) or RETURN restores the normal situation. Remember to hold CTRL down while pressing color keys or RVS ON or OFF.

Experiment with the keys for a while. Try pressing various combinations. What happens when you press SHIFT and a key that has no uppercase symbol? Put the computer in text mode and press SHIFT and @, the up-arrow key, or the British pound sign. The results are symbols that are not even on the keyboard.

That is enough for now. We have introduced the COMMODORE, SHIFT, color, RVS ON and OFF, CTRL, and SHIFT LOCK keys. We will now digest them slowly (the brown coloring is quite tasty). One thing you should remember is that you can always push SHIFT LOCK down (it catches) to enter directly whatever SHIFT would produce normally (that is, capital letters in the text mode, right-hand graphics in the graphics mode).

## **TALKING TO A COMPUTER**

Now that we have described the computer a little, we are ready to tell it to do something. But how do you talk to a computer? Does it understand French, Spanish, or Swahili? No, none of these. Computers have their own languages, with names like BASIC, COBOL, FORTRAN, and Pascal. We will use BASIC, the most popular language for small computers. We will briefly introduce BASIC here and then tell you more about it as we go along, rather than describing it completely right away. After all, no one tells you all about English in the first grade.

## **A FIRST LOOK AT BASIC**

BASIC lets us enter instructions or statements into the computer's memory. A set of instructions (a *program*) forms a procedure that the computer can follow. You may compare programs to recipes, orders of the day, or kit assembly instructions.

Two essential features of BASIC are:

1. Each line (*statement*) must start with a number. We usually choose these line numbers at least 10 apart (e.g., 100, 110, 120, 130) to permit later insertions (e.g., 105 or 136). We end a line (as far as the computer is concerned) by pressing the RETURN key. This puts the line in the computer's memory. RETURN acts like the carriage return on a typewriter, although there is, of course, no physical carriage to move.

2. The computer does what the lines say in numerical order. That is, it starts with the lowest-numbered line and proceeds upward, unless specifically told to do otherwise. You can enter the lines in any order; the computer arranges them in numerical order automatically.

Besides statements, BASIC also has commands that tell the computer what to do with the instructions; these do not have numbers. The most common commands are:

NEW Clear the computer (i.e., start with a clean slate).

LIST Display the current program, placing the lines in numerical order.

RUN Do what the program says, starting with the lowest-numbered statement and continuing in numerical order.

Thus, to enter a BASIC program and have the computer do what it says, you must:

1. Type NEW to clear the computer. Be sure you really want to do this before you press RETURN, since any work you have in the computer will be lost.
2. Enter the statements with their line numbers in any order.
3. Type RUN to have the computer execute the program. Remember to end each statement or command by pressing the RETURN key.

You may type LIST at any time to get an ordered listing of the program. To slow the listing down, press CTRL.

## **TYPING ERRORS**

What if you make a typing mistake? After all, no one's perfect. Three keys will help you correct mistakes:

INST/DEL (insert/delete, top right-hand corner) lets you insert or delete a character. To insert one (since INST is uppercase), you must press SHIFT and INST/DEL simultaneously and then type the character. To delete, just press INST/DEL, but be sure SHIFT LOCK is up.

CRSR with arrows pointing left and right (lower right-hand corner) moves the cursor left or right. To move it left (since the left arrow is on top) either press SHIFT at the same time or push SHIFT LOCK down.

CRSR with arrows pointing up and down (just left of the other CRSR key) moves the cursor up or down. To move it up, you must either press SHIFT at the same time, or push SHIFT LOCK down.

INST/DEL and the CRSR keys repeat. That is, holding them down is equivalent to pressing them repeatedly. Be careful, though; the repeat feature is fast and can easily move the cursor farther than you wanted or erase characters you intended to keep.

The procedure for correcting entries is as follows:

To correct a single wrong character (e.g., you typed PRUNT instead of PRINT), move the cursor to it and type the correct character. The new character replaces the old one; you do not have to delete the old one first as you do on a correcting typewriter.

To delete an extra character (e.g., as in PRINMT), move the cursor to the character just ahead (right) of it and press INST/DEL. Be careful, INST/DEL erases the character just left of the cursor, not the one underneath it. Note that the characters to the right move left automatically to fill the gap.

To insert a character, move the cursor to where you want the insertion. For example, if you typed PRNT instead of PRINT, move the cursor over the N, then press SHIFT and INST/DEL simultaneously. This opens up a space in which you can now type I. All characters to the right move right one space. Note that INST/DEL adds a character directly underneath the cursor. Again, be careful; if you want to insert a space, you must press the space bar. The space that seems to be there already is like an optical illusion.

If you have already entered a line by pressing RETURN, simply type it again with the same number. The computer erases the old line automatically.

But what if you don't see the mistake and you press RETURN or type RUN? If the error makes no sense to the computer, it will tell you that something is wrong by printing the message

```
PSYNTAX ERROR
```

This means you typed something the computer did not understand (such as PRUNT instead of PRINT). In most simple cases, the mistake is obvious and you can reenter the line and start over. To erase an old line completely, enter its number and press RETURN.

## **HELPFUL HINTS**

Until you get used to it, the Commodore 64 is confusing. In the first place, there is no way of telling which mode it is in unless you type something. Then you must delete what you typed if the mode is wrong. Also, you have to check whether SHIFT LOCK is down. You must remember the meaning of the CTRL, SHIFT, and COMMODORE keys. Can you recall which key produces left-hand graphics and which right-hand graphics? Don't worry; it may be enough to make your head swim now, but you will soon get used to the Commodore 64. You will be amazed at what you can create; it just takes a little time.

As with most tasks, there are a few things you should remember. We therefore conclude this chapter with a list of those—refer to it whenever you aren't quite sure what to do next. In a short time, these things will become second nature.

1. To stop the computer when you are not sure what it is doing, press the RUN/STOP key. Be sure SHIFT LOCK is up. If this does not do the job, try pressing RUN/STOP and RESTORE simultaneously.

2. To stop the computer and start over from scratch, turn it off and then back on again. Do this only as a last resort, never while playing a disk or cassette.
3. To type the upper symbol on a key with two symbols, press the key while holding SHIFT down (or press it with SHIFT LOCK down). The computer's mode does not matter.
4. To enter a character on the right front of a key (that is, right-hand graphics), press the key while holding SHIFT down. This works only in the graphics mode.
5. To enter a character on the left front of a key (that is, left-hand graphics), press the key while holding the COMMODORE key down. This works in either graphics or text mode.
6. To change modes, press SHIFT and COMMODORE simultaneously.
7. To determine which mode the computer is in, press a letter key (with SHIFT LOCK up). If you get a capital letter, the computer is in graphics mode; if you get a lowercase letter, the computer is in text mode.
8. To enter lowercase letters, you must put the computer in text mode. You can enter capital letters and left-hand graphics in either mode, but remember to press SHIFT or push SHIFT LOCK down to type capital letters in text mode.
9. To correct a typing error, move the cursor to it and use the INST/DEL key if you must insert or delete something. To change a character, simply move the cursor to it and type the correct character. If you have already entered the line into the computer, just retype it.
10. To display the current program in numerical order, enter LIST. To slow the listing down, press CTRL.
11. To clear out old programs and start over, enter NEW.
12. To change the cursor and character color, press CTRL and a color key. The color keys are the digits 1 through 8; each key has a color name on its front. Remember, however, that dark blue on dark blue is invisible, and black on dark blue is difficult to read.
13. To type characters in reverse colors, press CTRL and RVS ON (the 9 key). To restore the normal arrangement, press CTRL and RVS OFF (the 0 key) or RETURN.

# 2

## SIMPLE PRINTING

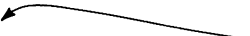
Before starting this chapter, be sure your computer is in the graphics mode. To do this, check that **SHIFT LOCK** is up (release it if it isn't) and press a letter key (say, **A**). If a capital letter appears on the screen, the computer is in graphics mode and you may proceed. If a lowercase letter appears, the computer is in text mode. To put it in graphics mode, press **SHIFT** and **COMMODORE** simultaneously. You may also want to change the printing color to cyan (**CYN**) for better visibility.

### **PRINT STATEMENT**

**PRINT** tells the computer to print something on the screen. The sequence

```
100 PRINT "DOG"  
RUN
```

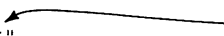
To type **"**, hold **SHIFT** down and press the **2** key.



prints the word **DOG**. We can put anything between the quotation marks. For example,

```
100 PRINT "*-HELLO-*"  
RUN
```

**\*** is right of **P** and **-** is right of **O**.



prints the message **\*-HELLO-\***

When you have the computer in text mode, be careful. The Commodore 64 will only accept statements like **PRINT** and commands like **RUN** in capital letters (e.g.,

PRINT, RUN) in graphics mode. In text mode, it will only accept lowercase (e.g., print, run). It will not accept any capital letters (e.g., PRINT, RUN, or even Print or Run) in text mode.

To see how PRINT works, enter Program 2-1.

### Program 2-1 Print Animal Names

```

NEW
100 PRINT "DOG"
110 PRINT "CAT"
120 PRINT "BIRD"
RUN
  
```

Clears old programs and starts with a clean slate. The computer responds by skipping a line and printing READY (or ready in the text mode).

Press the RETURN key after each line. Watch that you do not accidentally press RESTORE (the key directly above RETURN).

Tells the computer to perform the instructions in numerical order.

The screen should show

```

DOG
CAT
BIRD
  
```

Whatever you put inside quotation marks appears on the screen.

To print several things on a line, separate them with commas or semicolons. Comma space things out, while semicolons draw them together. For example, type

```

NEW
100 PRINT "DOG", "CAT", "BIRD", "FISH"
RUN
  
```

Note that , (comma) is a lowercase character only.

The screen should look like

```

DOG   CAT   BIRD   FISH
  
```

Note the wide spacing between words. To bring them together, replace the commas with semicolons.

```

NEW
100 PRINT "DOG"; "CAT"; "BIRD"; "FISH"
RUN
  
```

; (semicolon) is right of L, but one key farther than on a normal typewriter.

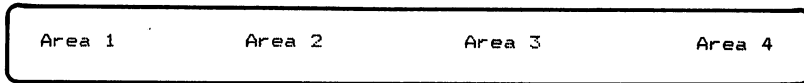
The screen should look like

```

DOGCATBIRDFISH
  
```

The words are crowded together. Of course, you could separate them by putting spaces inside the quotation marks.

A comma makes the computer move to the next area of the screen. There are four areas, each of which can hold 10 characters.



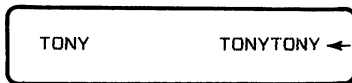
Semicolons make the computer crowd things together with no spaces in between. Now try

```
NEW
100 PRINT "TONY", "TONY"; "TONY"
RUN
```

Put your name here.

Be sure to type a quotation mark before and after your name each time.

The screen should look like



Your name will be here, but the second and third repetitions will start in the third area if your name is Rumpelstiltskin, Aristophanes, or anything else with 10 or more characters.

## REPEATING

So far, we have asked the computer to print each line only once. The FOR NEXT statement tells the computer to repeat a line. Obviously, it must tell the computer what to repeat and how many times. Therefore, FOR-NEXT has two parts:

1. FOR tells the computer how many times to repeat instructions. The usual form is FOR J=1 TO N, where N is the number of repetitions. We call J the *loop variable* or *index*.
2. NEXT (further along in the program) tells the computer how far (from FOR) to repeat. The usual form is NEXT J, where J is the loop variable in FOR.

For example, look at the following program. Compare how the computer moves through it to how a person bounces on a trampoline.

### Program 2-2 Print DOG 20 Times

```
NEW
100 FOR J=1 TO 20
110 PRINT "DOG"
120 NEXT J
RUN
```



= is just left of RETURN. Note that it is both uppercase and lowercase, so you do not have to press SHIFT to type it.

Repeat everything 20 times.

Continue this far and bounce back.

Program 2-2 tells the computer to do everything between FOR J (line 100) and NEXT J (line 120) 20 times. The result is that the computer prints DOG 20 times. Note that the old DOGs move rapidly up the screen as the computer prints new ones. Try changing line 100 to

```
100 FOR J=1 TO 50
```

Now DOGs fill the screen and disappear at the top.

NEXT acts like a trampoline that bounces the computer back up to FOR. Every time the computer reaches NEXT J, it bounces back up to FOR J=1 TO 20. Programmers call this moving down and bouncing back a *loop*. After the twentieth repetition, the computer falls through NEXT J to the line below. Now try printing your name 10 times.

### Program 2-3 Printing Your Name 10 Times

```
NEW
100 FOR J=1 TO 10
110 PRINT "TONY"
120 NEXT J
RUN
```

The screen should look like

```
TONY
TONY
TONY
TONY
TONY
TONY
TONY
TONY
TONY
TONY
```

To print your name 30 or 50 times, simply change the number after TO in line 100 to 30 or 50. Now try filling the screen with your name. First, use a comma.

### Program 2-4 Your Name in Columns

```
NEW
100 FOR J=1 TO 40
110 PRINT "TONY",
120 NEXT J
RUN
```

The screen should quickly show your name printed 40 times, four times per line. Remember that the computer divides the screen into four areas, and a comma sends it to the next area.



```

TONY    TONY    TONY    TONY
TONY    TONY    TONY    TONY
TONY    TONY    TONY    TONY
TONY    TONY    TONY    TONY
TONY    TONY    TONY    TONY
TONY    TONY    TONY    TONY
TONY    TONY    TONY    TONY
TONY    TONY    TONY    TONY
TONY    TONY    TONY    TONY
TONY    TONY    TONY    TONY

```

Now try a semicolon.

**Program 2-5 Your Name Filling Screen**

```

NEW
100 FOR J=1 TO 50
110 PRINT "TONY";
120 NEXT J
RUN

```

Type a semicolon at the end.

Put your name here.

The screen should show your name printed 50 times like

```

TONYTONYTONYTONYTONYTONYTONYTONYTONYTONYTONYTONY
TONYTONYTONYTONYTONYTONYTONYTONYTONYTONYTONYTONY
TONYTONYTONYTONYTONYTONYTONYTONYTONYTONYTONYTONY
TONYTONYTONYTONYTONYTONYTONYTONYTONYTONYTONYTONY
TONYTONYTONYTONYTONYTONYTONYTONYTONYTONYTONYTONY

```

The effect is like a repeating pattern on a shirt, blanket, or piece of pottery. To produce a different effect, type \* or – before and after your name in PRINT. For example,

**Program 2-6 Your Name Fancy**

```

NEW
100 FOR J=1 TO 40
110 PRINT "-TONY-";
120 NEXT J
RUN

```

– is in the top row, right of zero (\* is in the second row, right of P).

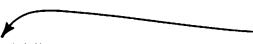
What do you see? The pattern will be skewed unless multiple copies of your name with hyphens around it fit on a line evenly.

## SAVING YOUR WORK ON A CASSETTE

As the programs become longer, you may want to save your work on a tape or a disk. To save a program on a tape, proceed as follows:

1. Rewind the tape.
2. Press the STOP/EJECT key on the recorder. Be sure no keys are down.
3. If you have a blank cassette, skip to step 6. Obviously it is much simpler (but more expensive) to use a separate cassette for each program. If you already have programs on the cassette, you must move the tape beyond them. To do this, enter the command

```
VERIFY "P12-01"
```



The name of the last program.  
This will change with each program  
you save.

We have chosen the standard names Pcc-*nn* where *cc* is the chapter number and *nn* is the program number. You can use whatever names you want, although you should settle on some standard that you will remember. Note that you must put quotation marks before and after the name. Mark the program's name and the current value of the tape counter in a notebook; this will help you find the program later.

4. When you see the message

```
PRESS PLAY ON TAPE
```

press the PLAY key on the recorder. The tape will start and the computer will search the cassette for P12-01 (or whatever name you entered). You must keep track of all programs stored on a cassette; in particular, you should mark the name of the last one on a label. The computer will display

```
SEARCHING FOR P12-01
```

on the screen. As it finds programs, it will report FOUND followed by their names.

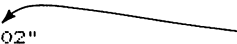
5. When you see the message

```
FOUND P12-01  
VERIFYING  
?VERIFYING ERROR
```

the tape is at the end of the last program. You are now ready to save the program currently in memory. Press the STOP/EJECT key on the recorder and make sure that no keys are down.

6. Enter

```
SAVE "P12-02"
```



This program's name.

7. When you see the message

PRESS RECORD AND PLAY ON TAPE

press the PLAY and RECORD keys simultaneously. The computer will now store your program on tape under the name P12-02.

8. When the computer is finished, rewind the tape.

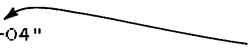
Do not use long cassettes (more than 30 minutes) or put too many programs on a cassette. It takes the recorder a long time (many seconds) to rewind or search a tape. The more programs you have on the tape, the longer it takes. Remember to reset the tape counter to 000 each time you rewind a tape.

## **LOADING A PROGRAM FROM A CASSETTE**

To load a program stored on tape,

1. Rewind the tape.
2. Press the STOP/EJECT key on the recorder. Be sure no keys are down.
3. Enter

LOAD "P13-04"

The name of the program you want.

4. You will see the message

PRESS PLAY ON THE TAPE

5. Press the PLAY key on the recorder. The computer now searches the tape for P13-04. When it finds the program, the message

LOADING P13-04

appears and the computer loads the program into memory. If it cannot find the program, the computer will read the entire tape. Stop the tape before it goes too far!

## **SAVING YOUR WORK ON A DISK**

To save a program on a disk, enter

SAVE "P02-07",8

or

SAVE "P02-06",8

The program names have the form Pcc-nn, where cc is the chapter number and nn is the program number. See the discussion under "Saving Your Work on a Cassette."

## **DETERMINING WHAT IS ON A DISK**

Type

```
LOAD "$",B
LIST
```

The computer will list all the programs on the disk. Keep these lists in a notebook for handy reference. Examining a list will keep you from repeating a name.

## **LOADING A PROGRAM FROM A DISK**

To load a program stored on disk, type LOAD as in

```
LOAD "P02-07",B
```

Remember that P02-07 must already be on the disk.

## **TAB STATEMENT**

To make the computer form patterns, we need a way to tell it where to start a line. The key here is the TAB statement, which works like the tab on a typewriter. For example, TAB(10) tells the computer to move right ten columns, (that is, to column 11, since the cursor starts in column 1). Similarly, TAB(C) tells the computer to move right C columns. TAB thus lets the computer start a message anywhere on a line.

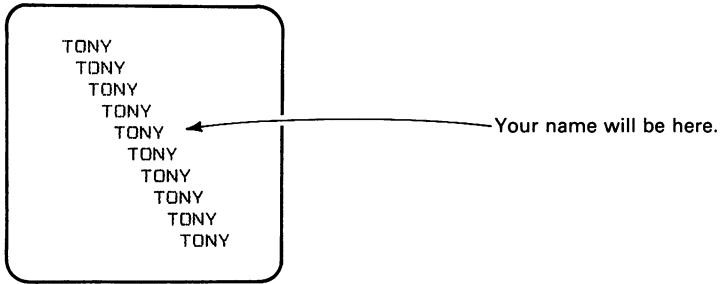
### **Program 2-7 Your Name Diagonal**

```
NEW
100 FOR C=1 TO 10
110 PRINT TAB(C)"TONY"
120 NEXT C
RUN
```

Remember to press SHIFT to enter ( and )!  
We want to print 10 times.  
Do not put a space here.  
Put your name here.

Starts in column 2 when C is 1.  
Starts in column 3 when C is 2.  
Starts in column 4 when C is 3.  
Etc.

Note that we use C (for column) as our loop variable in Program 2-7; the computer does not care what letter we use as long as we are consistent. The screen should look like



Each line starts one space farther right, since the C in TAB(C) is one larger.

To start the first line in column 11 and make subsequent lines move left one column, we use

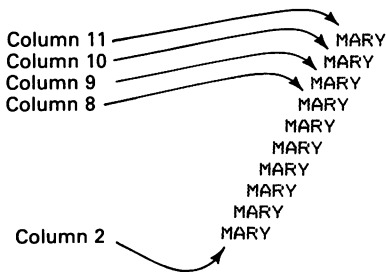
```
FOR C=10 TO 1 STEP -1
```

C starts at 10 and is reduced by 1 each time the computer reaches NEXT C. This continues until reducing C makes it less than 1. Note that we must add the word STEP and the step size, if the size is not 1. What happens if you make the lower limit 0 instead of 1? TAB(0) does not move the cursor at all.

**Program 2-8 Friend's Name Diagonal**

```
NEW
100 FOR C=10 TO 1 STEP -1
110 PRINT TAB(C) "MARY"
120 NEXT C
RUN
```

The screen shows



The name moves left instead of right. We can combine Programs 2-7 and 2-8 to print both names in a crossing pattern.

## Program 2-9 Two Names in Crossing Pattern

```
NEW  
100 FOR C=1 TO 15  
110 PRINT TAB(C) "TONY"  
120 PRINT TAB(16-C) "MARY"  
130 NEXT C  
RUN
```

Put your name here.

Put your friend's name here.

The screen shows

```
TONY  
TONY MARY  
TONY MARY  
TONY MARY  
TONY MARY  
TONY MARY  
TONY MARY  
TONY MARY  
TONY MARY  
TONY MARY  
MARY TONY  
MARY TONY  
MARY TONY  
MARY TONY  
MARY TONY  
MARY TONY  
MARY TONY  
MARY TONY  
MARY TONY  
MARY TONY
```

Part of the pattern, however, disappears off the top of the screen as the computer adds lines at the bottom.

Your name first starts in column 2, then in columns 3, 4, 5, 6, etc. At the same time, your friend's name first starts in column 16, then in 15, 14, 13, etc. Your friend's name moves left because

when C is 1,  $16-C$  is 15

when C is 2,  $16-C$  is 14

when C is 3,  $16-C$  is 13

.

.

.

when C is 15,  $16-C$  is 1

Since the computer prints your name after  $\text{TAB}(C)$  and your friend's name after  $\text{TAB}(16-C)$ , we get a crossing pattern.

# 3

## DRAWING PICTURES

We will now draw objects and make them move. To begin, we need a way to clear the screen. From the keyboard, this is easy. You must press the SHIFT and CLR/HOME keys together; CLR/HOME is in the top row, just left of INST/DEL. This not only clears the screen, but it also moves the cursor to the top left-hand corner. To clear the screen from within a program, we must tell the computer to do the same thing it does when we press SHIFT and CLR/HOME. The statement we need is PRINT CHR\$(147). This probably seems mysterious at the moment, but it is just the Commodore's version of "CLEAR SCREEN." Program 3-1 draws a face and moves it, using TAB as in Programs 2-8 and 2-9.

### **Program 3-1 Face Moving Right**

```
NEW
100 PRINT CHR$(147)
110 FOR C=1 TO 30
120 PRINT CHR$(147)
130 PRINT
140 PRINT TAB(C) "----"
150 PRINT TAB(C) "( O O )"
160 PRINT TAB(C) "( X )"
170 PRINT TAB(C) "----"
180 NEXT C
RUN
```

Remember to hold down the SHIFT key to type ", (, ), and !. If you use SHIFT LOCK, remember to release it before starting the next line.

Be sure to put quotation marks on both sides of the face.

Be sure to press RETURN after each line.

The face travels rapidly across the screen from left to right and stops. Now type RUN again. Note that the face blinks as it moves. That is because line 120 clears the screen before each drawing. You may want to enlarge the face or add features like a smile, a frown, a beard, or ears. This is a simple matter of experimenting with the PRINT statements until the picture looks the way you want it.



Pictures require a lot of typing. There are, however, two simple ways to keep the amount within reason:

1. Use ? instead of PRINT. The Commodore accepts either. The problem with ? is that it looks strange in listings, so we will not use it in ours.
2. Use the cursor-moving (CRSR) keys to obtain one line from another. Note, for example, in Program 3-1 that lines 140 and 170 are the same except for the line numbers. You can save a lot of typing by deriving line 170 from line 140 as follows:
  - a) Use the CRSR keys to move the cursor to the 4 in 140.
  - b) Type 7. This replaces the 4 automatically.
  - c) Press RETURN.

Line 170 is now in the computer's memory along with line 140. Of course, line 140 is no longer on the screen; you must enter LIST to get it back. Furthermore, if you entered lines after 140, you must move the cursor back to where it was before you started editing.

Line 130 (PRINT with nothing after it) simply produces a blank line. Inserting more lines like 130 (131 PRINT, 132 PRINT, etc.) will move the face farther down. If you do not want the face to stop at the far right, make the computer clear the screen one last time before exiting. You can do this by adding

```
190 PRINT CHR$(147)
```

to Program 3-1.

The face moves and flickers at a dizzying rate. We can slow it down by having the computer pause between drawings. To do this, we introduce a tight loop in which nothing happens. The statement

```
FOR K=1 TO 400:NEXT K
```

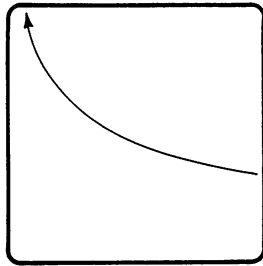
is such a loop. It works like telling a person to count up to 400 before opening his or her eyes. The colon lets us put NEXT K on the same line as FOR K=1 TO 400, thus making it clear that there is nothing between FOR K and NEXT K. The computer therefore simply bounces back and forth 400 times, doing nothing.

Insert the line

```
175 FOR K=1 TO 400:NEXT K
```

into Program 3-1 and run it. The face will travel slowly with a jerky motion. To speed it up, change 400 in line 175 to 300. Experiment with this number until the motion looks reasonable (50 to 150 is a good range).

To keep the picture from blinking, replace 120 PRINT CHR\$(147) with 120 PRINT CHR\$(19). PRINT CHR\$(19) moves the cursor to the upper left-hand corner of the screen without erasing anything. This is equivalent to pressing CLR/HOME without SHIFT (we call this *homing* the cursor).



PRINT CHR\$(19) puts the cursor here.

### Program 3-2 Creature Moving Right

```

NEW
100 PRINT CHR$(147)
110 FOR C=1 TO 30
120 PRINT CHR$(19)
130 PRINT
140 PRINT TAB(C) " (---) "
150 PRINT TAB(C) " ( O O ) "
160 PRINT TAB(C) " ( X ) "
170 PRINT TAB(C) " (---) "
175 FOR K=1 TO 100:NEXT K
180 NEXT C
RUN

```

Put spaces here.

Be sure to use the dash or hyphen in the top row.

Slows the computer down.

Now the face does not blink.

We can easily produce two moving faces instead of just one.

### Program 3-3 Two Faces Moving Right

```

NEW
100 PRINT CHR$(147)
110 FOR C=1 TO 20
120 PRINT CHR$(19)
130 PRINT TAB(C) " !--! !--! "
140 PRINT TAB(C) " (OO) (OO) "
150 PRINT TAB(C) " !==! !==! "
160 FOR K=1 TO 200:NEXT K
170 NEXT C
RUN

```

The exclamation mark is above the 1 key.

Slows the computer down.

Adding

```
180 PRINT CHR$(147)
```

makes the faces disappear rather than stop at the far right. Both Programs 3-2 and 3-3 move the faces. To make two faces move left, change line 110 of Program 3-3 to

```
110 FOR C=20 TO 1 STEP -1
```

This change makes the computer start C at 20 and count it down to 1. TAB(C) makes the printing start first in column 21, then in 20, then in 19, etc., until it reaches column 2. We now draw another face and move it left.

### Program 3-4 Monster Face Moving Left

```
NEW
100 PRINT CHR$(147)
110 FOR C=20 TO 1 STEP -1
120 PRINT CHR$(19)
130 PRINT TAB(C) " !====!"
140 PRINT TAB(C) " ! @ @ !"
150 PRINT TAB(C) " ! X !"
160 PRINT TAB(C) " {VVVVV} "
170 FOR K=1 TO 200:NEXT K
180 NEXT C
190 PRINT CHR$(147)
RUN
```

Annotations:

- 5 = (equal) signs.
- Be sure to leave a space on each side.
- @ is just right of the P key.
- Slows the computer down.

Another editing hint: If you have so many mistakes on a line that it is not worth correcting, simply press SHIFT and RETURN simultaneously. This will move the cursor down but will not enter the line into memory.

So far we have drawn small faces. We will now draw a large gorilla.

### Program 3-5 Gorilla Moving Left

```
NEW
100 PRINT CHR$(147)
110 FOR C=20 TO 1 STEP -1
120 PRINT CHR$(19)
130 PRINT TAB(C) " -- ---- - "
140 PRINT TAB(C) " !!---!00!---!"
150 PRINT TAB(C) " ! ! ! ! "
160 PRINT TAB(C) " !--- ----!"
170 PRINT TAB(C) " ! ! ! ! "
180 PRINT TAB(C) " !---!---!"
190 PRINT TAB(C) " ! ! ! ! "
200 PRINT TAB(C) " !---!---!"
210 FOR K=1 TO 100:NEXT K
220 NEXT C
230 PRINT CHR$(147)
RUN
```

Annotations:

- 3 spaces
- Put one space on each side.
- Slows the computer down.

The gorilla moves left awkwardly and a bit out of focus. Obviously, a large picture requires much more typing than a small picture. Furthermore, it is much harder to place things correctly. We have indicated our choice of spacing where it is difficult to judge. Fortunately, the spacing for most lines is easy to judge from the line above. We have also kept a consistent right margin, with at least one space at the end of each line. Of course, you do not have to type the extra spaces on lines such as 170 that do not extend to the boundary. The only advantage of our approach is that it makes the listing look neater.

One problem is that more lines lead to typing errors. Errors are easy to correct, but the picture often ends up in fragments on the screen. It looks like a jigsaw puzzle before final assembly. If this happens, press SHIFT and CLR/HOME together to clear the screen. Then enter LIST to get a current listing of the program before you RUN it.

Now let's draw some other pictures; feel free to use your imagination to improve or change them.

### Program 3-6 Car Moving Left

```
NEW
100 PRINT CHR$(147)
110 FOR C=20 TO 1 STEP -1
120 PRINT CHR$(19)
130 PRINT TAB(C) "  !!  "
140 PRINT TAB(C) " /-----< "
150 PRINT TAB(C) " !-0--0-< "
160 FOR K=1 TO 200:NEXT K
170 NEXT C
180 PRINT CHR$(147)
RUN
```

4 spaces  
3 spaces  
6 dashes. < is the uppercase character above the comma.

### Program 3-7 Airplane Moving Right

```
NEW
100 PRINT CHR$(147)
110 FOR C=1 TO 20
120 PRINT CHR$(19)
130 PRINT TAB(C) "      !! "
140 PRINT TAB(C) "    !! !! "
150 PRINT TAB(C) " 0----->"
160 PRINT TAB(C) " 0----->"
170 PRINT TAB(C) "    !! !! "
180 PRINT TAB(C) "      !! "
190 FOR K=1 TO 200:NEXT K
200 NEXT C
210 PRINT CHR$(147)
RUN
```

6 spaces  
6 dashes. > is the uppercase character above the period.  
2 spaces

Note that the airplane is symmetric, so lines 150 and 160 are the same, as are lines 140 and 170 and lines 130 and 180.

### Program 3-8 Sailboat Moving Left

```
NEW
100 PRINT CHR$(147)
110 FOR C=20 TO 1 STEP -1
120 PRINT CHR$(19)
130 PRINT TAB(C) " (+++)"
140 PRINT TAB(C) "  !*  "
150 PRINT TAB(C) "  ! * "
160 PRINT TAB(C) "  ! * "
170 PRINT TAB(C) " <-----"
180 PRINT TAB(C) " <-----"
190 PRINT TAB(C) " *****"
200 FOR K=1 TO 200:NEXT K
210 NEXT C
220 PRINT CHR$(147)
RUN
```

2 spaces  
Leave at least one space at the end of each line. Lines 170 and 180 are identical.  
8 asterisks

### Program 3-9 Tank Moving Left

```
NEW
100 PRINT CHR$(147)
110 FOR C=20 TO 1 STEP -1
120 PRINT CHR$(19)
130 PRINT TAB(C) " <---- "
140 PRINT TAB(C) " ===<----! "
150 PRINT TAB(C) " (ODD) "
160 FOR K=1 TO 200:NEXT K
170 NEXT C
180 PRINT CHR$(147)
RUN
```

4 spaces

Leave a space here.

So far our pictures have moved only one way, either right or left. To make them move back and forth, we must combine a program that moves a picture left with one that moves the same picture right.

### Program 3-10 Alien Face Moving Back and Forth

```
NEW
100 PRINT CHR$(147)
110 FOR C=1 TO 30
120 PRINT CHR$(19)
130 PRINT TAB(C) " !--! "
140 PRINT TAB(C) " (DD) "
150 PRINT TAB(C) " !==! "
160 FOR K=1 TO 100:NEXT K
170 NEXT C
200 PRINT CHR$(147)
210 FOR C=30 TO 1 STEP -1
220 PRINT CHR$(19)
230 PRINT TAB(C) " !--! "
240 PRINT TAB(C) " (DD) "
250 PRINT TAB(C) " !==! "
260 FOR K=1 TO 100:NEXT K
270 NEXT C
280 PRINT CHR$(147)
RUN
```

Leave spaces here.

One FOR-NEXT loop goes from line 110 to line 170.

Another FOR-NEXT loop goes from line 210 to line 270.

Note how we chose the line numbers in Program 3-10 to simplify entry. After we enter lines 100 through 170, we can move the cursor up to the 1 in line 100. Now press the 2 key, followed by RETURN. The screen displays 200 PRINT CHR\$(147) and the cursor moves to the 1 in 110 FOR C=1 TO 30. Press 2 again, followed by RETURN. This shortcut saves us from having to retype virtually identical lines. Keep entering 2s followed by RETURNs until you have converted lines 120 through 170 into lines 220 through 270. Enter LIST to check the instructions, enter line 210 correctly, and add line 280 to complete the program.

It takes practice to master this shortcut. Try it on some old programs, but do not save them afterward. Once you master it, this approach makes program entry much faster, especially when the same BASIC statements occur many times. Watch, however, for lines such as 210 that you must change. If you forget to change 210, the face will move right, disappear, and start at the left again. The two FOR-NEXT statements make the face move right and then back left. We can apply this approach to other pictures.

### Program 3-11 Bug Moving Back and Forth

```
NEW
100 PRINT CHR$(147)
110 FOR C=1 TO 20
120 PRINT CHR$(19)
130 PRINT TAB(C)" ***** "
140 PRINT TAB(C)" * @* "
150 PRINT TAB(C)" ***** "
160 PRINT TAB(C)" )))) "
170 FOR K=1 TO 200:NEXT K
180 NEXT C
200 PRINT CHR$(147)
210 FOR C=20 TO 1 STEP -1
220 PRINT CHR$(19)
230 PRINT TAB(C)" ***** "
240 PRINT TAB(C)" *@ * "
250 PRINT TAB(C)" ***** "
260 PRINT TAB(C)" ((((( "
270 FOR K=1 TO 200:NEXT K
280 NEXT C
290 PRINT CHR$(147)
RUN
```



One FOR-NEXT loop.



Another FOR-NEXT loop.

To enter Program 3-11 using the CRSR keys, proceed as follows:

1. Enter lines 100 through 180.
2. Move the cursor up to the 1 in line 100.
3. Enter 2 and press RETURN repeatedly to convert lines 100 through 180 into lines 200 through 280.
4. Enter lines 210, 240, and 260 correctly.
5. Enter line 290.
6. Enter LIST to see the entire program at once.

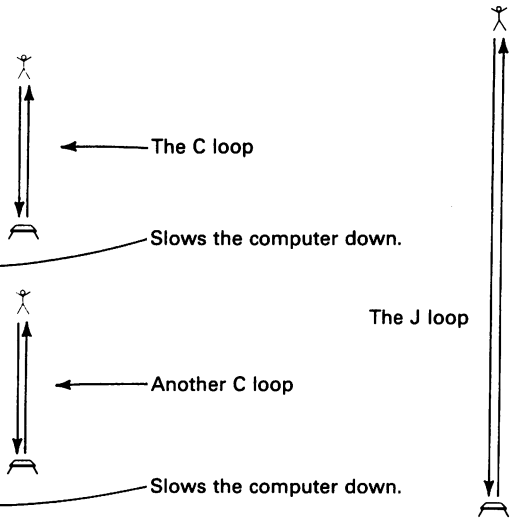
To make a picture move back and forth repeatedly, we could type RUN over and over. An alternative is a third FOR-NEXT loop.

### Program 3-12 Cow's Face Moving Back and Forth Three Times

```

NEW
 90 FOR J=1 TO 3
100 PRINT CHR$(147)
110 FOR C=1 TO 20
120 PRINT CHR$(19)
130 PRINT TAB(C) " (----) "
140 PRINT TAB(C) " !@ @! "
150 PRINT TAB(C) " ! ** ! "
160 PRINT TAB(C) " ! -- ! "
170 PRINT TAB(C) " ! _ ! "
180 PRINT TAB(C) " !-----! "
190 FOR K=1 TO 100:NEXT K
195 NEXT C
200 PRINT CHR$(147)
210 FOR C=20 TO 1 STEP -1
220 PRINT CHR$(19)
230 PRINT TAB(C) " (----) "
240 PRINT TAB(C) " !@ @! "
250 PRINT TAB(C) " ! ** ! "
260 PRINT TAB(C) " ! -- ! "
270 PRINT TAB(C) " ! _ ! "
280 PRINT TAB(C) " !-----! "
290 FOR K=1 TO 100:NEXT K
295 NEXT C
300 NEXT J
310 PRINT CHR$(147)
RUN

```



The C loops fit inside the big J loop. The J loop makes the computer repeat everything from line 90 to line 300 three times. This program takes a while to finish. If you see an error right away, you may want to stop the action and correct it immediately. To do this, press the RUN/STOP key. RUN/STOP is in the leftmost column just above the COMMODORE key and just left of SHIFT LOCK.

Similarly, we can draw a tank and move it back and forth.

### Program 3-13 Tank Moving Back and Forth Four Times

```
NEW
 90 FOR J=1 TO 4
100 PRINT CHR$(147)
105 REM MOVE TANK RIGHT
110 FOR C=1 TO 20
120 PRINT CHR$(19)
130 PRINT TAB(C)" (-----) "
140 PRINT TAB(C)" (      ) "
150 PRINT TAB(C)" (      ===== "
160 PRINT TAB(C)" (-----! "
170 PRINT TAB(C)" (00000) "
180 FOR K=1 TO 200:NEXT K
190 NEXT C
200 PRINT CHR$(147)
205 REM MOVE TANK LEFT
210 FOR C=20 TO 1 STEP -1
220 PRINT CHR$(19)
230 PRINT TAB(C)" (-----) "
240 PRINT TAB(C)" (      ) "
250 PRINT TAB(C)" ===== ) "
260 PRINT TAB(C)" !-----) "
270 PRINT TAB(C)" (00000) "
280 FOR K=1 TO 200:NEXT K
290 NEXT C
300 NEXT J
310 PRINT CHR$(147)
RUN
```

5 dashes

3 spaces

The tank does not turn around. It simply disappears and is replaced by a tank moving in the opposite direction. Draw other pictures and watch them move back and forth. Note that we have introduced a minor BASIC statement called REM (or REMARK). REM lines just explain to a reader what the program is doing; the computer ignores them. You may omit them to save typing when you enter programs.



# 4

## SOME THINGS MOVE WHILE OTHERS REMAIN STATIONARY

In Chapter 3 we made entire pictures move. To make some parts move while others remain stationary, we must put TABs before some printing but not all of it. We can then make bullets, rockets, cars, or bugs move against a stationary background. For example, let's first move a rocket right.

### **Program 4-1 Small Rocket Moving Right**

```
NEW
100 PRINT CHR$(147)
110 FOR C=1 TO 30
120 PRINT CHR$(19)
130 PRINT TAB(C) " !=>"
140 FOR K=1 TO 100:NEXT K
150 NEXT C
160 PRINT CHR$(147)
RUN
```

Put a space here.

Now let's make a stationary tank fire the rocket.

### **Program 4-2 Tank Firing Rocket**

```
NEW
100 PRINT CHR$(147)
110 FOR C=8 TO 30
120 PRINT CHR$(19)
130 PRINT "(----!"
140 PRINT "!ARMY!==" ; TAB(C) " !=>"
150 PRINT "(0000)"
160 FOR K=1 TO 100:NEXT K
170 NEXT C
180 PRINT CHR$(147)
RUN
```

4 dashes

The tank stands still while the rocket moves right. The key here is that TAB(C) in line 140 comes after the tank but before the rocket. Examine the line carefully; TAB(C) looks like part of the picture, but it is not inside quotation marks. The C loop starts at 8 and goes to 30. This makes the computer draw the rocket first, starting in column 9 (the end of the tank's gun, 8 columns right of column 1), then in 10, etc., until it reaches 30. We can determine the initial C value either by counting characters or by experimenting. If the printing goes over 40 columns, the Commodore 64 will automatically continue it on the next line. This looks strange but does not cause an error message.

A more elaborate scene is an airplane firing two rockets.

### Program 4-3 Airplane Firing Rockets

```

NEW
100 PRINT CHR$(147)
110 FOR C=7 TO 30
120 PRINT CHR$(19)
130 PRINT "  !! ";TAB(C) " !=>"
140 PRINT "  !!  !! "
150 PRINT "O----->"
160 PRINT "O----->"
170 PRINT "  !!  !! "
180 PRINT "    !! ";TAB(C) " !=>"
190 FOR K=1 TO 100:NEXT K
200 NEXT C
210 PRINT CHR$(147)
RUN

```

5 spaces  
 Lines 130 and 180, 140 and 170,  
 and 150 and 160 are identical.  
 Put a space here.

Slows the computer down.

Again we use TAB(C) " != > " to make the rockets move. Note that the upper rocket is always slightly ahead of the lower rocket. To fire a rocket from an artillery gun, we draw the gun and put TAB(C) " != > " at the end of the muzzle.

### Program 4-4 Artillery Gun Firing Rocket Right

```

NEW
100 PRINT CHR$(147)
110 FOR C=6 TO 30
120 PRINT CHR$(19)
130 PRINT " !-!"
140 PRINT " !-!===";TAB(C) " !=>"
150 PRINT "ODD"
160 FOR K=1 TO 100:NEXT K
170 NEXT C
180 PRINT CHR$(147)
RUN

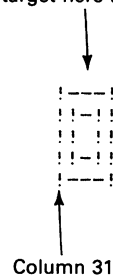
```

A different scene is an arrow shot at a target. Here the moving object travels toward the stationary object.

Start arrow here and move it.



Keep target here all the time.



### Program 4-5 Arrow Shot at Target

```

NEW
100 PRINT CHR$(147)
110 FOR C=0 TO 26
120 PRINT CHR$(19)
130 PRINT TAB(30)"!----!"
140 PRINT TAB(30)"!!-!!!"
150 PRINT TAB(C)" !=>";TAB(30)"!! !!"
160 PRINT TAB(30)"!!-!!!"
170 PRINT TAB(30)"!----!"
180 FOR K=1 TO 100:NEXT K
190 NEXT C
200 PRINT CHR$(147)
RUN

```

We want to move the arrow from column 1 to column 27. We cannot go beyond 27 since the arrow occupies 4 columns.

TAB(30) tells the computer to start printing in column 31 (30 right of column 1). TAB(C) tells the computer to start printing in column C+1, while FOR C=0 TO 26 changes C so the arrow appears to move. We must always put TAB(C) inside a loop.

Now let's drive a car through a town with three houses.

### Program 4-6 Car Driving Through Town

```

NEW
100 PRINT CHR$(147)
110 FOR C=0 TO 20
120 PRINT CHR$(19)
130 PRINT "/-----\ /-----\ /-----\"
140 PRINT "!-----!" !-----!" !-----!"
150 PRINT " ! | | | ! ! | | | ! ! | | | !"
160 PRINT " !-----!" !-----!" !-----!"
170 PRINT
180 PRINT TAB(C)" !=!"
190 PRINT TAB(C)" (----->"
200 PRINT TAB(C)" (-0-0->"
210 FOR K=1 TO 100:NEXT K
220 NEXT C
230 PRINT CHR$(147)
RUN

```

\ is shifted M.

Press COMMODE and + together.  
Press COMMODE and K together.  
3 spaces

Slows the computer down.

We used left-hand graphics (COMMODORE K and COMMODORE +) to put a door and window in each house. You can put extra spaces between the houses to enlarge the town.

We can make any object stand still while another moves. Suppose we want to draw a bumblebee flying toward a flower. To make the bee fly left, we use the statements

```
FOR C=30 TO 4 STEP -1
PRINT CHR$(19)
PRINT TAB(C)" **/ "
PRINT TAB(C)" (O/)" ← End each line of the bee with a space.
PRINT TAB(C)" /** "
FOR K=1 TO 200:NEXT K
NEXT C
```

### Program 4-7 Bee Flying Left to Flower

```
NEW
100 PRINT CHR$(147)
110 PRINT ← Move down 3 lines.
120 PRINT ← 4 spaces
130 PRINT ←
140 PRINT "  @@" ← @ is just right of P and left of *.
150 PRINT "  @ @@"
160 PRINT "   @ @"
170 PRINT "    @@"
180 PRINT " ** !! **"
190 PRINT " ***!***"
200 PRINT "-----" ← 9 dashes
210 FOR C=30 TO 4 STEP -1
220 PRINT CHR$(19)
230 PRINT TAB(C)" **/ " ← / is just left of the right-hand SHIFT key.
240 PRINT TAB(C)" (O/)" ← End each line of the bee with a space.
250 PRINT TAB(C)" /** "
260 FOR K=1 TO 200:NEXT K
270 NEXT C
280 PRINT CHR$(147)
RUN
```

Note how we place the figures. The computer first prints three blank lines (110 through 130) and then starts the flower on line 4. PRINT CHR\$(19) (line 220) ahead of the bee drawing makes the bee appear on lines 1 through 3.

We can easily change Program 4-7 to reverse the bee's direction. To place the flower at the right, we put TAB(28) ahead of each line in the drawing. To make the bee fly right, we use

```
FOR C=0 TO 29
PRINT CHR$(19) ← Put a space in front of each line
PRINT TAB(C)" \** " ← of the bee picture.
PRINT TAB(C)" (\O)"
PRINT TAB(C)" **\ " ← To type \, press SHIFT and M. \ is the
FOR K=1 TO 200:NEXT K ← right-hand graphics character on the
NEXT C ← front of the M key.
```

## Program 4-8 Bee Flying Right to Flower

```

NEW
100 PRINT CHR$(147)
110 PRINT
120 PRINT
130 PRINT
140 PRINT TAB(28) "   @@"
150 PRINT TAB(28) "  @@  @@"
160 PRINT TAB(28) "   @  @"
170 PRINT TAB(28) "    @@"
180 PRINT TAB(28) "  ** !! **"
190 PRINT TAB(28) "   **!!**"
200 PRINT TAB(28) "-----"
210 FOR C=0 TO 29
220 PRINT CHR$(19)
230 PRINT TAB(C) " \** "
240 PRINT TAB(C) " (\O)"
250 PRINT TAB(C) " **\ "
260 FOR K=1 TO 200:NEXT K
270 NEXT C
280 PRINT CHR$(147)
RUN

```

Move down 3 lines.  
 4 spaces  
 @ is just right of P and left of \*.  
 9 dashes  
 Begin each line of the bee with a space.  
 To type \ , press SHIFT and M.

You can insert TAB(28) into lines 140 through 200 of Program 4-7 by pressing SHIFT and INST/DEL seven times, then typing each character. Note that INST creates a space under the cursor, not to the right of it.

Let's now draw a ball and make it move back and forth. The ball looks like

```

*   line 1
* *  line 2
*   line 3

```

To make a ball move, the computer must draw it in one area, erase it, and draw it in a different area. Only one picture of the ball should be on the screen at a time. The computer must repeat the following steps:

1. Move the cursor to a location.
2. Draw a ball.
3. Erase the ball.

Let us first see what happens if we omit the erasure.

## Program 4-9 Moving Ball, Without Erasure

```

NEW
100 PRINT CHR$(147)
110 FOR C=1 TO 30
120 PRINT CHR$(19)
130 PRINT TAB(C) " * "
140 PRINT TAB(C) " * * "
150 PRINT TAB(C) " * "
160 FOR K=1 TO 100:NEXT K
170 NEXT C
180 PRINT CHR$(147)
RUN

```

Put 1 space on each side of the asterisk in line 130 and 150.

When we type RUN, the ball moves right, leaving a center line of asterisks behind it, somewhat like a moving snail. The reason for the residue is that parts of old drawings are still visible. You may want to sketch two successive drawings of the ball on a piece of paper to see which parts remain. One way to erase everything is to change line 120 to

```
120 PRINT CHR$(147)
```

in Program 4-9.

When you run the revised program, the trail behind the ball disappears. The problem here, as we saw earlier, is that CHR\$(147) erases the entire screen, not just the ball. PRINT CHR\$(147) would thus also erase the background, such as tennis players, a court, and a net. A better way to erase just the ball is simply to print spaces over it. That way only one ball appears on the screen at a time. In case you are wondering, this problem did not occur in many earlier programs because we put a space to the left of the pictures. That space then follows the picture as it moves right, erasing the leftmost column. To draw more than one picture, we must move the cursor to the top left-hand corner before starting each one.

### Program 4-10 Moving Ball, With Erasure

```
NEW
100 PRINT CHR$(147)
110 FOR C=1 TO 30
120 PRINT CHR$(19)
130 PRINT TAB(C) " * "
140 PRINT TAB(C) " * *"
150 PRINT TAB(C) " * "
160 FOR K=1 TO 100:NEXT K
170 PRINT CHR$(19)
180 PRINT TAB(C) "
190 PRINT TAB(C) "
200 PRINT TAB(C) "
210 NEXT C
220 PRINT CHR$(147)
RUN
```

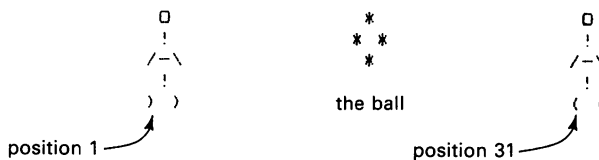
Put 1 space on each side of the asterisk  
in lines 130 and 150.

Put 3 spaces here.

To change the ball's speed, try different ending values in line 160.


We can draw two people and make the ball move between them as in a game of catch. One person must be on the left side and the other on the right. We can position the right-hand figure with TAB(30).

The picture looks like



## Program 4-11 Game of Catch

```
NEW
100 PRINT CHR$(147)
105 REM FORM LEFT-HAND FIGURE STARTING IN COLUMN 1
110 PRINT " O "
120 PRINT " ! "
130 PRINT "----"
140 PRINT " ! "
150 PRINT ") )"
155 REM FORM RIGHT-HAND FIGURE STARTING IN COLUMN 31
160 PRINT CHR$(19)
170 PRINT TAB(30)" O "
180 PRINT TAB(30)" ! "
190 PRINT TAB(30)"----"
200 PRINT TAB(30)" ! "
210 PRINT TAB(30)"( ("
215 REM MOVE BALL FROM LEFT TO RIGHT
220 FOR C=3 TO 27
230 PRINT CHR$(19)
240 PRINT TAB(C)" * "
250 PRINT TAB(C)"* *"
260 PRINT TAB(C)" * "
270 FOR K=1 TO 200:NEXT K
280 PRINT CHR$(19)
290 PRINT TAB(C)" "
300 PRINT TAB(C)" "
310 PRINT TAB(C)" "
320 NEXT C
330 PRINT CHR$(147)
RUN
```



Put 3 spaces here.

The figures stand still while the ball moves. We can make the ball move the other way by changing line 220 to

```
220 FOR C=27 TO 3 STEP -1
```

An obvious problem is that the ball is out of scale; it is almost as big as the figures, larger than even a basketball or medicine ball. To make the scene more realistic, enlarge the figures; try making them twice their current size, while keeping the ball as it is. You may also want to give the stick figures better forms and more personality.

# 5

## ROCKETS FLY, BLAST OFF, AND CHASE ONE ANOTHER

So far we have moved things horizontally. We would now like to move things vertically. First, let's draw a rocket. Then we will make it fly and have smoke trail from its engines.

### **Program 5-1 Rocket**

```
NEW
100 PRINT CHR$(147)
110 PRINT " /\ \"
120 PRINT " !! \"
130 PRINT " !! \"
140 PRINT " / \ \"
150 PRINT " ! ! \"
160 PRINT " ! ! \"
170 PRINT " ! ! \"
180 PRINT " --- \"
190 PRINT " VV \"
200 END
RUN
```

← / and \ are on the front of the N and M keys. Press SHIFT and N to type /. Press SHIFT and M to type \ .

The exclamation mark is uppercase I. Press SHIFT and I.

← Use the letter V to form the engines.

To make the rocket fly, we simply have the computer print blank lines under it. We must also start the rocket at the bottom using

```
101 FOR J=1 TO 25
103 PRINT
105 NEXT J
```

This sequence prints 25 blank lines, thus starting the rocket drawing at the bottom of the screen. Do not type NEW at this point, since we want to keep Program 5-1.



## Program 5-2 Rocket Flying Fast

```
101 FOR J=1 TO 25
103 PRINT
105 NEXT J
200 FOR J=1 TO 25
210 PRINT
220 FOR K=1 TO 200:NEXT K
230 NEXT J
240 PRINT CHR$(147)
RUN
```

Moves rocket to bottom of screen.

Makes rocket fly by printing blank lines underneath it.

Slows the computer down.

To make smoke trail from the rocket's engines, change line 210 to

```
210 PRINT "  **"
```

The asterisks under the engines represent smoke.

To draw many rockets at once, we put them all in PRINT statements. Let's draw two rockets being chased by another rocket.

## Program 5-3 Enemy Rocket Chasing Two Rockets, With Smoke

```
NEW
100 PRINT CHR$(147)
101 FOR J=1 TO 25
103 PRINT
105 NEXT J
109 REM DRAW TWO FRIENDLY ROCKETS
110 PRINT "  /\  /\  "
120 PRINT "  !!  !!  "
130 PRINT "  !!  !!  "
140 PRINT "  !!  !!  "
150 PRINT "  !!  !!  "
160 PRINT "  !!  !!  "
170 PRINT "  !!  !!  "
180 PRINT "  --  --  "
190 PRINT "  VV  VV  "
195 REM FLY FRIENDLY ROCKETS WITH TRAILING SMOKE
200 FOR J=1 TO 25
210 PRINT "  **  **  "
215 REM SLOW COMPUTER DOWN
220 FOR K=1 TO 300:NEXT K
230 NEXT J
235 REM DRAW ENEMY ROCKET
240 PRINT "  /-  "
250 PRINT "  ! !  "
260 PRINT "  !E!  "
270 PRINT "  !N!  "
280 PRINT "  !E!  "
290 PRINT "  !M!  "
300 PRINT "  !Y!  "
310 PRINT "  ---  "
320 PRINT "  V  "
325 REM FLY ENEMY ROCKET WITH TRAILING SMOKE
330 FOR J=1 TO 25
340 PRINT "  *  "
345 REM SLOW COMPUTER DOWN
350 FOR K=1 TO 200:NEXT K
360 NEXT J
370 PRINT CHR$(147)
RUN
```

Moves cursor to bottom of screen.

2 spaces

The double letter Vs represent each engine.

Smoke. You could try replacing the asterisks with the shaded character obtained by pressing COMMODORE and + together.

To enter the middle character, press COMMODORE and Y together.

Smoke

Enter the friendly rockets by copying line 120 to produce lines 130 through 170. The result will look like a squeezed-down version in the listing (since you will only see the last line), but will be correct when you RUN the program. To be sure everything is right, enter LIST to get a complete listing.

As the programs become longer, there will not be enough room on the screen for complete listings. To obtain a partial listing, type

```
LIST LOWER-UPPER
```

where LOWER and UPPER are line numbers in your program. You can omit LOWER or UPPER if you want to list everything from the beginning or through the end. For example, all the following commands work:

LIST 210-240 (lists lines starting with 210 and ending with 240)

LIST 210- (lists line 210 and everything after it)

LIST -240 (lists everything up to and including line 240)

LIST 210 (lists only line 210)

LIST (lists the entire program)

We can fly an alien spacecraft as follows.

#### **Program 5-4 Alien Spacecraft Flying, With Smoke**

```
NEW
100 PRINT CHR$(147)
101 FOR J=1 TO 25
103 PRINT
105 NEXT J
110 PRINT "X-----X"
120 PRINT "I @@@@ I"
130 PRINT "I ==== I"
140 PRINT "X-----X"
150 PRINT " !!  !! "
160 PRINT " --  -- "
170 FOR J=1 TO 25
180 PRINT "      @@@@ "
190 FOR K=1 TO 200:NEXT K
200 NEXT J
210 PRINT CHR$(147)
RUN
```

Moves cursor to bottom of screen.

6 dashes

Press COMMODORE and + simultaneously.

Of course, no launch is realistic without a countdown to blastoff. To produce a countdown, add the lines

```
50 PRINT CHR$(147)
60 FOR J=50 TO 1 STEP -1
70 PRINT CHR$(147)
80 PRINT J;" <==== "
90 NEXT J
```

To make the speed of the countdown more reasonable, insert

```
85 FOR K=1 TO 100:NEXT K
```

To draw a more complex warship, use the picture

```
  ( )
   ( )
    ( )
   ( )--( )--( )
  ( ) \MARS/ ( )
      \00/
```

Other exercises you could try include:

1. A rocket chasing an alien spacecraft.
2. A squadron of rockets flying in formation.
3. An airplane flying off the screen.
4. A three-stage rocket.

# 6

## CREATING MOTION PICTURES

To make a figure walk, we must draw several pictures of it, each in a different stage of walking. This is, after all, how motion pictures work. A movie is really just a series of still pictures separated in time. Let's look at three pictures of a bug.

Picture #1

```
!----\  
!  O!  
!----<  
////
```

Picture #2

```
!----\  
!  O!  
!----<  
!!!!
```

Picture #3

```
!----\  
!  O!  
!----<  
(((
```

If we draw the three pictures in the same place, the bug's legs appear to move. We can then TAB the bug to the right and it appears to walk. To make picture drawing easier, we introduce two new BASIC statements.

### **GOSUB AND RETURN STATEMENTS**

GOSUB sends the computer to another part of the program. RETURN tells it to go back to where it came from. Every GOSUB needs a RETURN and every RETURN a GOSUB. Think of the computer as a person reading a book. At a point on the page, the book says, "Look at the map on page 247." The reader marks the current page, turns to page 247, examines the map, and returns to where he or she left off. The reader then continues as if no interruption had occurred. That is how GOSUB and RETURN work. Let us now see how we use them in programs. We generally refer to the secondary part of the program (reached via a GOSUB) as a *subroutine*.

## Program 6-1 Moving Bug

```
NEW
100 PRINT CHR$(147)
110 FOR C=1 TO 25
120 GOSUB 1000
130 GOSUB 2000
140 GOSUB 3000
150 NEXT C
160 GOTO 9000
1000 PRINT CHR$(19)
1010 PRINT TAB(C) " !----\ "
1020 PRINT TAB(C) " !  O! "
1030 PRINT TAB(C) " !----< "
1040 PRINT TAB(C) "  //// "
1050 FOR K=1 TO 200:NEXT K
1999 RETURN
2000 PRINT CHR$(19)
2010 PRINT TAB(C) " !----\ "
2020 PRINT TAB(C) " !  O! "
2030 PRINT TAB(C) " !----< "
2040 PRINT TAB(C) "  !!!! "
2050 FOR K=1 TO 200:NEXT K
2999 RETURN
3000 PRINT CHR$(19)
3010 PRINT TAB(C) " !----\ "
3020 PRINT TAB(C) " !  O! "
3030 PRINT TAB(C) " !----< "
3040 PRINT TAB(C) "  ((( "
3050 FOR K=1 TO 200:NEXT K
3999 RETURN
9000 PRINT CHR$(147)
RUN
```

4 dashes followed by shifted M.  
Lines 2010 and 3010 are the same.

The legs are the only part of the bug that changes.

You could use shifted - instead of !. It produces a solid line.

If you use the CRSR keys to enter this program, you will surely need to LIST it to see all the lines. If you want to slow down the rate at which the listing moves up the screen, hold down the CTRL key. The lines will then appear one by one at a reasonable pace. Holding down CTRL always makes the computer run more slowly.

At line 120, GOSUB 1000 sends the computer to line 1000. It proceeds sequentially until it reaches the RETURN in 1999.

The computer then takes up where it left off—that is, right after GOSUB 1000. It does GOSUB 2000 and works through the second subroutine. When it returns from 2999, it does GOSUB 3000 and works through the third subroutine. When it returns from 3999, however, it proceeds to line 9000 and quits.

Line 160 introduces GOTO, another minor BASIC statement. GOTO simply tells the computer to do the specified line next rather than continuing in the usual numerical order. Thus GOTO 9000 means “do line 9000 next.”

The bug’s legs move as if it were running on a treadmill. To make its eye blink and its mouth open and close, all you must change is the eye in lines 2020 and 3020 and the mouth in lines 2030 and 3030. Program 6-2 contains the revisions. The rest of the program stays the same, so do not type NEW.

## Program 6-2 Moving Bug With Blinking Eyes and Open Mouth

Do not type NEW!

```

2020 PRINT TAB(C) " ! @! "
2030 PRINT TAB(C) " !-----= "
3020 PRINT TAB(C) " ! ■! "
3030 PRINT TAB(C) " !----- "
RUN

```

You can obtain all these lines easily from the old versions by using the CRSR keys.

Press SHIFT and Q simultaneously to enter a dark circle.

By changing the pictures, we can make any figure walk across the screen. Program 6-3 produces a walking stick figure.

## Program 6-3 Figure Walking

```

NEW
100 PRINT CHR$(147)
110 FOR C=1 TO 30
120 GOSUB 1000
130 GOSUB 2000
140 GOSUB 3000
150 NEXT C
160 GOTO 9000
1000 PRINT CHR$(19)
1010 PRINT TAB(C) " 0 "
1020 PRINT TAB(C) " ! "
1030 PRINT TAB(C) " -!- "
1040 PRINT TAB(C) " ! "
1050 PRINT TAB(C) " !-\ "
1060 PRINT TAB(C) " ! \ "
1070 FOR K=1 TO 100:NEXT K
1999 RETURN
2000 PRINT CHR$(19)
2010 PRINT TAB(C) " 0 "
2020 PRINT TAB(C) " ! "
2030 PRINT TAB(C) " /!- "
2040 PRINT TAB(C) " ! "
2050 PRINT TAB(C) " /-! "
2060 PRINT TAB(C) " / ! "
2070 FOR K=1 TO 100:NEXT K
2999 RETURN
3000 PRINT CHR$(19)
3010 PRINT TAB(C) " 0 "
3020 PRINT TAB(C) " ! "
3030 PRINT TAB(C) " -!\ "
3040 PRINT TAB(C) " ! "
3050 PRINT TAB(C) " /-/ "
3060 PRINT TAB(C) " / / "
3070 FOR K=1 TO 100:NEXT K
3999 RETURN
9000 PRINT CHR$(147)
RUN

```

3 spaces

You may want to use shifted - instead of ! in lines 1020 and 1040, and shifted + instead of ! in line 1030. Experiment and see how these characters look.

To have little aliens make faces as they move, we change pictures again.

## Program 6-4 Funny, Moving Aliens

```

NEW
100 PRINT CHR$(147)
110 FOR C=1 TO 30
120 GOSUB 1000
130 GOSUB 2000
140 GOSUB 3000
150 NEXT C
160 GOTO 9000
1000 PRINT CHR$(19)
1010 PRINT TAB(C) " \--/ "
1020 PRINT TAB(C) " (00) "
1030 PRINT TAB(C) " /--\ "
1040 FOR K=1 TO 100:NEXT K
1999 RETURN
2000 PRINT CHR$(19)
2010 PRINT TAB(C) " \--/ "
2020 PRINT TAB(C) " <@@" "
2030 PRINT TAB(C) " /--\ "
2040 FOR K=1 TO 100:NEXT K
2999 RETURN
3000 PRINT CHR$(19)
3010 PRINT TAB(C) " \--/ "
3020 PRINT TAB(C) " !●●! "
3030 PRINT TAB(C) " /--\ "
3040 FOR K=1 TO 100:NEXT K
3999 RETURN
9000 PRINT CHR$(147)
RUN

```

Press SHIFT and N for / , SHIFT and M for \ .

The solid circle is shifted Q.

Programs 6-1 through 6-4 change the appearances of objects as they move. If you want a creature to just make funny faces, simply remove the TAB(C)s. Now the creature does not move.

## Program 6-5 Stationary Creature Making Faces

```

NEW
100 PRINT CHR$(147)
110 FOR J=1 TO 10
120 GOSUB 1000
130 GOSUB 2000
140 GOSUB 3000
150 NEXT J
160 GOTO 9000
1000 PRINT CHR$(19)
1010 PRINT " !-----! "
1020 PRINT " ! 0 0 ! "
1030 PRINT " !  X  ! "
1040 PRINT " !  ■  ! "
1050 PRINT " !-----! "
1060 FOR K=1 TO 200:NEXT K
1999 RETURN
2000 PRINT CHR$(19)
2010 PRINT " !-----! "
2020 PRINT " ! @ @ ! "
2030 PRINT " ! 0  ! "
2040 PRINT " !  ■■■  ! "
2050 PRINT " !-----! "
2060 FOR K=1 TO 200:NEXT K
2999 RETURN

```

Lines 1010, 1050, 2010, 2050, 3010, and 3050 are all the same.

Press COMMODORE and U together 3 times.

Press COMMODORE and + together 3 times.

```

3000 PRINT CHR$(19)
3010 PRINT " !-----! "
3020 PRINT " ! X X ! "
3030 PRINT " ! @ ! "
3040 PRINT " ! VVV ! "
3050 PRINT " !-----! "
3060 FOR K=1 TO 200:NEXT K
3999 RETURN
9000 PRINT CHR$(147)
RUN

```

To draw a horse grazing in a field, we need three pictures:

PICTURE #1	PICTURE #2	PICTURE #3
<pre>       0     !---/     (---)     /! /!     /! /! </pre>	<pre>     !-----0     (---)     /! /!     /! /! </pre>	<pre>     !---     (---)\     /! /! \     /! /! 0 </pre>

### Program 6-6 Horse Grazing

```

NEW
100 PRINT CHR$(147)
110 FOR J=1 TO 10
120 GOSUB 1000
130 GOSUB 2000
140 GOSUB 3000
150 GOSUB 2000
160 GOSUB 1000
170 NEXT J
180 GOTO 9000
1000 PRINT CHR$(19)
1010 PRINT "      0 "
1020 PRINT " !---/ "
1030 PRINT " (---) "
1040 PRINT " /! /! "
1050 PRINT " /! /! "
1060 PRINT "#####"
1070 FOR K=1 TO 200:NEXT K
1990 RETURN
2000 PRINT CHR$(19)
2010 PRINT " "
2020 PRINT " !-----0"
2030 PRINT " (---) "
2040 PRINT " /! /! "
2050 PRINT " /! /! "
2070 FOR K=1 TO 200:NEXT K
2990 RETURN
3000 PRINT CHR$(19)
3010 PRINT " "
3020 PRINT " !--- "
3030 PRINT " (---)\ "
3040 PRINT " /! /! \ "
3050 PRINT " /! /! 0"
3070 FOR K=1 TO 200:NEXT K
3999 RETURN
9000 PRINT CHR$(147)
RUN

```

Annotations:

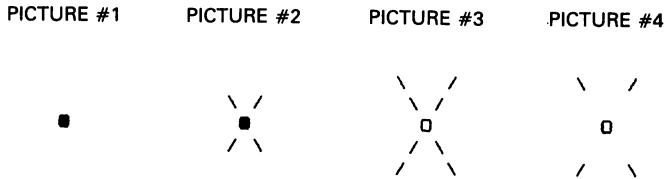
- 6 spaces (pointing to line 1010)
- Press COMMODORE and + together 8 times to give the horse some grass to eat. (pointing to line 1060)
- 8 spaces (pointing to line 2010)
- Putting shifted I here will connect the horse's neck to its body. (pointing to line 3020)



Be careful to extend all lines to the right boundary. The computer then prints spaces over the old drawings of the horse's neck and head. Otherwise, the horse will look like a two- or three-headed freak. It will look particularly strange if it leaves a head behind on the grass. If you want to make the grass (and horse) green, press CTRL and 6 together.

We can also use multiple pictures to create an exploding star like the ones often seen in video games.

Here we use four pictures.



### Program 6-7 Exploding Stars

```

NEW
100 PRINT CHR$(147)
110 FOR J=1 TO 10 ← 10 explosions
120 GOSUB 1000
130 GOSUB 2000
140 GOSUB 3000
150 GOSUB 4000
160 NEXT J
170 GOTO 9000
1000 PRINT CHR$(19)
1010 PRINT "
1020 PRINT "
1030 PRINT " █ ← Shifted Q
1040 PRINT "
1050 PRINT "
1060 FOR K=1 TO 200:NEXT K
1070 RETURN
2000 PRINT CHR$(19)
2010 PRINT "
2020 PRINT " \ / ← Shifted Q
2030 PRINT " █ ← Shifted Q
2040 PRINT " / \ ← Shifted Q
2050 PRINT "
2060 FOR K=1 TO 200:NEXT K
2070 RETURN
3000 PRINT CHR$(19)
3010 PRINT " \ /"
3020 PRINT " \ / ← Shifted W
3030 PRINT "  ○ ← Shifted W
3040 PRINT " / \ "
3050 PRINT " / \ "
3060 FOR K=1 TO 200:NEXT K
3070 RETURN
4000 PRINT CHR$(19)
4010 PRINT " \ /"
4020 PRINT "
4030 PRINT "  ○ ← Shifted W
4040 PRINT "
4050 PRINT " / \ "
4060 FOR K=1 TO 200:NEXT K
4070 RETURN
9000 PRINT CHR$(147)
RUN
    
```

To make the exploding star yellow, press CTRL and 8 together before running the program. Press STOP/RUN and RESTORE to return the screen to its normal light blue printing on dark blue background.

# 7

## POSITIONING FIGURES

In previous chapters, we used TAB to move objects right and left. For example, the following program moves an asterisk right.

### ***Program 7-1 Moving Asterisk Using TAB***

```
NEW
100 PRINT CHR$(147)
110 FOR C=1 TO 30
120 PRINT CHR$(19)
130 PRINT TAB(C) " * "
140 FOR K=1 TO 200:NEXT K
150 NEXT C
160 PRINT CHR$(147)
RUN
```

Put 1 space on each side of the asterisk.

TAB, however, has limitations. It can produce only horizontal motion, and the computer must move the cursor back to the upper left-hand corner after each picture. From the keyboard, we have more general control over where the computer prints—we can use the CRSR keys to move the cursor anywhere. But how do we put that control into a program? The answer is that we simply tell the computer to print a cursor control key. For example, if after typing PRINT "", you press the CRSR right-arrow key and then an ending quotation mark, you have instructed the computer to move the cursor right one column. Similarly, you can tell the computer to move the cursor left, up, or down within a program.

But what does the instruction look like on the screen? The television set cannot produce the key marking with the arrows. Table 7-1 shows what you see when you press the cursor-moving keys. We will use the following notation:

{UP}	Cursor up
{DOWN}	Cursor down
{RIGHT}	Cursor right
{LEFT}	Cursor left

A number in front of the direction indicates how many times you should press the key. For example,

{3 UP}	means	Press Cursor up three times
{5 RIGHT}	means	Press Cursor right five times

Now let us move the asterisk right with the CRSR keys. We must tell the computer to repeat the following steps:

1. Print the asterisk with a space on each side. Doing this moves the cursor three columns right.
2. Move the cursor left two columns (ending up one column right of where the current picture started).

The statements we need are

```
PRINT " * ";
PRINT "{2 LEFT}";
```

Note that the cursor moves right when you type or when the computer puts characters on the screen. The semicolons at the end of the PRINT statements tell the computer to continue where it left off.

**TABLE 7-1 CURSOR MOVING KEYS**

To move the cursor	Notation you will see in the book	Symbol you will see on the screen	Press these keys
Down	{DOWN}	␣	↑ CRSR ↓
Up	{UP}	␣	SHIFT    ↑ CRSR ↓
Right	{RIGHT}	␣	← CRSR →
Left	{LEFT}	␣	SHIFT    ← CRSR →

## Program 7-2 Asterisk Moving Left, Using Cursor Control

```

NEW
100 PRINT CHR$(147)
110 FOR C=1 TO 30
120 PRINT " * ";
130 PRINT "{2 LEFT}";
140 FOR K=1 TO 200:NEXT K
150 NEXT C
160 PRINT CHR$(147)
RUN

```

Put a space on each side of the asterisk.  
 Be sure to put a semicolon here.  
 Press SHIFT and CRSR twice.

We enter line 130 as follows:

Type 130 PRINT "

Press SHIFT and CRSR simultaneously

SHIFT and CRSR simultaneously

"

;

RETURN

The line appears on the screen as

```

130 PRINT "|||"; ← Reversed characters

```

Program 7-2 produces the same picture as Program 7-1. Note, however, that line 130 could easily move the cursor anywhere. The new position could be above or below the old one, at a diagonal from it, or separated from it by a random space warp.

To move a larger object, we must make the computer print a line, move the cursor, print another line, move the cursor, and so on. Let us, for example, consider the following three-line alien:

```

\ - /
[ O ] ← [ is shifted : (colon), ] is shifted ; (semicolon).
/ - \

```

This drawing is five columns wide, including one space on each side of it. Thus after the computer prints the first line, it must move the cursor left five columns and down one row before starting the second line. After printing the third line, it must move the cursor to the starting position for the next picture; that is, left four columns and up two rows to start one column right of where the previous picture started. Figure 7-1 contains a diagram of the cursor movement on a standard screen grid.

The instructions are

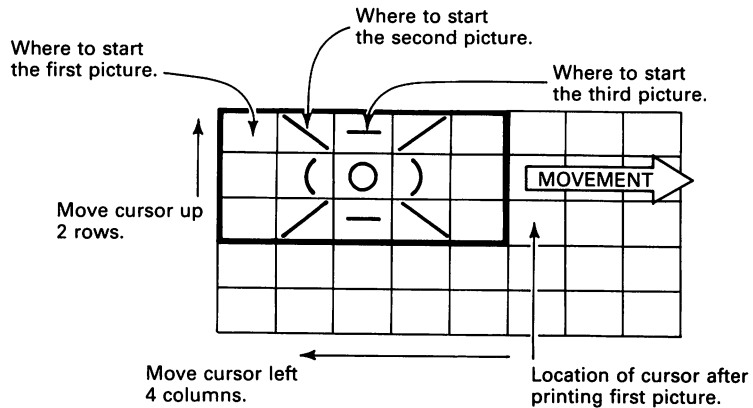
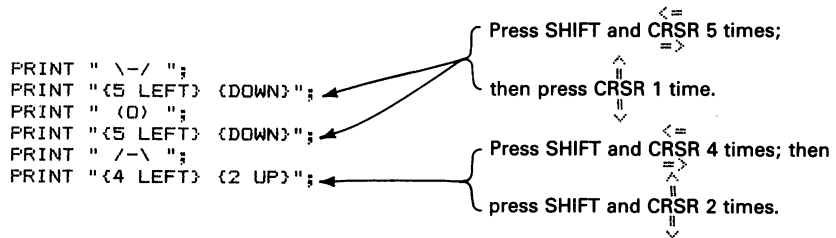


Figure 7-1 Cursor Control for Left to Right Movement of Alien.



The last PRINT moves the cursor left 4 columns and up two rows to where the next picture starts. To enter the line

```
PRINT "{5 LEFT} {DOWN}";
```

first enter the instruction

```
PRINT "
```

then press SHIFT and CRSR together five times. Note that the cursor moves right (since you are entering characters) as you enter the cursor left commands. Then press CRSR once. After that, enter the ending quotation marks (") and the semicolon (:). The line appears on the screen as

```
PRINT "■■■■[0]";
```

Reversed characters

### Program 7-3 Alien Moving Right, Using Cursor Control

```

NEW
100 PRINT CHR$(147)
110 FOR C=1 TO 30
120 PRINT " \-/ ";
130 PRINT "{5 LEFT} {DOWN}";
140 PRINT " (0) ";
150 PRINT "{5 LEFT} {DOWN}";
160 PRINT " /-\ ";
170 PRINT "{4 LEFT} {2 UP}";
180 FOR K=1 TO 200:NEXT K
190 NEXT C
200 PRINT CHR$(147)
RUN

```

Be sure to put a semicolon at the end of lines 120 through 170.

Put 1 space on each side of the alien.

Press SHIFT and CRSR 5 times;  
then press CRSR 1 time.

Press SHIFT and CRSR 4 times;  
then press SHIFT and CRSR 2 times.

Note that the cursor-moving commands are the same for all picture lines except the last one. The command for the last line (line 170) moves the cursor to where the next picture starts. In this case, since we want the alien to move right, the computer must move the cursor one column right of where it started the previous picture.

A new problem occurs when you enter Program 7-3. Obviously, you can use the cursor-moving keys as we described earlier to derive line 150 from line 130. That is, after entering line 130, you press SHIFT and CRSR, 1, 5, RETURN to enter line 150. But how about line 170? It is not very different from line 150, so perhaps we could change the line number and then move the cursor to change the last two cursor commands. This does not work—you cannot use the cursor-moving keys to edit lines that include cursor-moving keys (it sounds incestuous or cannibalistic, anyway). When you are editing lines, the cursor-moving keys move the cursor—they do not appear as entries.

Let us now move a larger picture, such as the following face:

```

!----!
! 0 0!
! XX !
!====!

```

This picture is eight columns wide (including a space on each side) and four rows high. The cursor commands for the first three lines are

```

PRINT "{8 LEFT} {DOWN}"; ← (8 columns left, 1 row down)

```

Figure 7-2 shows the use of a grid to determine the required cursor-moving commands. The cursor command after the last line (see Fig. 7-3) is

```

PRINT "{7 LEFT} {3 UP}"; ← (7 columns left, 3 rows up)

```

Remember, the cursor-moving command after the last picture line must move the cursor to where the next picture starts.

## Program 7-4 Face Moving Right, Using Cursor Control

```

NEW
100 PRINT CHR$(147)
110 FOR C=1 TO 30
120 PRINT " !----! ";
140 PRINT " !O O! ";
160 PRINT " ! XX ! ";
180 PRINT " !====! ";
130 PRINT "{8 LEFT} {DOWN}";
150 PRINT "{8 LEFT} {DOWN}";
170 PRINT "{8 LEFT} {DOWN}";
190 PRINT "{7 LEFT} {3 UP}";
200 FOR K=1 TO 200:NEXT K
210 NEXT C
220 PRINT CHR$(147)
RUN

```

Put 1 space on each side of the face.

Be sure to put a semicolon at the end of lines 120 through 190.

Press SHIFT and CRSR 8 times;

then press CRSR 1 time.

Press SHIFT and CRSR 7 times;

then press SHIFT and CRSR 3 times.

Note that we have separated the picture lines from the cursor-moving lines to make it easier to see how the picture will look. Besides, you can see the CRSR keys to derive lines 150 and 170 from line 130, although you cannot do this for line 190.

Listing the program produces the following display:

```

100 PRINT CHR$(147)
110 FOR C=1 TO 30
120 PRINT " !----! ";
130 PRINT " !■■■■■■■■O ";
140 PRINT " !O O! ";
150 PRINT " !■■■■■■■■O ";
160 PRINT " ! XX ! ";
170 PRINT " !■■■■■■■■O ";
180 PRINT " !====! ";
190 PRINT " !■■■■■■■■II ";
200 FOR K=1 TO 200:NEXT K
210 NEXT C
220 PRINT CHR$(147)

```

Reversed characters

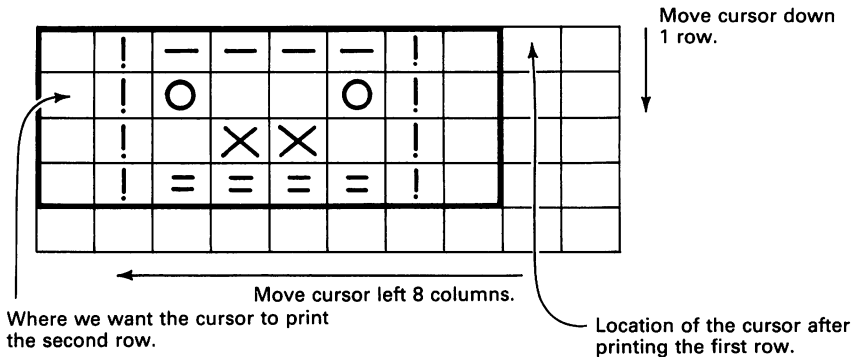
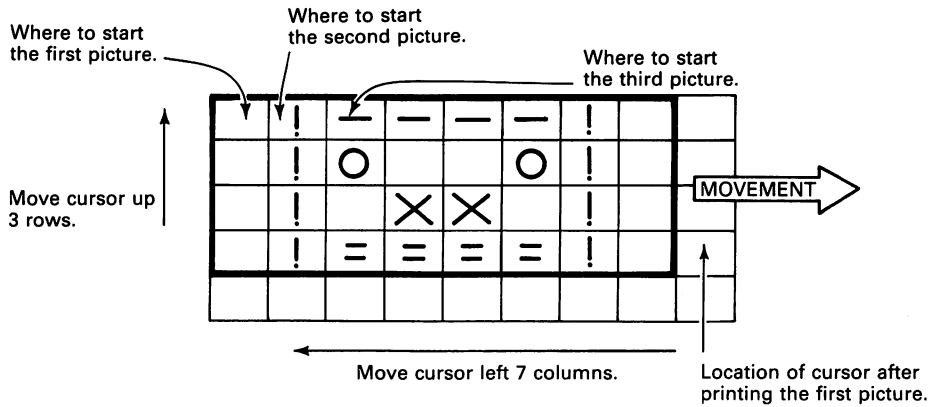


Figure 7-2 Grid Layout for the First Row of the Face.





**Figure 7-3** Cursor Control for Left to Right Movement of the Face.

We can use the cursor-moving keys to put messages anywhere on the screen. Let's use them to put names at several different places.

### **Program 7-5 Names All Over**

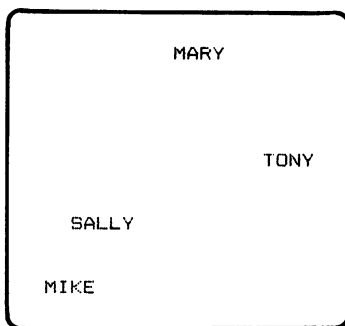
```

NEW
100 PRINT CHR$(147)
110 PRINT "{6 DOWN} {17 RIGHT}";
120 PRINT "TONY"
200 PRINT CHR$(19)
210 PRINT "{9 DOWN} {3 RIGHT}";
220 PRINT "SALLY"
300 PRINT CHR$(19)
310 PRINT "{2 DOWN} {8 RIGHT}";
320 PRINT "MARY"
400 PRINT CHR$(19)
410 PRINT "{12 DOWN}";
420 PRINT "MIKE"
RUN

```

← Down 6 rows and right 17 columns. You can use the repeat feature of the CRSR keys, but you must count the reversed characters.  
 ← Down 9 rows and right 3 columns.  
 ← Down 2 rows and right 8 columns.  
 ← Down 12 rows.

We do not need to end lines 120, 220, or 320 with semicolons, since we are using PRINT CHR\$(19) to return the cursor to the top left-hand corner after printing each name. The screen looks like



Now we can draw a lunar station at the bottom of the screen and let it fire shells upward. The lunar station looks like

```
!-!-!  
!-----!
```

Figure 7-4 illustrates the location of the station and the cursor positions.

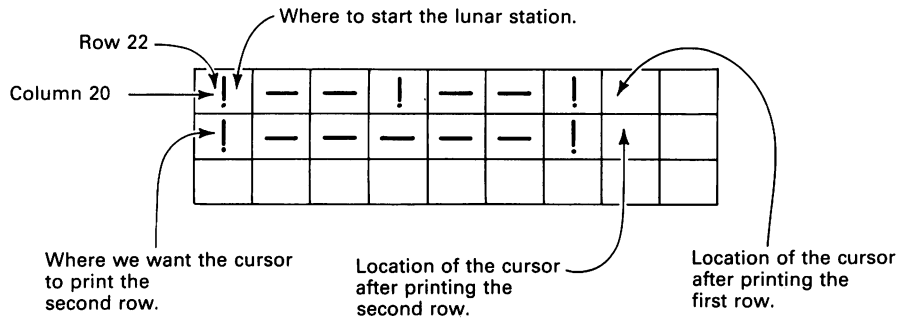


Figure 7-4 Grid Layout for Lunar Station.

To start drawing the lunar station in column 20 of row 22, we need

```
FOR K=1 TO 22  
PRINT  
NEXT K  
PRINT TAB(19); ← Moves the cursor right 19 columns.
```

Since the lunar station is seven columns wide (we do not need spaces around it), the PRINT commands to draw it are

```
PRINT "!-!-!";  
PRINT "{7 LEFT} {DOWN}";  
PRINT "!-----!";
```

After the computer draws the station, we want to move the cursor just above its center to begin firing a shell. This requires

```
PRINT "{4 LEFT} {2 UP}";
```

To move the shell upward, we use

```
FOR R=1 TO 22  
PRINT "*";  
PRINT "{UP} {LEFT}"; ← Moves the cursor up 1 row and  
FOR K=1 TO 200:NEXT K  
NEXT R
```

## Program 7-6 Lunar Station Firing a Shell

```

NEW
100 PRINT CHR$(147)
105 REM MOVE DOWN 22 LINES
110 FOR J=1 TO 22
120 PRINT
130 NEXT J
135 REM MOVE RIGHT TO COLUMN 20
140 PRINT TAB(19);
150 PRINT "!-!-!-!";
160 PRINT "<7 LEFT> <DOWN>";
170 PRINT "!-!-!-!";
180 PRINT "<4 LEFT> <2 UP>";
190 FOR R=1 TO 22
200 PRINT "*";
210 PRINT "<LEFT> <UP>";
220 FOR K=1 TO 200:NEXT K
230 NEXT R
240 PRINT CHR$(147)
RUN

```

Place a semicolon here.

Press SHIFT and CRSR 7 times,  
then press CRSR.

Press SHIFT and CRSR 4 times,  
then press SHIFT and CRSR twice.

Press SHIFT and CRSR, then  
SHIFT and CRSR.

To speed the shell up, change 200 in line 220 to 50. To eliminate the column of shells, we must erase each one after printing it. This requires backspacing the cursor (left one column) and printing a space.

## Program 7-7 Lunar Station Firing a Shell, With Erasure

```


210 PRINT "<LEFT>";
230 PRINT " ";
240 PRINT "<LEFT> <UP>";
250 NEXT R
260 PRINT CHR$(147)
RUN

```

Put 1 space here.


# 8

## USING GRAPHICS CHARACTERS

To enter graphics symbols, we must use either the SHIFT key or the COMMODORE key . To enter a right-hand graphics symbols on the front of a key, press the key while holding SHIFT down. To enter a left-hand symbol, press the key while holding COMMODORE (bottom left-hand corner) down. For example, let's draw a box.

### Program 8-1 Simple Box

```
NEW
100 PRINT CHR$(147)
110 PRINT " "
120 PRINT " "
130 PRINT " "
140 PRINT " "
RUN
```



Press COMMODORE and L to form the left side.  
Press COMMODORE and U to form the top.  
Press COMMODORE and J to form the right side.  
Press COMMODORE and I to form the bottom.

We enter line 110 as follows:

Type	110 PRINT "
Press	COMMODORE L
	COMMODORE U ■
	COMMODORE U ■
	COMMODORE U ■
	COMMODORE U ■
	COMMODORE J
	SHIFT 2 "
	RETURN

Note: You can simply hold COMMODORE down and type L, U, U, U, U, J. Then release COMMODORE and press SHIFT and 2 together.

Figure 8-1 contains a list of the graphics symbols and their locations on the keyboard. Be careful not to accidentally press COMMODORE and SHIFT together. This puts the computer in text mode (usually obvious since all the letters on the screen change to lowercase). To get back to graphics mode, press COMMODORE and SHIFT together again.































































Use SHIFT		Use COMMODORE	
and	to get	and	to get
Z		Z	
X		X	
C		C	
V		V	
B		B	
N		N	
M		M	
A		A	
S		S	
D		D	
F		F	
G		G	
H		H	
J		J	
K		K	
L		L	
Q		Q	
W		W	
E		E	
R		R	
T		T	
Y		Y	
U		U	
I		I	
O		O	
P		P	
@		@	
*		*	
+		+	
-		-	
£		£	

Figure 8-1 Commodore 64 Graphics Characters

We can draw more complex pictures by combining graphics symbols.

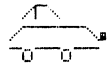


### Program 8-4 Moving Car

```

NEW
100 PRINT CHR$(147)
110 FOR C=1 TO 30
120 PRINT CHR$(19)
130 PRINT TAB(C) "
140 PRINT TAB(C) "
150 PRINT TAB(C) "
160 FOR K=1 TO 200:NEXT K
170 NEXT C
180 PRINT CHR$(147)
RUN

```



← Put a space at each end of the car.  
This makes line 130 start with 3 spaces.

We can draw a strange face and move it across the screen. The face looks like



← COMMODORE L, 5 COMMODORE Us, COMMODORE J

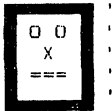
← COMMODORE L, 5 COMMODORE Is, COMMODORE J

### Program 8-5 Moving Graphics Face

```

NEW
100 PRINT CHR$(147)
110 FOR C=1 TO 30
120 PRINT CHR$(19)
130 PRINT TAB(C) "
140 PRINT TAB(C) "
150 PRINT TAB(C) "
160 PRINT TAB(C) "
170 PRINT TAB(C) "
180 FOR K=1 TO 200:NEXT K
190 NEXT C
200 PRINT CHR$(147)
RUN

```



← Use shifted W for the eyes.

← Put 1 space on each side of the picture.

To make eyes blink as they move, we must draw two pictures of them in different stages of blinking at each position.

### Program 8-6 Moving, Blinking Eyes

```

NEW
100 PRINT CHR$(147)
110 FOR C=1 TO 30
120 PRINT CHR$(19)
130 PRINT TAB(C) " <O> <O> "
140 FOR K=1 TO 100:NEXT K
150 PRINT CHR$(19)
160 PRINT TAB(C) " <■> <■> "
170 FOR K=1 TO 100:NEXT K
180 NEXT C
190 PRINT CHR$(147)
RUN

```

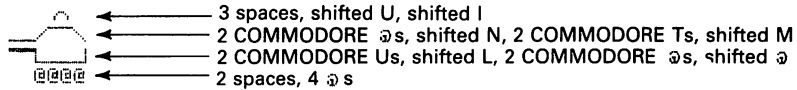
← Shifted W

← Shifted Q

The eyes are moving too fast. To slow them down, instead of using a tight loop, we make them blink five times at each location. This involves a loop inside a loop, called a *nested* loop. The outer loop moves the eyes right, while the inner loop makes them blink five times. The additions are

```
115 FOR J=1 TO 5
175 NEXT J
```

We can also draw a tank and move it left. The tank looks like

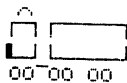


Note the mixture of right-hand and left-hand graphics. It may take you a few tries to enter the picture correctly.

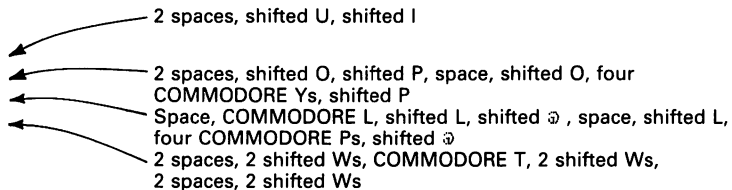
### Program 8-7 Graphics Tank Moving Left

```
NEW
100 PRINT CHR$(147)
110 FOR C=30 TO 1 STEP -1
120 PRINT CHR$(19)
130 PRINT TAB(C) " "
140 PRINT TAB(C) " " ← Put 1 space on each side of the tank.
150 PRINT TAB(C) " "
160 PRINT TAB(C) " "
170 FOR K=1 TO 200:NEXT K
180 NEXT C
190 PRINT CHR$(147)
RUN
```

Another picture is a large truck with a trailer. It looks like



The lines are





## Program 8-8 Graphics Truck/Trailer Moving Left

```
NEW
100 PRINT CHR$(147)
110 FOR C=25 TO 1 STEP -1
120 PRINT CHR$(19)
130 PRINT TAB(C)" " "
140 PRINT TAB(C)"  | | " ← Put 1 space on each side of the truck.
150 PRINT TAB(C)"  | | "
160 PRINT TAB(C)" 00 00 00 "
170 FOR K=1 TO 200:NEXT K
180 NEXT C
190 PRINT CHR$(147)
RUN
```

# 9

## PLACING PICTURES RANDOMLY

We can easily place objects randomly on the screen. BASIC has a special way to pick a random number, one that is equally likely to have any value within its limits. This is like rolling an honest die, which is equally likely to give any value from 1 to 6. On the Commodore 64,

```
K=INT(RND(1)*N)+1
```

Note: \* (not X) means "multiply" in BASIC.

makes the computer pick a random whole number K between 1 and N. RND(1) selects a random number between 0 and 1, while INT makes RND(1)\*N into a whole number (or integer). Note that RND is always less than 1, so RND(1)\*N is always less than N. For example,

```
D=INT(RND(1)*6)+1
```

produces a random number between 1 and 6 that could simulate the rolling of dice. Similarly,

```
R=INT(RND(1)*38)+1
```

simulates a random spin of a roulette wheel (one color only). To select an arbitrary position on the screen, we need two random numbers. The row (line) number R must be between 1 and 24 lines down, while the column number C must be between 0 and 39 columns right.

To move down a random number of rows (1 to 24), we use

```
R=INT(RND(1)*24)+1  
FOR J=1 TO R  
PRINT  
NEXT J
```

Will be a number between 1 and 24.  
Note that we are not using the  
top line here.

Similarly, to move right to a random column, we use

```
C=INT(RND(1)*40)  
PRINT TAB(C)
```

Will be a number between 0 and 39.

The following program places asterisks at random positions, creating the effect of falling snowflakes.

### Program 9-1 Snowfall

```
NEW  
100 PRINT CHR$(147)  
110 FOR N=1 TO 30  
120 PRINT CHR$(19);  
125 REM MOVE TO A RANDOM ROW  
130 R=INT(RND(1)*24)+1  
140 FOR J=1 TO R  
150 PRINT  
160 NEXT J  
165 REM MOVE TO A RANDOM COLUMN  
170 C=INT(RND(1)*40)  
180 PRINT TAB(C)"*";  
190 NEXT N  
200 PRINT CHR$(147)  
RUN
```

We will draw 30 objects.

Put a semicolon here.

The computer makes up a  
number between 1 and 24.

The computer makes up a  
number between 0 and 39.

The computer selects the vertical and horizontal positions of the asterisks randomly. To slow down the rate at which asterisks appear, insert

```
185 FOR K=1 TO 100:NEXT K
```

We can also make the asterisks disappear by adding

```
188 PRINT "{LEFT}"; " "
```

Press SHIFT and CRSR.

1 space here.

The result looks like a firefly flitting rapidly around the screen. You can slow it down by making the upper value in line 185 either 200 or 500. A value of 500 gives the appearance of a slow-motion camera.

If you run the original Program 9-1 several times, you will probably notice some erratic behavior. Occasionally, the entire display shifts up a line. This occurs because the rightmost column (column 40) and the bottom row (row 25) are special. When the com-

puter prints in column 40, it then moves automatically to the next line. When it finishes printing in row 25, it then automatically moves the entire screen up one line. Obviously, these actions are necessary to give the computer a place to put the next character.

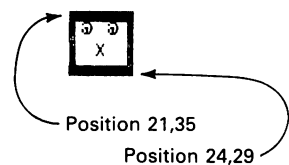
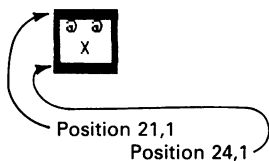
The way to keep these automatic adjustments from spoiling pictures is simple: Never print in row 25 or column 40. To keep Program 9-1 away from the boundaries, change lines 130 and 170 to

```
130 R=INT (RND (1) *23) +1
170 C=INT (RND (1) *39)
```

Now the row number will never exceed 24, and the column number will never exceed 39. A more complex figure is a face that occupies four lines and five columns.



We must make sure the face fits everywhere the computer is to print it.



This drawing shows that the face cannot fit if it starts beyond row 21 or column 35. As before, we must avoid the bottom row and the rightmost column. Thus R and C cannot be larger than 20 and 34, respectively. So, we must use

```
R=INT (RND (1) *20) +1
```

and

```
C=INT (RND (1) *35)
```

**Program 9-2 Faces Appear and Disappear at Random Positions**

```
NEW
100 PRINT CHR$(147); ← Put a semicolon here.
105 REM DRAW AND ERASE 30 FACES
110 FOR N=1 TO 30
115 REM MOVE TO RANDOM ROW
120 R=INT (RND (1) *20) +1
130 FOR J=1 TO R
140 PRINT
150 NEXT J
155 REM MOVE TO RANDOM COLUMN
```

```

160 C=INT(RND(1)*35)
170 PRINT TAB(C);
200 PRINT "  ";
220 PRINT "  @  " ←
240 PRINT "  X  " ←
260 PRINT "  ";
210 PRINT "{5 LEFT} {DOWN}";
230 PRINT "{5 LEFT} {DOWN}"; ←
250 PRINT "{5 LEFT} {DOWN}"; ←
270 FOR K=1 TO 200:NEXT K
275 REM ERASE FACE WITH SPACES
276 REM MOVE TO START OF FACE
290 PRINT "  ";
310 PRINT "  "; ←
330 PRINT "  ";
350 PRINT "  ";
280 PRINT "{5 LEFT} {3 UP}"; ←
300 PRINT "{5 LEFT} {DOWN}"; ←
320 PRINT "{5 LEFT} {DOWN}"; ←
340 PRINT "{5 LEFT} {DOWN}"; ←
360 PRINT CHR$(19); ←
370 NEXT N
380 PRINT CHR$(147)
RUN

```

Use the graphics characters on the J (left side), O (top), L (right side), and U (bottom) keys.

Press SHIFT and CRSR 5 times, then press CRSR.

Put 5 spaces here.

Press SHIFT and CRSR 5 times, then SHIFT and CRSR 3 times.

Put a semicolon here.

Programs 9-2 draws the face, starting at a random position, and then makes it disappear by printing spaces over it. Note that we put semicolons after all PRINTs. We even put a semicolon after PRINT CHR\$(147) and PRINT CHR\$(19). After each PRINT, regardless of whether it contains letters, numbers, cursor moves, or screen commands, the computer normally proceeds to the next line. A semicolon prevents this automatic advancement.

To slow the face movement down, insert

```
365 FOR K=1 TO 200:NEXT K
```

To make the face stay in each place longer, change the 200 in line 270 to 500.

We can easily scatter objects randomly on the screen using any character in Table 9-1. To make the computer print a character, we use CHR\$ with the corresponding number. Typical examples are

CHARACTER	USE
\$	PRINT CHR\$(36)
+	PRINT CHR\$(43)
#	PRINT CHR\$(35)
*	PRINT CHR\$(42)

Note, for example, that either PRINT "\*" or PRINT CHR\$(42) prints an asterisk.

Let's redo Program 9-1 but, instead of printing an asterisk, print a different character each time. We can use Table 9-1 to select:

1. A random letter. The random number must lie in the range 65 to 90 inclusive, so we need

```
L=INT(RND(1)*26)+65
```

2. A random right-hand graphics character. The random number must lie in the range 96 to 127, so we need

```
RH=INT(RND(1)*32)+96
```

3. A random left-hand graphics character. The random number must lie in the range 161 to 191, so we need

```
LH=INT(RND(1)*31)+161
```

4. A random letter, right-hand graphics character, or symbol. The random number must lie in the range 64 to 127, so we need

```
S=INT(RND(1)*64)+64
```

So Program 9-1 can be modified to scatter random letters, right-hand graphics, and other miscellaneous characters as follows.

### ***Program 9-3 Random Characters at Random Positions***

```
NEW
100 PRINT CHR$(147);
110 FOR N=1 TO 30
120 PRINT CHR$(19);
125 REM MOVE DOWN TO A RANDOM ROW
130 R=INT(RND(1)*23)+1
140 FOR J=1 TO R
150 PRINT
160 NEXT J
165 REM MOVE RIGHT TO A RANDOM COLUMN
170 C=INT(RND(1)*39)
175 REM GENERATE A RANDOM SYMBOL
180 S=INT(RND(1)*64)+64
190 PRINT TAB(C)CHR$(S);
200 NEXT N
210 PRINT CHR$(147)
RUN
```

← We will place 30 characters.

← Put a semicolon here.

← The computer makes up a number between 1 and 23.

← C is between 0 and 38.

← S is between 64 and 127.

← Puts the random symbol in a random column.

To leave the characters on the screen, delete line 210. You will now have to clear the screen manually (by pressing SHIFT and CLR/HOME together) before starting again. If you change line 210 to

```
210 GOTO 110
```

the program will run forever, filling the screen with a living, constantly changing picture. To stop the process, press RUN/STOP and clear the screen. To change the mixture of symbols, change the values in line 180.

We can combine random positioning with multiple pictures to create random exploding stars (or supernovas). Our three-star pictures here are

PICTURE #1 — Unextended

•

PICTURE #2 — Partially Extended

```
  \!/  
 -0-  
 /!\
```

PICTURE #3 — Fully Extended

```
  \ ! /  
   \!/  
  --0--  
   /!\  
  / ! \
```

Since this picture in its extended form is five lines high and five columns wide, we must restrict its starting position to rows 1 through 20 and columns 1 through 35.

#### ***Program 9-4 Exploding Supernovas at Random Positions***

```
NEW  
100 PRINT CHR$(147);  
105 REM FIND RANDOM ROW AND COLUMN FOR STAR  
110 FOR N=1 TO 10  
120 PRINT CHR$(19);  
130 R=INT(RND(1)*19)+1  
140 C=INT(RND(1)*35)  
145 REM MOVE DOWN A RANDOM NUMBER OF ROWS  
150 FOR J=1 TO R  
160 PRINT  
170 NEXT J  
175 REM MOVE RIGHT A RANDOM NUMBER OF COLUMNS  
180 PRINT TAB(C);  
185 REM START WITH STAR UNEXTENDED  
190 GOSUB 1000  
200 PRINT "{4 UP} {5 LEFT}";  
210 FOR K=1 TO 100:NEXT K  
215 REM DRAW STAR PARTIALLY EXTENDED  
220 GOSUB 2000  
230 PRINT "{4 UP} {5 LEFT}";  
240 FOR K=1 TO 100:NEXT K  
245 REM DRAW STAR FULLY EXTENDED  
250 GOSUB 3000  
260 PRINT "{4 UP} {5 LEFT}";  
270 FOR K=1 TO 100:NEXT K  
275 REM DRAW STAR PARTIALLY EXTENDED  
280 GOSUB 2000  
290 PRINT "{4 UP} {5 LEFT}";  
300 FOR K=1 TO 100:NEXT K  
305 REM DRAW STAR UNEXTENDED  
310 GOSUB 1000  
320 PRINT "{4 UP} {5 LEFT}";  
330 FOR K=1 TO 100:NEXT K  
340 NEXT N  
350 GOTO 9000
```

```

995 REM PICTURE OF STAR UNEXTENDED
1000 PRINT "      " ;
1020 PRINT "      " ;
1040 PRINT "      ■ " ; ← Shifted Q
1060 PRINT "      " ;
1080 PRINT "      " ;
1010 PRINT "{DOWN} {5 LEFT}";
1030 PRINT "{DOWN} {5 LEFT}";
1050 PRINT "{DOWN} {5 LEFT}";
1070 PRINT "{DOWN} {5 LEFT}";
1090 RETURN
1995 REM PICTURE OF STAR PARTIALLY EXTENDED
2000 PRINT "      " ; ← 5 spaces
2020 PRINT "    \!/" ; ← Space, shifted (M, B, N)
2040 PRINT "  -D-" ; ← Space, shifted (C, W, C), space
2060 PRINT "  /!\ " ; ← Space, shifted (N, B, M)
2080 PRINT "      " ; ← 5 spaces
2010 PRINT "{DOWN} {5 LEFT}";
2030 PRINT "{DOWN} {5 LEFT}";
2050 PRINT "{DOWN} {5 LEFT}";
2070 PRINT "{DOWN} {5 LEFT}";
2090 RETURN
2995 REM PICTURE OF STAR FULLY EXTENDED
3000 PRINT "\ ! /" ; ← Shifted M, space, shifted B, space, shifted N
3020 PRINT " \!/" ; ← Space, shifted (M, B, N), space
3040 PRINT "--D--" ; ← Shifted (C, C, W, C, C)
3060 PRINT " /!\ " ; ← Space, shifted (N, B, M), space
3080 PRINT "/ ! \\" ; ← Shifted N, space, shifted B, space, shifted M
3010 PRINT "{DOWN} {5 LEFT}";
3030 PRINT "{DOWN} {5 LEFT}";
3050 PRINT "{DOWN} {5 LEFT}";
3070 PRINT "{DOWN} {5 LEFT}";
3090 RETURN
9000 PRINT CHR$(147)
RUN

```

To change the rate at which the star expands and contracts, change the upper limits in the do-nothing loops in lines 210, 240, 270, 300, and 330. To erase the burned-out hulks of the exploded supernovas, add

```

325 PRINT "{2 UP} {3 LEFT}";
326 PRINT " "

```

← 1 space

We can use random letters to generate random four-letter words. How do you think they will turn out?



## Program 9-5 Random Four-Letter-Word Generator

```
NEW
100 PRINT CHR$(147)
110 FOR J=1 TO 50
120 PRINT CHR$(19)
125 REM GENERATE FOUR RANDOM LETTERS
130 L1=INT(RND(1)*26)+65 ←
140 L2=INT(RND(1)*26)+65 ←
150 L3=INT(RND(1)*26)+65 ←
160 L4=INT(RND(1)*26)+65 ←
165 REM DISPLAY WORD NEAR CENTER OF SCREEN
170 FOR R=1 TO 10
180 PRINT
190 NEXT R
200 PRINT "THE NEXT WORD IS ";CHR$(L1);CHR$(L2);CHR$(L3);CHR$(L4)
210 FOR K=1 TO 1000:NEXT K
220 NEXT J
RUN
```

L1, L2, L3, and L4 are all between 65 and 90.

If you want a different sequence of random letters each time you RUN the program, insert

```
105 L=RND(-TI)
```

Unfortunately (or, fortunately, if you are a worried teacher), most four-letter combinations don't make much sense, although they could serve as government acronyms. Run Program 9-5 for a while and see how long it takes the computer to generate a real word. One way to improve your chances is to change line 140 to










































































```
140 L2=2*INT(RND(1)*11)+65
```

Examine Table 9-1 to see why this increases the likelihood of L2 being a vowel.

Note that lines 130 through 160 can produce four different letter codes, although they all look the same. RND, remember, produces a different random number each time.

The reason why the four-letter combinations are so disappointing is that letters in English words are not random. As anyone who has played word games knows, some letters, such as vowels and the consonants *l*, *n*, *t*, *r*, and *s* are very common, while others, such as *j*, *k*, *q*, *x*, and *z* are uncommon. Furthermore, some pairs, such as *ch*, *sh*, and *th* are common, while others, such as *kq*, *lz*, *qd*, and *nj*, are either uncommon or nonexistent.

TABLE 9-1 CHARACTERS AND THEIR CHR\$ VALUES

@	64	U	85		106		161		181
A	65	V	86		107		162		182
B	66	W	87		108		163		183
C	67	X	88		109		164		184
D	68	Y	89		110		165		185
E	69	Z	90		111		166		186
F	70	[	91		112		167		187
G	71	£	92		113		168		188
H	72	]	93		114		169		189
I	73	↑	94		115		170		190
J	74	←	95		116		171		191
K	75		96		117		172		
L	76		97		118		173		
M	77		98		119		174		
N	78		99		120		175		
O	79		100		121		176		
P	80		101		122		177		
Q	81		102		123		178		
R	82		103		124		179		
S	83		104		125		180		
T	84		105		126				
					127				

# 10

## PERSONAL MESSAGES: NAMES AND CARDS

### **TALKING TO THE COMPUTER**

INPUT statements let the computer ask you for information. A typical question-and-answer session works much like communicating with a person via telegraph (or doing your banking on an automatic teller terminal). In the program we have a statement like

```
100 INPUT "WHAT IS YOUR NAME";N$
```

The \$ indicates that your name consists of letters (and perhaps a hyphen if you are fancy) rather than numbers. When the computer reaches line 100, it prints

```
WHAT IS YOUR NAME?
```

on the screen, followed by the cursor, and waits for you to answer. Note that INPUT makes the computer print a question mark automatically to show you when and where to respond.

If you type

```
TONY (and then press RETURN)
```

in response, N\$ will be the letters T, O, N, Y. Let us now use INPUT to revise some simple name-printing programs from Chapter 2.

### **Program 10-1 Printing Your Name 20 Times**

```
NEW
100 PRINT CHR$(147)
110 INPUT "WHAT IS YOUR NAME";N$
120 PRINT CHR$(147)
130 FOR J=1 TO 20
140 PRINT N$
150 NEXT J
RUN
```

You will see

```
WHAT IS YOUR NAME?
```

on the screen, followed by a space and the cursor. Type your name and press RETURN. The screen now looks the same as it did after Program 2-3. You can use INST/DEL and the CRSR keys to edit your response just as if it were a program line.

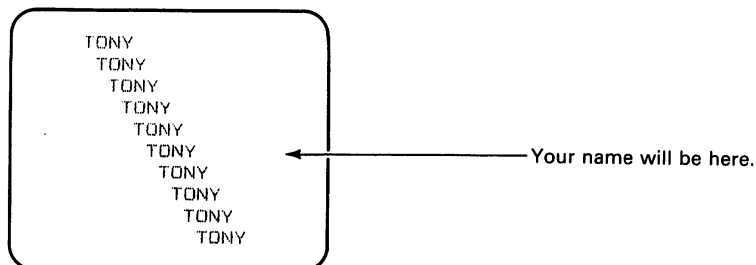
### **Program 10-2 Your Name Moving Right**

```
NEW
100 PRINT CHR$(147)
110 INPUT "WHAT IS YOUR NAME";N$
120 PRINT CHR$(147)
130 FOR C=1 TO 10
140 PRINT TAB(C)N$
150 NEXT C
RUN
```

Again you will see

```
WHAT IS YOUR NAME?
```

on the screen. Type your name and press RETURN. The screen should look like



```
TONY
  TONY
    TONY
      TONY
        TONY
          TONY
            TONY
              TONY
                TONY
                  TONY
```

The nice feature of using INPUT is that we can type RUN again and enter another name. The computer will print the other name; we do not have to change the program. We can print any name just by typing it when we see

```
WHAT IS YOUR NAME?
```

on the screen. Run Program 10-2 and enter a friend's name when you see

```
WHAT IS YOUR NAME?
```

You can see how handy INPUT is.

Let's redo Program 2-9 and produce a giant crossing pattern.

### **Program 10-3 Input Names in Crossing Pattern**

```
NEW
100 PRINT CHR$(147)
110 INPUT "WHAT IS YOUR NAME";N$
120 INPUT "WHAT IS YOUR FRIEND'S NAME";M$
130 PRINT CHR$(147)
140 FOR C=1 TO 15
150 PRINT TAB(C)N$
160 PRINT TAB(16-C)M$
170 NEXT C
RUN
```

This statement overruns the 40-column screen, so the computer automatically continues it on the next line. The result looks odd, but works correctly. Be sure, however, to press RETURN after each statement.

The computer prints

```
WHAT IS YOUR NAME?
```

Type your name and press RETURN. The computer now prints

```
WHAT IS YOUR FRIEND'S NAME?
```

Go ahead and enter your friend's name! You should get the same result as from Program 2-10. By using two INPUT statements, we can enter two different names and the computer will store them in N\$ and M\$. We can then put N\$ and M\$ wherever we want the names to appear. If we enter the names TONY and SALLY in Program 10-3, N\$ will be T, O, N, Y and M\$ will be S, A, L, L, Y.

```
N$←TONY
M$←SALLY
```

Be careful of statements that extend to or past the right-hand screen boundary. You must press RETURN after completing them even though the cursor has already moved down a line. The same precaution applies when responding to INPUT statements. If your friend's name is ERATOSTHENES or CATHERINE THE GREAT, it will spill over onto the next line. You must press RETURN after typing it.

We can extend Program 10-3 to make our own greeting cards. For Valentine's Day, we could draw a heart and put a friend's name on it.

## Program 10-4 Valentine's Day Card

```

NEW
100 PRINT CHR$(147)
110 INPUT "WHAT IS YOUR FRIEND'S NAME";N$
120 PRINT CHR$(147)
130 PRINT " ** * * "
140 PRINT " * * * * "
150 PRINT " * * * * "
160 PRINT " * * * * "
170 PRINT " * * * * "
180 PRINT " * * * * "
190 PRINT " * * * * "
200 PRINT
210 PRINT "HAPPY VALENTINE'S"
220 PRINT "DAY, ";N$
RUN

```

3 spaces

To avoid having READY and the blinking cursor spoil the card, add

```
230 GOTO 230
```

This line puts the computer in an endless loop. You must press RUN/STOP to regain control.

To center the card and the heart, insert

```

122 FOR J=1 TO 5
124 PRINT
126 NEXT J
128 PRINT TAB(15);
135 PRINT TAB(15);
145 PRINT TAB(15);
155 PRINT TAB(15);
165 PRINT TAB(15);
175 PRINT TAB(15);
185 PRINT TAB(15);
205 PRINT TAB(11);
215 PRINT TAB(15);

```

← Watch line 205—the TAB is different.

For a Christmas card, we could draw a Christmas tree and print a friend's name.

## Program 10-5 Christmas Card

```

NEW
100 PRINT CHR$(147)
110 INPUT "WHAT IS YOUR NAME";N$
120 INPUT "WHAT IS YOUR FRIEND'S NAME";M$
130 PRINT CHR$(147)
140 PRINT " / \ "
150 PRINT " / ** \ "
160 PRINT " / **** \ "
170 PRINT " / ***** \ "
180 PRINT " / ***** \ "
190 PRINT " _____ "
200 PRINT " □ "
210 PRINT
220 PRINT "MERRY CHRISTMAS"
230 PRINT "AND HAPPY NEW YEAR"

```

4 spaces

← 3 COMMODORE Ts, shifted (P, O, P, O), 3 COMMODORE  
← 3 spaces, COMMODORE M, shifted L, shifted @,  
COMMODORE G

```

240 PRINT
250 PRINT "TO: ";M$
260 PRINT
270 PRINT "FROM: ";N$
RUN

```

The parentheses after SHIFT indicate that you should hold SHIFT down while typing all the characters inside.

To stop a program that is waiting for a response, you must press RUN/STOP and RESTORE (directly above RETURN) simultaneously. RUN/STOP alone is not enough in this case. Note that pressing RUN/STOP and RESTORE also restores the computer's normal light blue printing color.

We can use the same approach to create other cards.

### Program 10-6 New Year's Card

```

NEW
100 PRINT CHR$(147)
110 INPUT "WHAT IS YOUR NAME";N$
120 INPUT "WHAT IS YOUR FRIEND'S NAME";M$
130 PRINT CHR$(147)
140 PRINT "  /\  "
150 PRINT " /  \ "
160 PRINT "\  HAPPY  / "
170 PRINT " \  NEW  / "
180 PRINT " /  YEAR  \ "
190 PRINT "\  \  "
200 PRINT " /  \  /  \  "
210 PRINT "  \  /  "
220 PRINT
230 PRINT "BEST WISHES"
240 PRINT
250 PRINT "TO: ";M$
260 PRINT
270 PRINT "FROM: ";N$
RUN

```

4 spaces  
 / is shifted N, \ is shifted M  
 3 COMMODORE @s, shifted N, 2 spaces, shifted M, 3 COMMODORE @s  
 3 COMMODORE Ts, shifted M, 2 spaces shifted N, 3 COMMODORE Ts

For a birthday card we could write a greeting.

### Program 10-7 Birthday Card

```

NEW
100 PRINT CHR$(147)
110 INPUT "WHAT IS YOUR NAME";N$
120 INPUT "WHAT IS YOUR FRIEND'S NAME";M$
130 PRINT CHR$(147)
140 FOR R=1 TO 11
150 PRINT
160 NEXT R
170 PRINT TAB(12)"HAPPY BIRTHDAY"
180 PRINT
190 PRINT TAB(15)"TO: ";M$
200 PRINT
210 PRINT TAB(15)"FROM: ";N$
RUN

```

For Halloween, we could draw a pumpkin and print both our name and a friend's name on the card.

### Program 10-8 Halloween Card

```

NEW
100 PRINT CHR$(147)
110 INPUT "WHAT IS YOUR NAME";N$
120 INPUT "WHAT IS YOUR FRIEND'S NAME";M$
130 PRINT CHR$(147)
140 PRINT "DEAR ";M$
150 PRINT
160 PRINT " " ← 3 spaces, COMMODORE I
170 PRINT " " ← Shifted (U, C, C), COMMODORE U, shifted (C, C, I)
180 PRINT " " ← Shifted (B, Q), 3 spaces, shifted (Q, H)
190 PRINT " " ← Shifted B, 5 spaces, shifted H
200 PRINT " " ← Shifted B, 2 spaces, shifted Z, 2 spaces, shifted H
210 PRINT " " ← Shifted B, space, 3 COMMODORE £s, space, shifted H
220 PRINT " " ← Shifted J, 5 shifted Rs, shifted K
230 PRINT
240 PRINT "HAPPY HALLOWEEN"
250 PRINT "FROM YOUR FRIEND"
260 PRINT N$
RUN

```

A Halloween card can have a picture of a black cat. The picture looks like

COMMODORE M, shifted M, 3 spaces, shifted N, COMMODORE G  
 COMMODORE M, space, 3 COMMODORE Ts, space, COMMODORE G  
 Shifted N, space, \*, space, \*, space, shifted M  
 Shifted M, space, COMMODORE £, shifted Q, COMMODORE £, space, shifted N  
 COMMODORE M, 5 spaces, COMMODORE G  
 Shifted N, 5 spaces, shifted M  
 Shifted M, 5 spaces, shifted N  
 Shifted N, 5 spaces, shifted M  
 Shifted M, 5 spaces, shifted N  
 Space, shifted P, COMMODORE M, shifted P, COMMODORE M, COMMODORE T  
 2 spaces, COMMODORE +, space, COMMODORE +



## Program 10-9 Halloween Cat

```

NEW
100 PRINT CHR$(147)
110 INPUT "WHAT IS YOUR NAME";N$
120 PRINT CHR$(147)
130 PRINT "  /-----\  "
140 PRINT " |         |  "
150 PRINT " | * * *   |  "
160 PRINT " |  *  *   |  "
170 PRINT " |         |  "
180 PRINT " |         |  "
190 PRINT " |         |  "
200 PRINT " |         |  "
210 PRINT " |         |  "
220 PRINT " |  _ _ _  |  "
230 PRINT " |  *  *  |  "
240 PRINT "BOO!!!!"
250 PRINT
260 PRINT "FROM YOUR FRIEND, ";N$
RUN

```

For Mother's Day, we could create a screen full of I LOVE YOU's.

## Program 10-10 Mother's Day Card

```

NEW
100 PRINT CHR$(147)
110 INPUT "WHAT IS YOUR NAME";N$
120 PRINT CHR$(147)
130 PRINT "HAPPY MOTHER'S DAY, MOM"
140 PRINT
150 FOR J=1 TO 50
160 PRINT "-I LOVE YOU-"; ← Be sure to put the semicolon here.
170 NEXT J
180 PRINT
190 PRINT
200 PRINT "FROM YOUR DAUGHTER, ";N$
RUN

```

For Easter we could draw a giant cross.

## Program 10-11 Easter Card

```

NEW
100 PRINT CHR$(147)
110 INPUT "WHAT IS YOUR NAME";N$
120 PRINT CHR$(147)
130 PRINT "  " ← Shifted O, COMMODORE T, shifted P
140 PRINT "  " ← COMMODORE G, space, COMMODORE M
150 PRINT "  " ← COMMODORE G, space, COMMODORE M
160 PRINT "  " ← COMMODORE (3 @s, G), space, COMMODORE (M, 3 @s)
170 PRINT " | PEACE | " ← COMMODORE G, space, PEACE, space, COMMODORE M
180 PRINT " |     | " ← COMMODORE (3Ts, G), space, COMMODORE (M, 3Ts)
190 PRINT " |     | " ← COMMODORE G, space, COMMODORE M
200 PRINT " |     | " ← COMMODORE G, space, COMMODORE M
210 PRINT " |     | " ← COMMODORE G, space, COMMODORE M
220 PRINT " |     | " ← COMMODORE G, space, COMMODORE M
230 PRINT " |     | " ← Shifted L, COMMODORE @, shifted @
240 PRINT
250 PRINT "HAPPY EASTER"
260 PRINT
270 PRINT "FROM: ";N$
RUN

```

What would Easter be like without the Easter bunny?

### Program 10-12 Easter Bunny

```
NEW
100 PRINT CHR$(147)
110 INPUT "WHAT IS YOUR NAME";N$
120 PRINT CHR$(147)
130 PRINT "  ^  ^  ^  " ← Shifted (N, M), space, shifted (N, M)
140 PRINT " | | | | " ← COMMODORE (G, M), space, COMMODORE (G, M)
150 PRINT " | | | | " ← COMMODORE (G, M), space, COMMODORE (G, M)
160 PRINT "  ^  ^  ^  " ← Shifted (M, N), COMMODORE T, shifted (M, N)
170 PRINT "  ^  ^  ^  " ← Shifted (N, F), space, shifted (F, M)
180 PRINT " | * * | " ← COMMODORE G, *, space, *, COMMODORE M
190 PRINT " = * * = " ← Shifted M, =, shifted Q, =, shifted N
200 PRINT "  ^  ^  ^  " ← Shifted N, 3 spaces, shifted M
210 PRINT " | | | | " ← COMMODORE G, 3 spaces, COMMODORE M
220 PRINT "  ^  ^  ^  " ← Shifted M, 3 spaces, shifted N
230 PRINT "  O _ O  " ← (, ), shifted F, (, )
240 PRINT
250 PRINT "HAPPY EASTER"
260 PRINT
270 PRINT "FROM:   ";N$
RUN
```

# 11

## THE PERSONAL TOUCH

We can also use INPUT to personalize stories and letters, make up forms and notices, and ask questions. For example, we can write a simple personalized story.

### **Program 11-1 Personalized Story**

```
NEW
100 PRINT CHR$(147)
110 INPUT "ENTER YOUR NAME";N$
120 PRINT CHR$(147)
125 REM CENTER STORY ON SCREEN
130 FOR J=1 TO 5
140 PRINT
150 NEXT J
155 REM TITLE STORY
160 PRINT "A STORY ABOUT";N$
170 PRINT
180 PRINT
190 PRINT "SEE DICK RUN! SEE JANE RUN!"
200 PRINT "SEE SPOT RUN! SEE ";N$;" RUN!"
210 PRINT "WATCH OUT, ";N$;", HERE COMES"
220 PRINT "THE MAN WHOSE WINDOW YOU BROKE!"
230 PRINT "RUN FASTER, ";N$;"!"
RUN
```

We could replace lines 130–150 with  
130 PRINT "{5 DOWN}"

The computer prints only 40 characters on a line. It then continues on the next line, starting with the forty-first character. Thus, if you enter a long line, the computer will break it in an arbitrary place, making the output look strange. To keep this from happening, restrict all your lines to a maximum of 40 characters. Even then, the right-hand border will be uneven.

If you write your own story, be careful where you put the quotation marks. Look at lines 200, 210, and 230 as typical examples. All background words such as RUN, WATCH OUT, HERE COMES, and RUN FASTER must have quotation marks around

them. So also must punctuation such as commas, exclamation marks, and periods, as well as extra spaces that separate the name from the background words. The name N\$, however, must not be inside quotation marks—this is difficult to see in lines with many quotation marks, so always balance marks from left to right. Here's another story about N\$ for you to try:

```
190 PRINT "LONG AGO, IN THE FARAWAY LAND OF OG,"
200 PRINT "THERE LIVED THE GALLANT ";N$;"."
210 PRINT "ONE DAY, ";N$;" WENT OUT TO SLAY"
220 PRINT "A DRAGON. UNFORTUNATELY, THE DRAGON"
230 PRINT "SLEW ";N$;" INSTEAD. TOO BAD,"
240 PRINT N$;" , BETTER LUCK NEXT TIME!"
```

We can use many INPUTs to obtain information. The computer can then produce a customized letter or notice.

### ***Program 11-2 Homework Assignment Notice***

```
NEW
100 PRINT CHR$(147)
110 INPUT "ENTER YOUR NAME";N$
120 INPUT "ENTER SUBJECT";S$
130 INPUT "ENTER CHAPTER NUMBER";C
140 PRINT CHR$(147)
150 FOR J=1 TO 10
160 PRINT
170 NEXT J
180 PRINT "HOMEWORK ASSIGNMENT"
190 PRINT
200 PRINT "INSTRUCTOR: ";N$
210 PRINT
220 PRINT "SUBJECT: ";S$
230 PRINT
240 PRINT "CHAPTER: ";C
RUN
```

As with greeting cards, you can eliminate READY and the blinking cursor from the notice by concluding the program with an endless loop:

```
250 GOTO 250
```

This program allows reasonably long entries. The only time it will break up a line is when the subject is something like CHEMICAL ENGINEERING—PRINCIPLES AND PRACTICES or ADVANCED UNDERWATER BASKET WEAVING. You may add blank lines, TABs, borders, or drawings to make the notice more readable and more attractive. The following program produces borders using asterisks.

### Program 11-3 Legal Notice With Three Names

```
NEW
100 PRINT CHR$(147)
110 INPUT "ENTER PLAINTIFF'S NAME";P$
120 INPUT "ENTER DEFENDANT'S NAME";D$
130 INPUT "ENTER JUDGE'S NAME";J$
140 PRINT CHR$(147)
150 PRINT "*****" ← 20 asterisks. Line 320 is
160 PRINT                                     the same.
170 PRINT "OFFICIAL PUBLIC NOTICE"
180 PRINT
190 PRINT "NEXT CASE IS:"
200 PRINT
210 PRINT P$;"", PLAINTIFF"
220 PRINT
230 PRINT "VERSUS"
240 PRINT
250 PRINT D$;"", DEFENDANT"
260 PRINT
270 PRINT "BEFORE JUDGE ";J$
280 PRINT
290 PRINT "*****" ← 20 asterisks
RUN
```

Note that we can call the answers anything, although we have generally meaningful names, such as P\$ for plaintiff, D\$ for defendant, and J\$ for judge. Of course, we can use INPUT to enter any characters, not just letters. For example, in response to

```
INPUT "ENTER YOUR ADDRESS";A$
```

we can enter letters, numbers, hyphens, # signs, periods, etc. However, if the lines are

```
INPUT "ENTER YOUR AGE";A
```

then we can only enter a number.

A\$ holds typed characters!

A holds only a number!

If you accidentally enter something other than a number, such as TWENTY-SEVEN instead of 27 for your age, the computer will scold you by printing

```
?REDD FROM START
```

However, once the scolding is over, you may enter the proper answer without further complication.

The Commodore 64 can serve as an inexpensive version of the elaborate message boards you often see in airports, stadiums, convention centers, and large hotels. The following program, for example, provides a news bulletin service.

## **Program 11-4 Extra! Extra! Read All About It!**

```
NEW
100 PRINT CHR$(147)
110 INPUT "ENTER DATELINE";D$
120 INPUT "ENTER LINE 1";L1$
130 INPUT "ENTER LINE 2";L2$
140 INPUT "ENTER LINE 3";L3$
150 INPUT "ENTER LINE 4";L4$
160 PRINT CHR$(147)
170 PRINT COMMODORE 64 COMPUTER NEWS BULLETIN"
180 PRINT {3 DOWN}DATELINE: ";D$
190 PRINT {3 DOWN}";L1$
200 PRINT {3 DOWN}";L2$
210 PRINT {3 DOWN}";L3$
220 PRINT {3 DOWN}";L4$
RUN
```

You might try a story like

DATELINE: WASHINGTON, D.C.  
MARTIANS INVADE CITY AT NOON!  
TAKE OVER CONGRESS, WHITE HOUSE,  
AND SUPREME COURT! SURPRISE ATTACK  
LEAVES MILITARY AND POLICE STUNNED!

or

DATELINE: WASHINGTON, D.C.  
MARTIANS SURRENDER UNCONDITIONALLY!  
MANY DIE OF BOREDOM LISTENING TO  
CONGRESSIONAL SPEECHES! OTHERS FORCED  
TO SERVE ON PRESIDENTIAL COMMISSIONS!

or

DATELINE: BOSTON, MA  
TEA PARTY SCHEDULED AT NOON AT HARBOR!  
PLEASE COME IN PERIOD COSTUME  
NO ENGLISH INVITED!  
RSVP: SAMUAL ADAMS, ESQUIRE.

There are a few things you must watch here (besides political and national sensitivities). In the first place, if your text includes commas, colons, or leading spaces that you want included, you will have to enclose them in quotation marks. For example, to enter WASHINGTON, D.C., you must type "WASHINGTON, D.C." If you forget the quotation marks, the computer will drop the comma and everything after it and report

?EXTRA IGNORED

What happens if you reverse the characters in line 170? That is, after typing the quotation mark, press CTRL and 9 (RVS ON). We indicate this in shorthand as {RVS} and CTRL and 0 (RVS OFF) as {OFF}. Line 170 thus should be

```
170 PRINT "{RVS}COMMODORE 64 COMPUTER NEWS BULLETIN"
```

Reversed characters are highly noticeable and easy to read. Try putting {RVS} in the news bulletin lines (180 through 220). Note that a RETURN turns {RVS} off automatically; you do not need to put {OFF} at the end of a line.

You could use this same approach to create a computerized bulletin board for a store, classroom, or clubhouse.

We can also use INPUT to write a program that asks a series of questions. The computer can then serve as an interviewer for hospitals, clinics, government agencies, insurance offices, and other businesses or institutions that must collect vital statistics.

### **Program 11-5 Computer Interviewer**

```
NEW
100 PRINT CHR$(147)
110 INPUT "FIRST NAME";F$
120 INPUT "MIDDLE INITIAL";M$
130 INPUT "LAST NAME";L$
140 INPUT "STREET ADDRESS";A$
150 INPUT "CITY";C$
160 INPUT "STATE";S$
170 INPUT "ZIP CODE";Z$
180 PRINT CHR$(147)
190 PRINT TAB(5)"CLIENT'S VITAL STATISTICS"
200 PRINT "{3 DOWN}NAME: ";L$;" ", "F$;" "M$;" ". "
210 PRINT "{3 DOWN}STREET ADDRESS: ";A$
220 PRINT "{3 DOWN}CITY/STATE/ZIP: ";C$;" ", "S$;" " ";Z$
RUN
```

Beware of addresses that contain a comma, such as "2150 MAIN ST., APT. 35". You must put quotation marks around them. If you are the kind of person who carries an umbrella "just in case," you may want to put quotation marks around all text entries.

In practice, many people actually prefer a computer interviewer over a human interviewer, much as many people prefer to use an automatic teller machine even when a bank is open. While the computer lacks warmth and sympathy, it also lacks human prejudices. It does not notice that a person has a strange appearance, is handicapped or disfigured, has language problems, or is slow to respond to questions.

On the other hand, computer interviewers are quite naive. If you tell the computer your name is PUSS N BOOTS and your address is TOAD HALL, EMERALD CITY, OZ, 55555, it will simply accept your response.

Besides acting as interviewers, computers of the future will surely provide a wide range of expert assistance. While the following programs are humorous, they show the computer's inherent capabilities. Can you tell from the questions and answers alone whether you are talking to a computer or a person?

## Program 11-6 Doctor's Assistant

```
NEW
100 PRINT CHR$(147)
110 PRINT "DR. SMITH IS NOT IN AT THIS TIME. I AM"
120 PRINT "HIS COMPUTER ASSISTANT, READY TO HELP"
130 PRINT
140 INPUT "WHAT IS YOUR NAME ";N$
150 INPUT "WHAT IS YOUR PROBLEM ";P$
160 PRINT CHR$(147)
170 PRINT "DEAR ";N$;" : "
180 PRINT
190 PRINT "I AM SORRY THAT YOU HAVE ";P$
200 PRINT "GO TO BED EARLY, DRINK PLENTY OF"
210 PRINT "LIQUIDS, TAKE TWO ASPIRIN, AND CALL"
220 PRINT "DR. SMITH IN THE MORNING. I HOPE"
230 PRINT "YOU FEEL BETTER SOON."
240 PRINT
250 PRINT TAB(10)"SINCERELY, YOUR FRIEND,"
260 PRINT
270 PRINT TAB(12)"THE COMPUTER"
280 GOTO 280 ← An endless loop. You must press
                STOP to regain control.
RUN
```

This simple-minded assistant does surprisingly well if your problem is A COLD, A SORE THROAT, or A HEADACHE. It runs into grammatical and medical difficulties if your problem is BROKEN ARM, A NAIL IN MY FOOT, or A COMPLAINT ABOUT MY BILL.

## Program 11-7 Tax Collector's Assistant

```
NEW
100 PRINT CHR$(147)
110 PRINT "I AM YOUR FRIENDLY TAX COLLECTOR'S"
120 PRINT "ASSISTANT. WE ARE HERE TO SERVE YOU."
130 PRINT
140 INPUT "WHAT IS YOUR NAME";N$
150 INPUT "HOW MUCH DID YOU EARN LAST YEAR";EARN
160 PRINT "REALLY, NOW, HOW MUCH DID YOU EARN"
170 INPUT "LAST YEAR";EARN
180 PRINT CHR$(147)
190 PRINT "YOU CANNOT FOOL ME, ";N$;" !"
200 PRINT "I KNOW YOU MADE MORE THAN $";EARN
210 PRINT "LAST YEAR! PAY YOUR TAX OR GO"
220 PRINT "TO JAIL! HAVE A NICE DAY, ";N$;". "
RUN
```

Note that we had to split a long question (lines 160 and 170) into two statements, a PRINT and an INPUT.

Programs 11-4 through 11-7 let the computer ask questions, read the answers, and repeat them like a parrot. However, it cannot determine if the answers are reasonable or correct. We need another BASIC statement that lets the computer compare things and decide what to do on the basis of the comparison. This statement is IF THEN. We call the basis for the computer's decision a *condition*. The condition follows IF, and following



THEN is the statement to do next if the condition is true. Typical examples of IF THEN are

```
IF YEAR=1776 THEN PRINT "THAT IS THE CORRECT YEAR"  
IF EXPLORER$="COLUMBUS" THEN GOTO 1200  
IF N$=KING$ THEN GOTO 450  
IF TEMP=FREEZING THEN PRINT "SOLUTION NOW FROZEN"
```

The following simple exercise shows how IF THEN works.

### **Program 11-8 Simple Arithmetic Quiz**

```
NEW  
100 PRINT CHR$(147)  
110 INPUT "WHAT IS 2+2";A  
120 IF A=4 THEN GOTO 140  
130 PRINT "SORRY, ANSWER IS 4"  
140 END  
RUN
```

GOTO is optional here. 140 by itself is equivalent to "GOTO 140".  
END does nothing except terminate a program.

After you type RUN, you will see the question

```
WHAT IS 2+2?
```

Answer by entering 4 and pressing RETURN. The computer does not print anything because  $A=4$  (the condition) in line 120 is true, thus causing it to do line 140 next. We call this departure from numerical order a *branch*.

Now type RUN again. This time enter 5 as your answer. You will see the message

```
SORRY, ANSWER IS 4
```

on the screen. Since at line 120,  $A=4$  is not true, the computer ignores THEN and proceeds normally to line 130.

Be careful when you type +. If you press SHIFT and +, you will get a graphics character that looks like a giant plus sign. The computer will not, however, accept this character as meaning "add."

A quiz with several questions requires a series of INPUT statements and IF THENs to check the answers.

### **Program 11-9 Arithmetic Quiz With Two Questions**

```
NEW
100 PRINT CHR$(147)
110 INPUT "WHAT IS 5+5";A
120 IF A=10 THEN 200
130 PRINT "WRONG, THE ANSWER IS 10"
200 INPUT "WHAT IS 3X4";B
210 IF B=12 THEN 300
220 PRINT "WRONG, THE ANSWER IS 12"
300 END
RUN
```

Remember, GOTO is optional here.

In line 110, the first INPUT, the computer calls the answer A. It then checks A against the correct answer. If A is not 10, the computer goes to the next line (line 130) and prints an error message. At line 200, the computer asks another question. Everything is the same, except that the answer is B instead of A.

Now let's try questions where the answer is a name instead of a number.

### **Program 11-10 Geography Quiz With an Alphabetic Question**

```
NEW
100 PRINT CHR$(147)
110 INPUT "WHAT IS THE CAPITAL OF GEORGIA";C$
120 IF C$="ATLANTA" THEN 140
130 PRINT "WRONG, THE ANSWER IS ATLANTA"
140 END
RUN
```

This time the answer is C\$, because it is text rather than a number. We can similarly write a program that asks several questions.

### **Program 11-11 Geography Quiz With Several Questions**

```
NEW
100 PRINT CHR$(147)
110 INPUT "WHERE IS WARSAW";C$
120 IF C$="POLAND" THEN 200
130 PRINT "SORRY, THE ANSWER IS POLAND."
200 INPUT "WHERE IS ROME";C$
210 IF C$="ITALY" THEN 300
220 PRINT "SORRY, THE ANSWER IS ITALY."
300 INPUT "WHAT IS THE CAPITAL OF FRANCE";C$
310 IF C$="PARIS" THEN 400
320 PRINT "SORRY, THE ANSWER IS PARIS."
400 END
RUN
```

Note that the computer will not excuse spelling or typing errors. You must, for example, answer POLAND when the computer asks "WHERE IS WARSAW?" POELAND, POLABD, or @OLAND will not do. Also, if you simply press RETURN without entering anything at all, the computer proceeds as if you had made a mistake.

# 12

## CONTROLLING ACTION FROM THE KEYBOARD

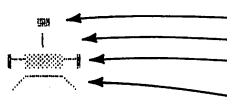
Program 7-7 drew a simple lunar station and had it fire a projectile. All we did was type RUN; we could not control when the projectile was fired. To control the firing from the keyboard (as in video games), we need the new BASIC statement GET. The following lines make the computer stop and wait for you to press a key.

```
300 GET K$
310 IF K#="" THEN 300
```

No space inside. Just type 2 quotation marks in a row.

Note that we can omit the GOTO after THEN.

We can easily revise Program 7-7 so that the player, not the computer, decides when to fire. We will draw a fancier lunar station with graphics symbols; it looks like



- 3 spaces, COMMODORE I, 3 spaces
- 3 spaces, shifted B, 3 spaces
- COMMODORE L, shifted C, 3 COMMODORE +s, shifted C, COMMODORE J
- Space, shifted N, 3 COMMODORE Ys, shifted M, space

We start with

```

NEW
100 PRINT CHR$(147)
105 REM MOVE DOWN 21 LINES
110 FOR J=1 TO 21
120 PRINT
130 NEXT J
135 REM MOVE RIGHT TO COLUMN 17
140 PRINT TAB(16);
150 PRINT "  ☐ ";
170 PRINT "  | ";
190 PRINT "  |▒▒▒▒| ";
210 PRINT "  ▽ ";
160 PRINT "{7 LEFT}{DOWN}"; ← Lines 180 and 200 are the same.
180 PRINT "{7 LEFT}{DOWN}";
200 PRINT "{7 LEFT}{DOWN}";
220 PRINT "{4 LEFT}{4 UP}"; ← Move to the top middle of the station
                             to start the missile.
230 FOR J=1 TO 20
235 REM FIRE MISSILE
240 PRINT "◆"; ← ◆ is shifted Z.
250 PRINT "{LEFT}";
260 FOR L=1 TO 100:NEXT L ← L rather than K to avoid confusion
                             with K$.
265 REM ERASE OLD MISSILE
270 PRINT " "; ← Put 1 space here.
280 PRINT "{LEFT}{UP}"; ← Move missile up a line.
290 NEXT J
300 PRINT CHR$(147)
RUN

```

When you type RUN, the lunar station fires a missile automatically. To make it wait for you to tell it to fire, add the lines

```

222 REM WAIT FOR PLAYER TO PRESS ANY KEY
224 GET K$
226 IF K$="" THEN 224

```

No space here, type 2 quotation marks in a row. This means "Go to line 224 if K\$ is nothing—the so-called null string."

Program 12-1 includes this addition.

### Program 12-1 Lunar Station Firing Missile, Under Keyboard Control

```

NEW
100 PRINT CHR$(147)
105 REM MOVE DOWN 21 LINES
110 FOR J=1 TO 21
120 PRINT
130 NEXT J
135 REM MOVE RIGHT TO COLUMN 17
140 PRINT TAB(16);
150 PRINT "  ☐ "; ← 3 spaces, COMMODORE I, 3 spaces
170 PRINT "  | "; ← 3 spaces, shifted B, 3 spaces
190 PRINT "  |▒▒▒▒| "; ← COMMODORE L, shifted C, 3 COMMODORE + s, shifted C,
210 PRINT "  ▽ "; ← COMMODORE J
160 PRINT "{7 LEFT}{DOWN}"; ← Space, shifted N, 3 COMMODORE Ys,
180 PRINT "{7 LEFT}{DOWN}"; ← shifted M, space
200 PRINT "{7 LEFT}{DOWN}";
220 PRINT "{4 LEFT}{4 UP}"; ← Move to the top middle of the
222 REM WAIT FOR PLAYER TO PRESS ANY KEY station to start the missile.
224 GET K$

```

```

226 IF K$="" THEN 224
230 FOR J=1 TO 20
235 REM FIRE MISSILE
240 PRINT "◆"; ← ◆ is shifted Z
250 PRINT "{LEFT}";
260 FOR L=1 TO 100:NEXT L ← L rather than K to avoid confusion with K$.
265 REM ERASE OLD MISSILE
270 PRINT " "; ← Put 1 space here.
280 PRINT "{LEFT} {UP}"; ← Move missile up a line.
290 NEXT J
300 PRINT CHR$(147)
RUN

```

After you type RUN, you must press a key (say, F for "fire") to make the station fire a missile. If you do not press a key, nothing happens. You now control the firing. Adding the following line lets the station fire repeatedly:

```
300 GOTO 100
```

Now the lunar station fires a missile every time you press F. Of course, you must wait for the old missile to disappear, since the computer moves it all the way to the top before examining the keyboard again. You must press the RUN/STOP key to end the program. You may notice that the station jerks a little after the missile disappears. This occurs because the program actually clears the screen and redraws the station, rather than just moving the cursor back to where the missile starts. The following instructions eliminate the extraneous motion:

```

295 REM RETURN TO WHERE MISSILE STARTS
300 FOR J=1 TO 20
310 PRINT
320 NEXT J
330 PRINT TAB(19);
340 GOTO 224

```

Video game manufacturers use this kind of control in arcade games. Rockets, spaceships, or creatures move across the screen, and you must press a button (or a key) to fire at them.

We will now draw moving targets and make our lunar station fire missiles at them. Let's start with the targets.

### **Program 12-2 Eye Creatures Moving Right**

```

NEW
100 PRINT CHR$(147)
110 FOR C=1 TO 30
120 PRINT CHR$(19)
130 PRINT TAB(C)" <O> (O) "
140 FOR L=1 TO 200:NEXT L ← Slows the computer down.
150 NEXT C
160 PRINT CHR$(147)
RUN

```

The targets move right near the top of the screen. Let us now combine them with Program 12-1. We will make the computer look specifically for the F key rather than just for any key.

## Program 12-3 Target Practice

```

NEW
100 PRINT CHR$(147)
105 REM MOVE DOWN 21 LINES
110 FOR J=1 TO 21
120 PRINT
130 NEXT J
135 REM MOVE RIGHT TO COLUMN 17
140 PRINT TAB(16);
150 PRINT "  I "; ← 3 spaces, COMMODORE I, 3 spaces
170 PRINT "  | "; ← 3 spaces, shifted B, 3 spaces
190 PRINT "  | "; ← COMMODORE L, shifted C, 3 COMMODORE +s, shifted C,
210 PRINT "  | "; ← COMMODORE J
160 PRINT "{7 LEFT}{DOWN}"; ← Space, shifted N, 3 COMMODORE Ys,
180 PRINT "{7 LEFT}{DOWN}"; ← shifted M, space
200 PRINT "{7 LEFT}{DOWN}";
215 REM DRAW MOVING CREATURES
220 FOR C=1 TO 30
230 PRINT CHR$(19)
240 PRINT TAB(C)" <O> (O) "
245 REM PRESS F KEY TO FIRE MISSILE
250 GET K$
260 IF K$="F" THEN 300
270 FOR L=1 TO 100:NEXT L
280 NEXT C
290 GOTO 100
295 REM MOVE DOWN TO WHERE MISSILE STARTS
300 FOR J=1 TO 18
310 PRINT
320 NEXT J
330 PRINT TAB(19);
340 FOR J=1 TO 20
350 PRINT "◆"; ← ◆ is shifted Z.
360 PRINT "{LEFT}";
370 FOR L=1 TO 100:NEXT L
380 PRINT " "; ← Put 1 space here.
390 PRINT "{LEFT}{UP}"; ← Move missile up a line.
400 NEXT J
410 PRINT CHR$(147)
RUN

```

We place GET K\$ inside the moving creature instructions:

```

FOR C=1 TO 30
PRINT CHR$(19)
PRINT TAB(C)" <O> (O) " ← Moves the creatures.
NEXT C

```

If the player does not press the F, the computer keeps moving the eye creatures. If the player presses F, the computer goes to line 300 and fires a missile. The line

```
260 IF K$="F" THEN 300
```

checks the keyboard without stopping the program. Note, however, that as soon as you press F, the creatures stop, since the computer has left the part of the program that moved them.

To speed the creatures up and make it more difficult to hit them, change line 270 to

```
270 FOR L=1 TO 50:NEXT L
```

Don't forget to press F to fire a missile.

The lunar station jerks a little each time the creatures move across the screen, just as it did when we made Program 12-1 run continuously. This is because line 290 redraws the station rather than just restarting the creatures (and erasing them from the far right). The following lines eliminate the problem:

```
288 REM ERASE CREATURES FROM RIGHT
290 PRINT CHR$(19)
291 PRINT TAB(30);" " ← 9 spaces
292 REM NOW MOVE THEM AGAIN
293 GOTO 220
```

# 13

## MAKING NOISE

Arcade games play music and make noise. We can do the same things with the Commodore 64. Of course, producing sound effects and music on a computer is just as complex as producing them with musical instruments. We will not delve into musical theory here; rather, we will describe the computer's sound-making capabilities in just enough detail so that you can experiment with them.

To produce sound, the computer must know:

1. What volume you want. You can set the volume with

POKE 54296,V

where V is a number between 0 and 15; 15 is the loudest level, 0 the quietest. We will generally just set V to 15 and control volume by adjusting the television set. However, you could, for example, change this level to make a bomb fall with a moderate whine, then land with an especially loud explosion.

2. What notes you want to play. You can select a particular note using values from Appendix A in

POKE 54273, HF ← High frequency  
POKE 54272, LF ← Low frequency

For example, to play an A note in Octave 3, you would need

POKE 54273, 14 ← High frequency  
POKE 54272, 24 ← Low frequency



Be careful; the high-frequency value is always smaller than the low-frequency value, but the first number in its POKE is larger (54273 as compared to 54272).

3. How long you want a note to last. You can make a note last a specific amount of time with the do-nothing loop

```
FOR L=1 TO D:NEXT L
```

D should be 250 for a quarter second or quarter note, 500 for a half-second or half-note, and 1000 for a whole second or whole note.

4. How fast you want the note to rise and fall. You can specify the rise time (called *attack*) as any number from 0 to 15; 0 is the shortest rise time, 15 the longest. You can specify the fall time (called *decay*) similarly. To put these values into the computer, you must use

```
POKE 54277,16*ATTACK+DECAY
```

Thus, to obtain a medium attack or rise time (8) and a medium decay or fall time (also 8), we would use

```
POKE 54277,16*8+8 or POKE 54277,136
```

Similarly, to obtain a maximum attack (15) and a minimum decay (0), we would use

```
POKE 54277,16*15+0 or POKE 54277,240
```

To obtain a minimum attack (0) and a maximum decay (15), we would use

```
POKE 54277,16*0+15 or POKE 54277,15
```

You can simply enter the attack and decay values and let the computer do the multiplication and addition.

5. What kind of waveform you want. We select this with

```
POKE 54276,WV+1
```

where WV is 16 for triangular form (#1), 32 for sawtooth form (#2), 64 for rectangular form (#3), and 128 for random (or *white*) noise (#4). For example, to select a triangular form (#1), we would use

```
POKE 54276,17
```

Besides determining the kind of sound we want, we must also perform some overhead functions. These are:

1. Clearing the computer's sound-producing device (called the SID chip), using

```
FOR J=54272 TO 54296
POKE J,0
NEXT J
```

2. Stopping the production of the note, using

```
POKE 54276,WV
```

where WV is the waveform value described earlier (16 for sawtooth, 32 for triangular wave, 64 for rectangular wave, and 128 for noise).

Let us produce a simple sound with one note. We will use a D from Octave 3 (high frequency = 9, low frequency = 104).

### ***Program 13-1 Single Note***

Be sure to turn the volume up on your television set.

```
NEW
100 PRINT CHR$(147)
105 REM CLEAR SOUND CHIP
110 FOR J=54272 TO 54296
120 POKE J,0
130 NEXT J
135 REM SET VOLUME TO LOUDEST
140 POKE 54296,15
145 REM SET ATTACK AND DECAY TO MIDDLE VALUES
150 POKE 54277,8*16+B
155 REM MAKE WAVEFORM TRIANGULAR
160 POKE 54276,17
165 REM PRODUCE D IN OCTAVE 3
170 POKE 54273,9
180 POKE 54272,104
185 REM HAVE NOTE PLAY FOR ABOUT 1 SECOND
190 FOR L=1 TO 1000:NEXT L
195 REM STOP WAVEFORM
200 POKE 54276,16
RUN
```

Run this several times to hear how it sounds. The result is like someone tuning a cello or bass fiddle.

Let us now experiment with the waveforms. Change lines 160 and 200 to

```
160 POKE 54276,33
200 POKE 54276,32
```

The new sound is harsher and less muted. Try

```
160 POKE 54276,65
200 POKE 54276,64
```

This sounds like someone tapping on a piece of leather. Finally, try

160 POKE 54276,129  
200 POKE 54276,128

The noise waveform sounds like a pile driver or wooden blocks colliding. Noise can produce sound effects to accompany explosions, rocket launches, and automobile crashes.

Let us now see what happens if we vary attack and decay. Leaving the noise waveform, try a maximum attack and minimum decay:

150 POKE 54277,16\*15+0

The sound rises slowly and then ends abruptly. It sounds like a gas flare or an acetylene welding torch. Now try minimum attack and maximum decay:

150 POKE 54277,16\*0+15

the sound rises quickly and dies slowly. It sounds like the noise from a jet engine.

Go back to the original waveform and try minimum attack and decay, that is

150 POKE 54277,16\*0+0  
160 POKE 54276,17  
170 POKE 54276,16

The note rises and falls rapidly, like a quick click. It hardly sounds like anything at all. Now try maximum attack and minimum decay, that is,

150 POKE 54277,16\*15+0

The note rises slowly to its maximum, then stops. It sounds like someone blowing slowly into a pipe or reed instrument.

Finally, try minimum attack and maximum decay, that is,

150 POKE 54277,16\*0+15

Now the note rises quickly, then falls off slowly, like a foghorn.

## Program 13-2 Rocket Sound Using White Noise

Keep the volume fairly low or the rocket will blast your ears off.

```
NEW
100 PRINT CHR$(147)
105 REM CLEAR SOUND CHIP AND SET VOLUME
110 FOR J=54272 TO 54296
120 POKE J,0
130 NEXT J
140 POKE 54296,15
150 FOR J=1 TO 10
155 REM PRODUCE TWO NOTES
158 REM START FIRST NOISE
160 POKE 54277,0*16+10 ← Set minimum attack, medium decay.
170 POKE 54276,129 ← Start noise waveform.
180 POKE 54273,9
190 POKE 54272,104
200 FOR L=1 TO 50:NEXT L
210 POKE 54276,128 ← Stop noise waveform.
215 REM START SECOND NOISE
220 POKE 54277,0*16+10 ← Set minimum attack, medium decay.
230 POKE 54276,129 ← Start noise waveform.
240 POKE 54273,10
250 POKE 54272,143
260 FOR L=1 TO 50:NEXT L
270 POKE 54276,128 ← Stop noise waveform.
280 NEXT J
RUN
```

Let us now have sound accompany a picture. We will produce the sound of a lift-off, then make a rocket appear and fly under the power of its engines. The sound changes from lift-off to normal engine noise before the rocket comes into view.

## Program 13-3 Rocket Launch With Sound

```
NEW
100 PRINT CHR$(147)
105 REM CLEAR SOUND CHIP
110 FOR J=54272 TO 54296
120 POKE J,0
130 NEXT J
135 REM MAKE LIFTOFF SOUND
140 POKE 54296,15
150 FOR J=1 TO 23 ← Move cursor to bottom of screen.
160 PRINT
170 NEXT J
175 REM MAKE BLASTOFF SOUND
180 FOR K=1 TO 40
185 REM MINIMUM ATTACK, MEDIUM DECAY
190 POKE 54277,16*0+10
195 REM WHITE NOISE
200 POKE 54276,129
205 REM PRODUCE NOISE
210 POKE 54273,7
220 POKE 54272,12
230 NEXT K
235 REM STOP WAVEFORM
240 POKE 54276,128
245 REM REDUCE VOLUME
250 POKE 54296,7
255 REM PRODUCE ENGINE NOISE
260 GOSUB 1000
```

```

265 REM DRAW ROCKET SHIP
270 PRINT "      ^" " ← 3 spaces, shifted N, shifted M
280 PRINT "     / \ "
290 PRINT "    /   \ "
300 PRINT " | MARS | " ← Space, COMMODORE G, MARS, COMMODORE M
310 PRINT " | SHIP | "
320 PRINT " |     | "
330 PRINT " |     | "
340 PRINT " |     | " ← Space, COMMODORE G, 4 spaces, COMMODORE M
350 PRINT " |     | "
360 PRINT " |     | " ← Space, shifted L, 4 COMMODORE @s, shifted @
370 PRINT " |  :  | " ← 2 spaces, COMMODORE (L, J, L, J)
380 PRINT " |  :  | "
390 PRINT " | : : : | " ← 2 spaces, 4 shifted Qs
395 REM LAUNCH ROCKET, PUT SMOKE UNDER ENGINES
400 FOR J=1 TO 23
410 PRINT " | : : : | " ← 2 spaces, 4 COMMODORE +s
415 REM MAKE ENGINE NOISE
420 GOSUB 1000
430 NEXT J
440 GOTO 9000

995 REM MAKE ENGINE NOISE
998 REM MINIMUM ATTACK, MEDIUM DECAY
1000 POKE 54277,16*0+10
1005 REM WHITE NOISE
1010 POKE 54276,129
1015 REM NOTE #1
1020 POKE 54273,9
1030 POKE 54272,104
1040 FOR L=1 TO 50:NEXT L
1045 REM TURN OFF NOTE 1
1050 POKE 54276,128
1055 REM MINIMUM ATTACK, MEDIUM DECAY
1060 POKE 54277,16*0+10
1065 REM WHITE NOISE
1070 POKE 54276,129
1075 REM NOTE #2
1080 POKE 54273,10
1090 POKE 54272,143
1100 FOR L=1 TO 50:NEXT L
1105 REM TURN OFF NOTE 2
1110 POKE 54276,128
1120 RETURN
9000 PRINT CHR$(147)
RUN

```

The next program makes strange eye creatures produce weird noises.

### **Program 13-4 Eye Creatures Making Noises**

```
NEW
100 PRINT CHR$(147)
105 REM CLEAR SOUND CHIP
110 FOR J=54272 TO 54296
120 POKE J,0
130 NEXT J
140 POKE 54296,15:REM SET VOLUME
150 FOR J=1 TO 10
160 PRINT CHR$(19)
170 PRINT "(00) <00>"
180 POKE 54277,16*0+8:REM MINIMUM ATTACK, MEDIUM DECAY
190 POKE 54276,33:REM START WAVEFORM 2
200 POKE 54273,9
210 POKE 54272,104
220 FOR L=1 TO 250:NEXT L
230 POKE 54276,32:REM STOP WAVEFORM 2
240 PRINT CHR$(19)
250 PRINT "(--> <-->)"
260 POKE 54277,16*0+8:REM MINIMUM ATTACK, MEDIUM DECAY
270 POKE 54276,17:REM START WAVEFORM 1
280 POKE 54273,4
290 POKE 54272,112
300 FOR L=1 TO 250:NEXT L
310 POKE 54276,16:REM STOP WAVEFORM 1
320 NEXT J
330 PRINT CHR$(147)
RUN
```

Note that Program 13-4 uses two different types of sounds, first waveform #2 and then waveform #1. After the note using waveform #2 finishes, the program stops it and starts waveform #1. In this listing, we use colons followed by REMarks to describe the meaning of key POKE values. A colon allows more than one statement on a line. As usual, you can omit the REMs when entering programs.

Change the waveform in lines 190 and 230 to

```
190 POKE 54276,129
230 POKE 54276,128
```

and rerun the program. the new waveform uses white noise to produce a sound like a hammer or metal crashing against metal. Next, change the attack in lines 180 and 260 to

```
180 POKE 54277,16*8+8
260 POKE 54277,16*8+8
```

Now the sound is like a steam locomotive chugging along. Change the attacks to 12 and rerun the program. This sound is like the scraping of a shovel in a gravel pit or against a sidewalk.

We can start and stop the same waveform, with different notes. The following program moves an alien across the screen, accompanied by a beeping sound; we use waveform #2 and the notes 18, 209 and 22, 96.

### **Program 13-5 Alien With Strange Sound**

```
NEW
100 PRINT CHR$(147)
105 REM CLEAR SOUND CHIP
110 FOR J=54272 TO 54296
120 POKE J,0
130 NEXT J
135 REM SET VOLUME
140 POKE 54296,10
145 REM MOVE ALIEN RIGHT
150 FOR C=1 TO 25
160 PRINT CHR$(19)
170 PRINT TAB(C)" \--/ "
180 PRINT TAB(C)" [OO] "
190 PRINT TAB(C)" /--\ "
200 GOSUB 1000
210 NEXT C
220 GOTO 9000
1000 POKE 54277,8*0+8:REM MINIMUM ATTACK, MEDIUM DECAY
1010 POKE 54276,33:REM START WAVEFORM 2
1020 POKE 54273,18
1030 POKE 54272,209
1040 FOR L=1 TO 50:NEXT L
1050 POKE 54276,32:REM STOP WAVEFORM 2
1060 POKE 54277,8*0+15:REM MINIMUM ATTACK, MAXIMUM DECAY
1070 POKE 54276,33:REM START WAVEFORM 2
1080 POKE 54273,22
1090 POKE 54272,96
1100 FOR L=1 TO 50:NEXT L
1110 POKE 54276,32:STOP WAVEFORM 2
1120 RETURN
9000 PRINT CHR$(147)
RUN
```

If you get tired of watching and listening to this program, you can press STOP, but you must also enter POKE 54276,32 (no line number) to stop the sound.

To make the alien blink as it moves, we add a second drawing with different eyes.

### **Program 13-6 Blinking, Moving Alien Making Noise**

Do not type NEW since we want to retain Program 13-5.

```
210 PRINT CHR$(19)
220 PRINT TAB(C)" \--/ "
230 PRINT TAB(C)" <■■> " ← Shifted Q
240 PRINT TAB(C)" /--\ "
250 GOSUB 1000
260 NEXT C
270 GOTO 9000
RUN
```

The program can produce a different sound for the blink if we use the CRSR keys to change the 1000's to 2000's and employ waveform #1 in the 2000's sound routine.

We can also produce many other "talking" pictures.

## Program 13-7 Police Car With Flashing Lights and Siren

```

NEW
100 PRINT CHR$(147)
105 REM CLEAR SOUND CHIP AND SET VOLUME
110 FOR J=54272 TO 54296
120 POKE J,0
130 NEXT J
140 POKE 54296,15
150 FOR C=1 TO 25
155 REM CAR WITH LIGHTS ON
160 PRINT CHR$(19)
170 PRINT TAB(C) "      " ← 3 spaces, 2 shifted Qs, space
180 PRINT TAB(C) "      " ← 3 spaces, shifted (O, P), space
190 PRINT TAB(C) "      " ← Space, shifted N, 4 COMMODORE Ts,
200 PRINT TAB(C) "      " ← shifted M, COMMODORE F, space
210 GOSUB 1000
215 REM LIGHTS OFF
220 PRINT CHR$(19)
230 PRINT TAB(C) "      " ← Space, 2 COMMODORE Ts, shifted W,
COMMODORE T, shifted W, COMMODORE T, space
240 GOSUB 1000
250 NEXT C
260 GOTO 9000
1000 POKE 54277,16*8+15:REM MEDIUM ATTACK, MAXIMUM DECAY
1010 POKE 54276,17:REM START WAVEFORM 1
1020 POKE 54273,12
1030 POKE 54272,143
1040 POKE 54277,16*12+8:REM HIGH ATTACK, MEDIUM DECAY
1050 POKE 54276,33:REM START WAVEFORM 2
1060 POKE 54273,28
1070 POKE 54272,49
1080 FOR K=1 TO 75:NEXT K
1090 POKE 54276,16:REM STOP WAVEFORM 1
1100 POKE 54276,32:REM STOP WAVEFORM 2
1110 RETURN
9000 PRINT CHR$(147)
RUN

```

Here we stop waveform #1 after starting waveform #2. Once the Commodore 64 generates a sound, it continues even when a second sound starts. This feature lets the computer play more than one note at the same time.

## Program 13-8 Alien Spacecraft With Engine Noise

```

NEW
100 PRINT CHR$(147)
105 REM CLEAR SOUND CHIP AND SET VOLUME
110 FOR J=54272 TO 54296
120 POKE J,0
130 NEXT J
140 POKE 54296,15
150 FOR K=1 TO 25 ← Move cursor to bottom of screen.
160 PRINT
170 NEXT K
180 PRINT "      " ← Shifted N, 5 COMMODORE Us, shifted M
190 PRINT " | 00000 |" ← COMMODORE J, 5 shifted Qs, COMMODORE L
200 PRINT "      " ← Shifted M, 5 COMMODORE @s, shifted N
210 PRINT "      " ← Space, shifted (L, @), space, shifted (L, @)
215 REM PRODUCE FIRE FROM ENGINES
220 FOR J=1 TO 25
230 PRINT "      " ← Space, 2 COMMODORE +s, space, 2 COMMODORE +s
240 GOSUB 1000
250 NEXT J
260 GOTO 9000

```



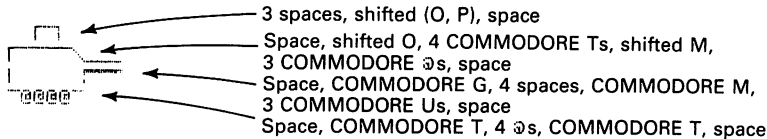
```

1000 POKE 54277,16*0+12:REM MINIMUM ATTACK, HIGH DECAY
1010 POKE 54276,129:REM START NOISE
1020 POKE 54273,9
1030 POKE 54272,104
1040 FOR L=1 TO 50:NEXT L
1050 POKE 54276,128
1060 POKE 54277,16*12+12:REM HIGH ATTACK, HIGH DECAY
1070 POKE 54276,129
1080 POKE 54273,10
1090 POKE 54272,143
1100 FOR L=1 TO 50:NEXT L
1110 POKE 54276,128:REM STOP NOISE
1120 RETURN
9000 PRINT CHR$(147)
RUN

```

Note how we changed the attack in line 1060 to produce a different type of sound in the second note of the engine noise. Adjusting attack and decay values determines how quickly the note rises and falls.

A more complicated scene is a squadron of tanks heading into battle to the accompaniment of ominous background sound. Each tank looks like



The squadron consists of three tanks.

### **Program 13-9 Tank Command With Strange Noises**

```

NEW
100 PRINT CHR$(147)
105 REM CLEAR SOUND CHIP AND SET VOLUME
110 FOR J=54272 TO 54296
120 POKE J,0
130 NEXT J
140 POKE 54296,15
145 REM MOVE TANK COMMAND RIGHT
150 FOR C=1 TO 25
160 PRINT CHR$(19)
170 GOSUB 1000
180 GOSUB 1000
190 GOSUB 1000
200 NEXT C
210 GOTO 9000
990 REM TANK PICTURE
1000 PRINT TAB(C) "
1010 PRINT TAB(C) "
1020 PRINT TAB(C) "
1030 PRINT TAB(C) "
1040 PRINT
1050 POKE 54277,0*16+10:REM MINIMUM ATTACK, MEDIUM DECAY
1060 POKE 54276,17:REM START WAVEFORM 1
1070 POKE 54273,5
1080 POKE 54272,237
1090 FOR L=1 TO 150:NEXT L
1100 POKE 54276,16:REM STOP WAVEFORM 1
1110 RETURN
9000 PRINT CHR$(147)
RUN

```

Diagram annotations for the tank picture:

- Line 1010: Put 1 space on each side of the tank.
- Line 1040: Put a blank line between tanks.

Insert

```
195 GOSUB 1000
```

to add a fourth tank to the squadron. The sound here is like a steady drumbeat in the distance.

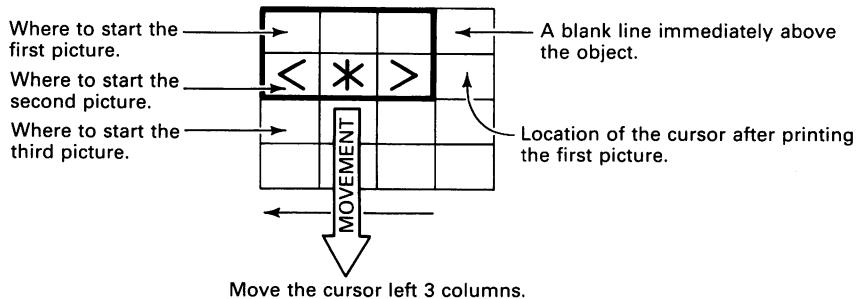
# 14

## FALLING OBJECTS

We have moved things right, left, and up. Now we will move things down. The simple object we start with is

< \* >

Figure 14-1 shows the grid layout for moving it down.

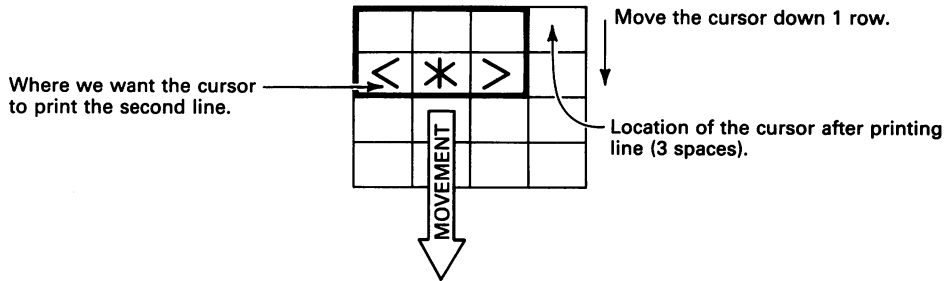


**Figure 14-1** Grid Layout for Falling Asterisk.

A FOR NEXT loop moves the object from row 1 to row 24. We limit the motion to row 24 (rather than 25) to keep the computer from ever printing on the bottom line (25). As we mentioned earlier, the computer automatically moves the entire screen display up when it finishes printing on the bottom line. We must also have the computer print spaces over the previous picture before drawing the current one. Since the object is three columns wide, the instruction to move it down a line after printing the spaces is

```
PRINT "{3 LEFT} {DOWN}";
```

Figure 14-2 shows the cursor positions during the printing of the spaces and the object.



**Figure 14-2** Grid Layout for Printing Spaces and the Object.

Let us first see what happens if we omit the spaces above the object.

### **Program 14-1 Falling Asterisk Without Erasure**

```
NEW
100 PRINT CHR$(147)
110 FOR R=1 TO 24
120 PRINT "<*>";
130 PRINT "{3 LEFT} {DOWN}";
140 NEXT R
RUN
```

A whole column of objects appears in an instant! Inserting

```
135 FOR K=1 TO 200:NEXT K
```

slows the computer down enough so that you can see individual objects appear. To avoid multiple images, the computer must print spaces over each picture.

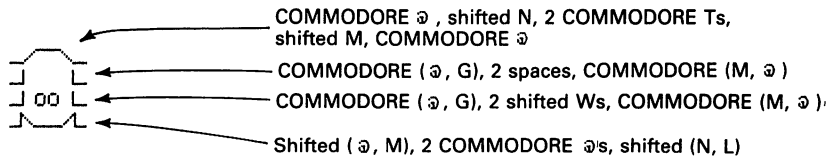
### **Program 14-2 Falling Asterisk With Erasure**

```
NEW
100 PRINT CHR$(147)
110 FOR R=1 TO 24
120 PRINT " ";
130 PRINT "{3 LEFT} {DOWN}";
140 PRINT "<*>";
150 PRINT "{3 LEFT}";
160 FOR K=1 TO 200:NEXT K
170 NEXT R
180 PRINT CHR$(147)
RUN
```

3 spaces

Put semicolons after lines 120 through 150.

We can use a similar approach to make a more complex figure fall. Let us try a spider that looks like



Be sure you move your finger from the SHIFT key to the COMMODORE key and back at the right times. As on a typewriter, pressing the space bar enters a space, regardless of whether you have SHIFT, SHIFT LOCK, or COMMODORE depressed.

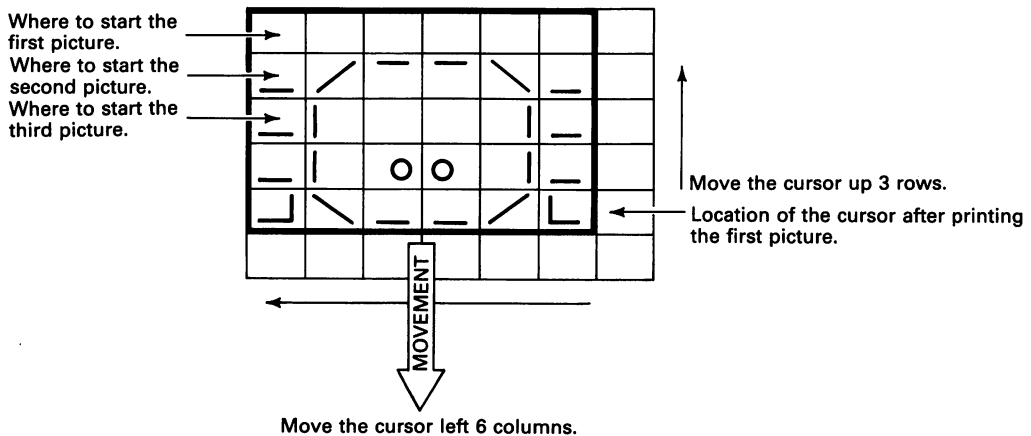


Figure 14-3 Grid Layout for Spider Movement.

As shown in Fig. 14-3, we put the picture on a grid to determine the proper cursor-moving commands. The command to move the entire picture down a line is

```
PRINT "{6 LEFT} {3 UP}";
```

As you can see from Fig. 14-4, the command to move from one line of the picture to the next is

```
PRINT "{6 LEFT} {DOWN}";
```

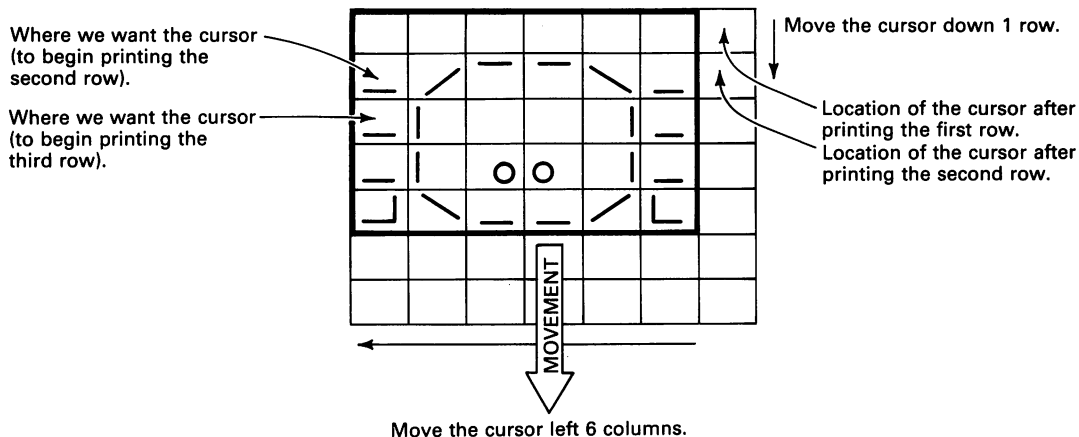


Figure 14-4 Grid Layout for Spider Lines.

### Program 14-3 Falling Spider

```

NEW
100 PRINT CHR$(147); ← Place a semicolon here.
110 FOR R=1 TO 20
120 GOSUB 1000
125 REM MOVE SPIDER DOWN 1 LINE
130 PRINT "{6 LEFT} {3 UP}";
140 FOR K=1 TO 200:NEXT K
150 NEXT R
160 GOTO 9000
990 REM PICTURE OF SPIDER
1000 PRINT " "; ← 6 spaces to avoid shadow.
1020 PRINT "  /  /  ";
1040 PRINT " |  |  ";
1060 PRINT " |  O O  ";
1080 PRINT " |  /  /  ";
1010 PRINT "{6 LEFT} {DOWN}"; ← These lines are all the same.
1030 PRINT "{6 LEFT} {DOWN}";
1050 PRINT "{6 LEFT} {DOWN}";
1070 PRINT "{6 LEFT} {DOWN}";
1090 RETURN
9000 PRINT CHR$(147)
RUN

```

We had to limit R to 20 here because the spider drawing occupies five lines, whereas the asterisk took only one. As usual, we must keep the picture on the screen and avoid printing on the bottom line.

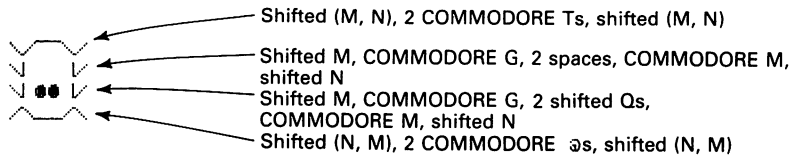
To make the spider leave a strand of web behind as it falls, change line 1000 to

```

1000 PRINT "  |  "; ← 2 spaces, COMMODORE M, 3 spaces

```

To make the spider's eyes blink and its legs move as it falls, we simply add another picture with different eyes and legs. The main program now draws two pictures of the spider at each position. The second picture is



To draw the two pictures in the same place, we must follow the first picture with

```
PRINT "{6 LEFT} {4 UP}";
```

### Program 14-4 Blinking, Falling Spider

```
NEW
100 PRINT CHR$(147); ← Put a semicolon here.
110 FOR R=1 TO 20
120 GOSUB 1000
130 PRINT "{6 LEFT} {4 UP}";
140 FOR K=1 TO 100:NEXT K
150 GOSUB 2000
160 PRINT "{6 LEFT} {3 UP}";
170 FOR K=1 TO 100:NEXT K
180 NEXT R
190 GOTO 9000
990 REM PICTURE OF SPIDER
1000 PRINT "  " ← 2 spaces, COMMODORE M, 3 spaces
1020 PRINT "  " ← COMMODORE @, shifted N, 2 COMMODORE Ts,
    shifted M, COMMODORE @
1040 PRINT "  " ← COMMODORE (@, G), 2 spaces, COMMODORE (M, @)
1060 PRINT "  OO  " ← COMMODORE (@, G), 2 shifted Ws, COMMODORE (M, @)
1080 PRINT "  " ← Shifted (@, M), 2 COMMODORE @s, shifted (N, L)
1010 PRINT "{6 LEFT} {DOWN}";
1030 PRINT "{6 LEFT} {DOWN}";
1050 PRINT "{6 LEFT} {DOWN}";
1070 PRINT "{6 LEFT} {DOWN}";
1090 RETURN
1990 REM PICTURE OF SPIDER
2000 PRINT "  " ← 2 spaces, COMMODORE M, 3 spaces
2020 PRINT "  " ← Shifted (M, N), 2 COMMODORE Ts, shifted (M, N)
2040 PRINT "  " ← Shifted M, COMMODORE G, 2 spaces, COMMODORE M, shifted N
2060 PRINT "  OO  " ← Shifted M, COMMODORE G, 2 shifted Qs, COMMODORE M,
    shifted N
2080 PRINT "  " ← Shifted (N, M), 2 COMMODORE @s, shifted (N, M)
2010 PRINT "{6 LEFT} {DOWN}";
2030 PRINT "{6 LEFT} {DOWN}";
2050 PRINT "{6 LEFT} {DOWN}";
2070 PRINT "{6 LEFT} {DOWN}";
2090 RETURN
9000 PRINT CHR$(147)
RUN
```

Be sure to put semicolons at the end of each PRINT line. You will have parts of spiders all over the screen if you omit any or type colons instead. Be careful; the colon key is next to the semicolon key, and the two symbols look similar.

We can add a strange noise using the POKE values from Chapter 13.

## Program 14-5 Blinking, Falling, Noisy Spider

Do not type NEW since we want to keep Program 14-4.

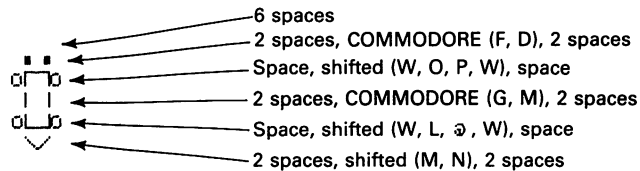
```
55 REM CLEAR SOUND CHIP AND SET VOLUME
60 FOR J=54272 TO 54296
70 POKE J,0
80 NEXT J
90 POKE 54296,15

140 GOSUB 8000
170 GOSUB 8000

7995 REM PRODUCE NOISE
8000 FOR K=1 TO 5
8010 POKE 54277,16*0+8:REM MINIMUM ATTACK, MEDIUM DECAY
8020 POKE 54276,17:REM START WAVEFORM 1
8030 POKE 54273,21
8040 POKE 54272,31
8050 FOR L=1 TO 25:NEXT L
8060 POKE 54276,16:REM STOP WAVEFORM 1
8070 NEXT K
8080 RETURN
```

The routine in lines 8000 through 8080 makes a strange noise. We simply replaced the time-wasting FOR-NEXT loops with GOSUBs to the sound-making routine, since making noise takes time and slows down the computer. Be sure to turn up the volume on your television set before running Program 14-5.

Let's now draw a race car moving down the screen. The car looks like



Since the car is six columns wide and six lines tall, the cursor commands are

```
PRINT "{6 LEFT} {DOWN}";
```

after each line except the last one, and

```
PRINT "{6 LEFT} {4 UP}";
```

after the last line.



## Program 14-6 Race Car Moving Down the Screen

```

NEW
100 PRINT CHR$(147);
110 FOR R=1 TO 18
120 GOSUB 1000
130 PRINT "{6 LEFT} {4 UP}";
140 FOR K=1 TO 200:NEXT K
150 NEXT R
160 GOTO 9000
995 REM PICTURE OF CAR HEADING DOWN
1000 PRINT "      "; ← 6 spaces
1020 PRINT "  ■■ ";
1040 PRINT "  O┌─┐ ";
1060 PRINT "  │ │ ";
1080 PRINT "  O└─┘ ";
1100 PRINT "  ∨ ";
1010 PRINT "{6 LEFT} {DOWN}"; ← These are all the same.
1030 PRINT "{6 LEFT} {DOWN}";
1050 PRINT "{6 LEFT} {DOWN}";
1070 PRINT "{6 LEFT} {DOWN}";
1090 PRINT "{6 LEFT} {DOWN}";
1110 RETURN
9000 PRINT CHR$(147)
RUN

```

The following changes let the car make noise:

## Program 14-7 Race Car With Loud Mufflers

Do not type NEW since we want to keep Program 14-6.

```

55 REM CLEAR SOUND CHIP AND SET VOLUME
60 FOR J=54272 TO 54296
70 POKE J,0
80 NEXT J
90 POKE 54296,15

140 GOSUB 8000

7995 REM PRODUCE SOUND
8000 FOR K=1 TO 5
8010 POKE 54277,0*16+12:REM MINIMUM ATTACK, HIGH DECAY
8020 POKE 54276,33:REM START WAVEFORM 2
8030 POKE 54273,6
8040 POKE 54272,167
8050 FOR L=1 TO 5:NEXT L
8060 POKE 54276,32:REM STOP WAVEFORM 2
8070 NEXT K
8080 RETURN
RUN

```

To make a bug fall, use the following drawing:

```

┌───┐ ← COMMODORE @, 3 shifted Ws, COMMODORE @
├───┤ ← Shifted @, 3 COMMODORE Ts, shifted L
└───┘ ← Shifted (@, M), COMMODORE @, shifted (N, L)

```

## Program 14-8 Falling Bug

```
NEW
100 PRINT CHR$(147);
110 FOR R=1 TO 20
120 GOSUB 1000
130 PRINT "{5 LEFT} {3 UP}";
140 FOR K=1 TO 200:NEXT K
150 NEXT R
160 GOTO 9000
990 REM PICTURE OF BUG
1000 PRINT " "; ← Blank line above bug — 5 spaces.
1020 PRINT " _000_ ";
1040 PRINT "  _ _ _ _ _ ";
1060 PRINT "  _ _ _ _ _ ";
1080 PRINT " "; ← Blank line below bug — 5 spaces.
1010 PRINT "{5 LEFT} {DOWN}";
1030 PRINT "{5 LEFT} {DOWN}";
1050 PRINT "{5 LEFT} {DOWN}";
1070 PRINT "{5 LEFT} {DOWN}";
1090 RETURN
9000 PRINT CHR$(147)
RUN
```

This bug starts in column 1. To draw a bug starting in a random column, we insert

```
105 C=INT(RND(1)*35)
108 PRINT TAB(C);
```

These instructions move the bug right a random number of columns between 0 and 34. The upper limit is 34, since we do not want any part of the bug in the rightmost column.

To produce weird sounds as the bug falls, we simply replace the slowdown statement (line 140) with a GOSUB to a sound-making subroutine. We must also remember to clear the sound chip and set the volume.

## Program 14-9 Infestation of Noisy Falling Bugs

Leave Program 14-8 plus lines 105 and 108 in the computer. Do not enter NEW!

```
55 REM CLEAR SOUND CHIP AND SET VOLUME
60 FOR J=54272 TO 54296
70 POKE J,0
80 NEXT J
90 POKE 54296,10

140 GOSUB 8000
7990 REM PRODUCE SOUND
8000 FOR K=1 TO 5
8010 POKE 54277,16*4+0:REM SET LOW ATTACK, MINIMUM DECAY
8020 POKE 54276,33:REM START WAVEFORM 2
8030 POKE 54273,22
8040 POKE 54272,96
8050 FOR L=1 TO 20:NEXT L
8060 POKE 54276,32:REM STOP WAVEFORM 2
8070 NEXT K
8080 RETURN
RUN
```

To create an infestation of falling bugs, change line 9000 to

```
9000 GOTO 60
```

The computer will now drop bugs forever. To stop it, press the RUN/STOP key and enter

```
POKE 54276,32
```

(without a line number) if the sound stays on.

Since we put blank lines above and below the bug, we can easily make it move up and down. We can add the following picture with legs extended and eyes shut:

•••• ← Shifted (M, Q, Q, Q, N)  
◡◡◡◡ ← Shifted (M, O), COMMODORE T, shifted (P, N)  
◡◡◡◡ ← Shifted (N, M), COMMODORE Ⓟ, shifted (N, M)

### **Program 14-10 Bug Moving Up and Down**

```
NEW
55 REM CLEAR SOUND CHIP AND SET VOLUME
60 FOR J=54272 TO 54296
70 POKE J,0
80 NEXT J
90 POKE 54296,15
100 PRINT CHR$(147); ← Put a semicolon here.
105 C=INT(RND(1)*35)
108 PRINT TAB(C); ← Put a semicolon here.
109 REM MOVE BUG DOWN AND UP SCREEN
110 FOR R=1 TO 20
120 GOSUB 1000
130 PRINT "{5 LEFT} {4 UP}";
140 GOSUB 8000
150 GOSUB 2000
160 PRINT "{5 LEFT} {3 UP}";
170 GOSUB 8000
180 NEXT R
209 REM MOVE BUG UP SCREEN
210 FOR R=1 TO 20
220 GOSUB 1000
230 PRINT "{5 LEFT} {4 UP}";
240 GOSUB 8000
250 GOSUB 2000
260 PRINT "{5 LEFT} {5 UP}";
270 GOSUB 8000
280 NEXT R
290 GOTO 9000
```

Lines 210 through 280 are the same as 110 through 180 except for the picture-moving commands (lines 160 and 260).

```

995 REM PICTURE OF BUG WITH EYES OPEN, LEGS OUT
1000 PRINT "      "; ← Blank line above bug — 5 spaces
1020 PRINT "_000_"; ← COMMODORE @ , 3 shifted Ws, COMMODORE
1040 PRINT "┌───┐"; ← Shifted @ , 3 COMMODORE Ts, shifted L
1060 PRINT "└─┬─┘"; ← Shifted ( @ , M), COMMODORE @ , shifted (N, L)
1080 PRINT "      "; ← Blank line below bug — 5 spaces
1010 PRINT "{5 LEFT} {DOWN}";
1030 PRINT "{5 LEFT} {DOWN}";
1050 PRINT "{5 LEFT} {DOWN}";
1070 PRINT "{5 LEFT} {DOWN}";
1090 RETURN
1995 REM PICTURE OF BUG WITH EYES CLOSED, LEGS UP
2000 PRINT "      "; ← Blank line above bug — 5 spaces
2020 PRINT "●●●┌───┐";
2040 PRINT "└─┬─┘";
2060 PRINT "      ";
2080 PRINT "      "; ← Blank line below bug — 5 spaces
2010 PRINT "{5 LEFT} {DOWN}";
2030 PRINT "{5 LEFT} {DOWN}";
2050 PRINT "{5 LEFT} {DOWN}";
2070 PRINT "{5 LEFT} {DOWN}";
2090 RETURN
7995 REM PRODUCE SOUND
8000 FOR K=1 TO 5
8010 POKE 54277,16*4+0:REM LOW ATTACK, MINIMUM DECAY
8020 POKE 54276,33:REM START WAVEFORM 2
8030 POKE 54273,22
8040 POKE 54272,96
8050 FOR L=1 TO 20:NEXT L
8060 POKE 54276,32:REM STOP WAVEFORM 2
8070 NEXT K
8080 RETURN
9000 GOTO 60
RUN

```

Since this program runs forever, you must use RUN/STOP to regain control. We could have the following single loop move first down and then up.

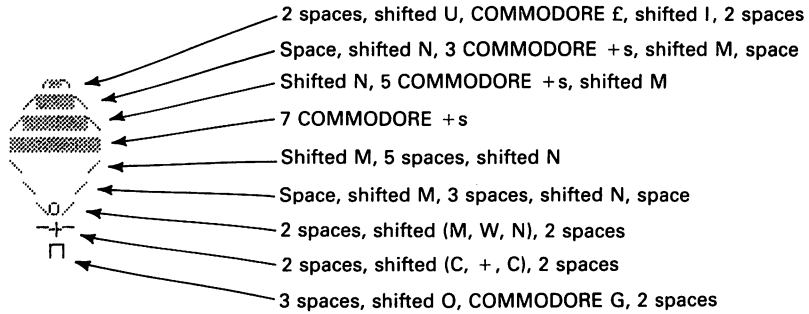
```

109 REM MOVE BUG DOWN AND UP
110 FOR J=1 TO 40
120 GOSUB 1000
130 PRINT "{5 LEFT} {4 UP}";
140 GOSUB 8000
150 GOSUB 2000
160 PRINT "{5 LEFT} {3 UP}";
162 REM ADJUST CURSOR CHANGE TO MOVE BUG UP
165 IF J>20 THEN PRINT "{2 UP}";
170 GOSUB 8000
180 NEXT J

```

The symbol > means “greater than.” Here J runs from 1 to 40, and line 165 makes up for the difference between line 260 and line 160 in Program 14-10.

We can also make a paratrooper fall from the sky and land on the ground. The paratrooper looks like



We must remember to draw the ground first. We will use a random number to select the paratrooper's horizontal position. Since the paratrooper is seven columns wide, the TAB value is

```
C=INT(RND(1)*33)
```

### Program 14-11 Paratrooper Landing

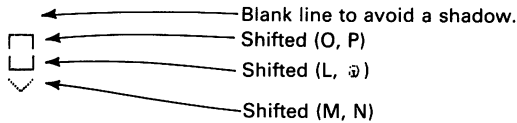
```
NEW
100 PRINT CHR$(147)
105 REM MOVE TO BOTTOM OF SCREEN
110 FOR R=1 TO 23
120 PRINT
130 NEXT R
135 REM DRAW GROUND
140 PRINT "████████████████████████████████████████████████████████████████████████████████";
150 PRINT CHR$(19);
160 C=INT(RND(1)*33)
170 PRINT TAB(C);
180 FOR R=1 TO 15
190 PRINT " " ;
210 PRINT " " ;
230 PRINT " " ;
250 PRINT " " ;
270 PRINT " " ;
290 PRINT " " ;
310 PRINT " " ;
330 PRINT " " ;
350 PRINT " " ;
370 PRINT " " ;
200 PRINT "{7 LEFT} {DOWN}";
220 PRINT "{7 LEFT} {DOWN}";
240 PRINT "{7 LEFT} {DOWN}";
260 PRINT "{7 LEFT} {DOWN}";
280 PRINT "{7 LEFT} {DOWN}";
300 PRINT "{7 LEFT} {DOWN}";
320 PRINT "{7 LEFT} {DOWN}";
340 PRINT "{7 LEFT} {DOWN}";
360 PRINT "{7 LEFT} {DOWN}";
380 PRINT "{7 LEFT} {B UP}";
390 FOR K=1 TO 200:NEXT K
400 NEXT R
410 GOTO 100
RUN
```

39 COMMODORE Us  
 Moves cursor to top of screen.  
 Put a semicolon here.  
 7 spaces to avoid a shadow.

# 15

## BOMBING GAME

We are now ready to create a simple game in which the player tries to drop bombs on targets (peacefully, of course). Let us start with a falling bomb that looks like



To leave room at the bottom for targets, we will use FOR R=1 TO 19 to move the bomb.

Since the bomb is two columns wide and four lines high, the cursor command to move it down a row is

```
PRINT "{2 LEFT} {2 UP}";
```

### **Program 15-1 Falling Bomb**

```
NEW
100 PRINT CHR$(147);
110 FOR R=1 TO 19
120 GOSUB 1000
130 PRINT "{2 LEFT} {2 UP}";
140 FOR K=1 TO 100:NEXT K
150 NEXT R
160 GOTO 9000
995 REM PICTURE OF BOMB
1000 PRINT "  ";
1020 PRINT "□";
1040 PRINT "□";
1060 PRINT "▽";
```

← 2 spaces  
Put semicolons at the end of lines 1000 through 1060.

```

1010 PRINT "{2 LEFT} {DOWN}";
1030 PRINT "{2 LEFT} {DOWN}";
1050 PRINT "{2 LEFT} {DOWN}";
1070 RETURN
9000 PRINT CHR$(147)
RUN

```

We can easily draw ground at the bottom for the bomb to strike, and have it make noise as it falls.

### **Program 15-2 Bombs Falling to Ground, With Noise**

```

NEW
55 REM CLEAR SOUND CHIP AND SET VOLUME
60 FOR J=54272 TO 54296
70 POKE J,0
80 NEXT J
90 POKE 54296,5
100 PRINT CHR$(147);
105 REM PUT GROUND AT BOTTOM OF SCREEN
110 FOR R=1 TO 22
120 PRINT
130 NEXT R
140 PRINT "
150 PRINT CHR$(19);
160 FOR R=1 TO 19
170 GOSUB 1000
180 PRINT "{2 LEFT} {2 UP}";
190 GOSUB 8000
200 NEXT R
210 GOTO 9000
995 REM PICTURE OF BOMB
1000 PRINT "  ";
1020 PRINT "□";
1040 PRINT "□";
1060 PRINT "√";
1010 PRINT "{2 LEFT} {DOWN}";
1030 PRINT "{2 LEFT} {DOWN}";
1050 PRINT "{2 LEFT} {DOWN}";
1070 RETURN
7995 REM PRODUCE BOMB SOUND
8000 FOR K=1 TO 5
8010 POKE 54277,0*16+10:REM MINIMUM ATTACK, MEDIUM DECAY
8020 POKE 54276,17:REM START WAVEFORM 1
8030 POKE 54273,23
8040 POKE 54272,181
8050 FOR L=1 TO 5:NEXT L
8060 POKE 54276,16:REM STOP WAVEFORM 1
8070 NEXT K
8080 RETURN
9000 PRINT CHR$(147)
RUN

```

39 COMMODORE Is  
Return to top of screen.

2 spaces  
Shifted (O, P)  
Shifted (L, @)  
Shifted (M, N)

Put semicolons at the end of line 1000 through line 1060.

We can use a similar approach to drop a smaller bomb, remembering to overwrite it with spaces to eliminate the shadow.

## Program 15-3 Small Bomb Falling, With Noise

```

NEW
55 REM CLEAR SOUND CHIP AND SET VOLUME
60 FOR J=54272 TO 54296
70 POKE J,0
80 NEXT J
90 POKE 54296,5
100 PRINT CHR$(147); ← Put a semicolon here.
105 REM DRAW GROUND AT BOTTOM OF SCREEN
110 FOR R=1 TO 23
120 PRINT
130 NEXT R
140 PRINT "          "; ← 39 COMMODORE is
150 PRINT CHR$(19); ← Put a semicolon here.
160 FOR R=1 TO 22 ← Return to top of screen.
170 GOSUB 1000
180 PRINT "{3 LEFT}"; ← Cursor move for next picture.
190 GOSUB 8000
200 NEXT R
210 GOTO 9000
995 REM PICTURE OF BOMB
1000 PRINT "      "; ← 3 spaces
1020 PRINT "(♦)"; ← Shifted Z
1010 PRINT "{3 LEFT} {DOWN}"; ← Cursor move between picture lines.
1030 RETURN
7995 REM PRODUCE BOMB SOUND
8000 FOR K=1 TO 5
8010 POKE 54277,16*0+10:REM MINIMUM ATTACK, MEDIUM DECAY
8020 POKE 54276,17:REM START WAVEFORM 1
8030 POKE 54273,23
8040 POKE 54272,181
8050 FOR L=1 TO 5:NEXT L
8060 POKE 54276,16:REM STOP WAVEFORM 1
8070 NEXT K
8080 RETURN
9000 PRINT CHR$(147)
RUN

```

A bomb now falls each time we type RUN. To control when it falls, we can use GET as in Program 12-1. We must add statements like

```

154 GET K$
156 IF K$<>"F" THEN 154

```

These statements freeze the computer until the player presses the F key. The sequence <> (that is, less than or greater than) means "not equal to." To enter it, press the < key (shifted comma) first, then the > key (shifted period). The steps in dropping the bomb are:

1. Clear the screen.
2. Draw a picture of the bomb at the top.
3. Wait for the player to press the F key.
4. Drop the bomb.
5. Make noise.



### **Program 15-4 Bombing Under Keyboard Control**

Do not type NEW since we want to retain Program 15-3.

```
151 REM PLACE BOMB IN COLUMN 20, TOP ROW
152 PRINT TAB(19)"(◆)"; ← Shifted Z
153 REM WAIT FOR PLAYER TO PRESS F KEY
154 GET K$
156 IF K$<>"F" THEN 154
157 REM DROP BOMB
158 PRINT "{3 LEFT}";
RUN
```

Program 15-4 always drops the bomb in column 20. That works well if a target is there, but does not present much of a challenge. We would like to be able to move the bomb horizontally before dropping it. To do this, we need more control keys, namely:

- R to move the bomb right one column
- L to move the bomb left one column
- E to end the game

A series of IF THEN statements will determine if the player has pressed F, R, L, or E. For example, to make the computer branch to line 230 if the player presses F, we use

```
IF K$="F" THEN 230
```

Similarly, to add one to C (column number) if the player presses R, we use

```
IF K$="R" THEN C=C+1
```

Here we follow THEN with an ordinary statement rather than a line number. The idea is the same, however; the computer does  $C=C+1$  (making C one larger than it was) only if K\$ is "R". Increasing C by one, of course, moves the bomb right one column if we position it with TAB(C).

To subtract one from C if the player presses L, we use

```
IF K$="L" THEN C=C-1
```

We must also put a space on each side of the bomb to eliminate any shadows when it moves.

## Program 15-5 Moving Bomb Before Dropping It

```

NEW
55 REM CLEAR SOUND CHIP AND SET VOLUME
60 FOR J= 54272 TO 54296
70 POKE J,0
80 NEXT J
90 POKE 54296,5
100 PRINT CHR$(147); ← Put a semicolon here.
110 FOR R=1 TO 23
120 PRINT
130 NEXT R
140 PRINT "          " ← 39 COMMODORE Is
142 REM START BOMB IN COLUMN 20
145 C=18 ← C is one less because of the space
150 PRINT CHR$(19); ← in front.
155 REM PLACE BOMB IN COLUMN C+1, TOP ROW
160 PRINT TAB(C) " (◆) "; ← Put spaces around bomb.
165 REM LOOK FOR COMMANDS FROM KEYBOARD ← Shifted Z
170 GET K$
175 REM F KEY DROPS BOMB
180 IF K$="F" THEN 230
185 REM R KEY MOVES BOMB RIGHT
190 IF K$="R" THEN C=C+1 ← Be sure you press + alone. Shifted +
195 REM L KEY MOVES BOMB LEFT ← looks like a giant + sign, but will
200 IF K$="L" THEN C=C-1 ← cause an error.
205 REM E KEY ENDS GAME
210 IF K$="E" THEN 9000
220 GOTO 150
225 REM MOVE CURSOR TO END OF ACTUAL BOMB (NOT SPACE)
230 PRINT "{LEFT}";
235 REM DROP BOMB
240 FOR R=1 TO 23
245 REM ERASE OLD BOMB, DRAW NEW ONE
250 PRINT "{3 LEFT}"; ← Cursor move for next picture.
260 GOSUB 1000
265 REM PRODUCE BOMB SOUND
270 GOSUB 8000
280 NEXT R
290 GOTO 9000
995 REM PICTURE OF BOMB
1000 PRINT "   "; ← 3 spaces
1020 PRINT " (◆) "; ← Shifted Z
1010 PRINT "{3 LEFT} {DOWN}"; ← Cursor move between lines.
1030 RETURN
7995 REM PRODUCE BOMB SOUND
8000 FOR K=1 TO 5
8010 POKE 54277,16*0+10:REM MINIMUM ATTACK, MEDIUM DECAY
8020 POKE 54276,17:REM START WAVEFORM 1
8030 POKE 54273,23
8040 POKE 54272,181
8050 FOR L=1 TO 5:NEXT L
8060 POKE 54276,16:REM STOP WAVEFORM 1
8070 NEXT K
8080 RETURN
9000 PRINT CHR$(147)
RUN

```

Note that the R and L keys do not repeat; you must press and release one of them each time you want to move the bomb.

We can make the program repeat forever by changing line 290 to

```
290 GOTO 100
```

Now, when you type RUN, the program returns to line 100 after dropping the bomb. You must press the E key to stop the program.

If you keep pressing the L or R key, C will eventually become less than 0 or more than 35. Values below 0 in a TAB make the computer stop and report an ILLEGAL QUANTITY ERROR. Values above 35 make the bomb wrap around on the screen, so it appears partly at the right end of a line and partly at the left end of the next line. This looks strange, to say the least. To keep these things from happening, we add the following statements to stop the player from moving the bomb too far:

```
192 REM KEEP BOMB ON SCREEN RIGHT
194 IF C>35 THEN C=35
```

and

```
202 REM KEEP BOMB ON SCREEN LEFT
204 IF C<0 THEN C=0
```

These statements restrict the bomb to starting in columns 1 through 36 (remember that TAB(C) moves the print position to column C+1). The symbol > means "greater than," and < means "less than." Now when we move the bomb too far, it seems to run into an invisible barrier at either side. You can keep pressing the direction key as much as you like, but the bomb will not move any farther.

We can also change the scene so the parentheses around the bomb represent a bomb bay (perhaps attached to an aircraft above the screen). Now only the bomb (the diamond) falls, while the bay stays at the top. To do this, we must change the cursor commands and the picture of the falling bomb. The new lines are

```
225 REM MOVE CURSOR TO END OF BOMB
230 PRINT "{2 LEFT}";
250 PRINT "{LEFT}"; ←————— Cursor move between pictures.

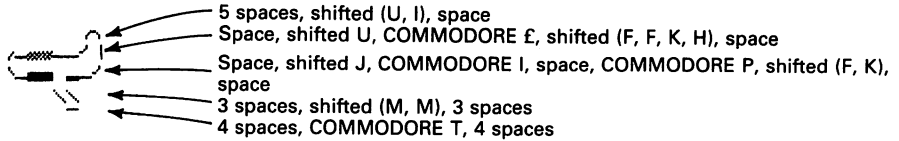
995 REM PICTURE OF BOMB
1000 PRINT " "; ←————— 1 space
1020 PRINT "◆"; ←————— Shifted Z
1010 PRINT "{LEFT} {DOWN}"; ←————— Cursor move between lines.
```

We could also add some targets by mixing shifted V's or shifted +'s with the COM-  
MODORE I's on line 140.



makes the game run continuously until you press E.

Let us now make the game more elaborate and more interesting by having an airplane deliver the bombs. The airplane with a space at both ends of each line looks like



The airplane is five lines tall and occupies eight columns with a space on either side. So we cannot start the airplane beyond column 32 and must therefore restrict C to 31.

### Program 15-7 Enemy Bomber Dropping Bombs

```

NEW
55 REM CLEAR SOUND CHIP AND SET VOLUME
60 FOR J=54272 TO 54296
70 POKE J,0
80 NEXT J
90 POKE 54296,5
100 PRINT CHR$(147);
110 FOR R=1 TO 23
120 PRINT
130 NEXT R
140 PRINT "  X  X  X  X  X  X  X  X";
142 REM START AIRPLANE IN COLUMN 16
145 C=15
150 PRINT CHR$(19);
155 REM DRAW AIRPLANE
160 GOSUB 2000
165 REM LOOK FOR COMMANDS FROM KEYBOARD
170 GET K$
175 REM F KEY DROPS BOMB
180 IF K$="F" THEN 230
185 REM R KEY MOVES AIRPLANE RIGHT
190 IF K$="R" THEN C=C+1
192 REM KEEP AIRPLANE ON SCREEN RIGHT
194 IF C>31 THEN C=31
195 REM L KEY MOVES AIRPLANE LEFT
200 IF K$="L" THEN C=C-1
202 REM KEEP AIRPLANE ON SCREEN LEFT
204 IF C<0 THEN C=0
205 REM E KEY ENDS GAME
210 IF K$="E" THEN 9000
220 GOTO 150
225 REM MOVE CURSOR TO BOMB BAY
230 PRINT "{5 LEFT} {UP}";
235 REM DROP BOMB
240 FOR R=1 TO 20
245 REM ERASE OLD BOMB, DROP NEW ONE
250 PRINT "{LEFT}";
260 GOSUB 1000
265 REM PRODUCE BOMB SOUND
270 GOSUB 8000
280 NEXT R
290 GOTO 100

995 REM PICTURE OF BOMB
1000 PRINT " ";
1020 PRINT "◆";
1010 PRINT "{LEFT} {DOWN}";
1030 RETURN

```

8 blocks of (4 COMMODORE Is, shifted V)

Put a semicolon here.

Be sure you press + alone. Shifted + looks like a giant + sign, but will cause an error.

Cursor move for next picture.

1 space


Shifted Z

Cursor move between lines.

```

2000 PRINT TAB(C) "
2010 PRINT TAB(C) "
2020 PRINT TAB(C) "
2030 PRINT TAB(C) "
2040 PRINT TAB(C) "
2100 RETURN
7995 REM PRODUCE BOMB SOUND
8000 FOR K=1 TO 5
8010 POKE 54277,16*0+10:REM MINIMUM ATTACK, MEDIUM DECAY
8020 POKE 54276,17:REM START WAVEFORM 1
8030 POKE 54273,23
8040 POKE 54272,181
8050 FOR L=1 TO 5:NEXT L
8060 POKE 54276,16:REM STOP WAVEFORM 1
8070 NEXT K
8080 RETURN
9000 PRINT CHR$(147)
RUN

```



Put a space on both sides of each line.

Put a semicolon here only to prepare for starting bomb.

Remember, the only way to stop this program is by pressing E or STOP. Also remember that you must press F to drop the bomb, R to move the airplane right, and L to move it left. When you press F, you will see that the bombs fall from the front of the wing. If you want to have them fall from the tail section, change line 230 to

```
230 PRINT "{2 LEFT} {UP}";
```

The airplane has the remarkable ability to fly backward when you press R (a great reverse gear). You could make the scene more realistic by adding a picture of the airplane moving right. The program would then draw the current picture when you press L and the new picture when you press R.

We could also have an explosion with sound effects and debris flying in the air when the bomb strikes the ground.

### **Program 15-8 Enemy Airplane Dropping Bombs, With Explosion**

```

285 REM PRODUCE EXPLOSION
290 GOSUB 6000
300 GOTO 100

5995 REM EXPLOSION - FRAGMENTS, CLOUD, AND NOISE
5998 REM DESTROY SOME GROUND
6000 PRINT "{2 LEFT}";
6010 PRINT "#####"; ← 3 COMMODORE Es
6015 REM SEND UP SOME DEBRIS
6020 PRINT "{3 LEFT} {UP}";
6030 PRINT "■ ■"; ← COMMODORE F, space, COMMODORE D
6035 REM MAKE SOME EXPLOSION NOISE
6040 GOSUB 7000
6045 REM SEND UP MORE DEBRIS
6050 PRINT "{4 LEFT} {UP}";
6060 PRINT "■ ■ ■ ■"; ← COMMODORE (F, F, E, D, D)
6065 REM MAKE MORE EXPLOSION NOISE
6070 GOSUB 7000
6075 REM SEND UP A MUSHROOM CLOUD
6080 PRINT "{5 LEFT} {UP}";
6090 PRINT "#####"; ← COMMODORE (E, +, +, +, E)
6095 REM MAKE MORE EXPLOSION NOISE
6100 GOSUB 7000
6110 RETURN

```

```
6995 REM PRODUCE EXPLOSION SOUND EFFECTS
7000 POKE 54277,16*4+12:REM LOW ATTACK, HIGH DECAY
7010 POKE 54276,129:REM START WAVEFORM 4
7020 POKE 54296,15:REM RAISE VOLUME
7030 POKE 54273,5
7040 POKE 54272,71
7050 FOR L=1 TO 500:NEXT L
7060 POKE 54276,128:REM STOP WAVEFORM 4
7070 RETURN
```

# 16

## MOVING THINGS DIAGONALLY

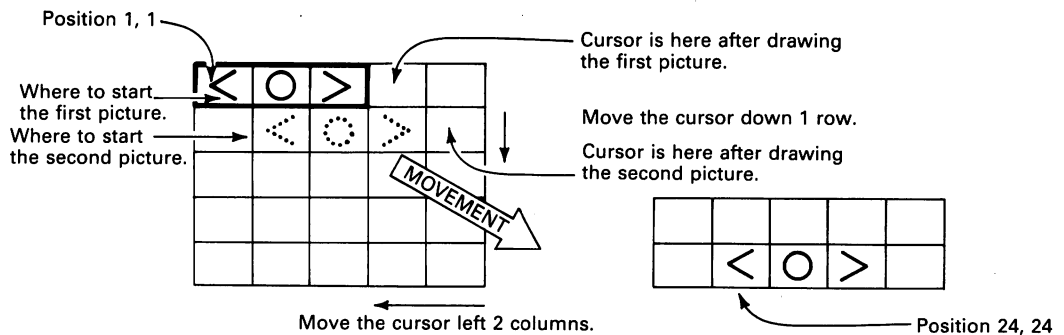
So far, we have moved objects right, left, up, and down. We will now learn how to move them diagonally.

Let's draw a small eye creature <O> and move it right diagonally.

### **Program 16-1 Eye Creature Moving Right Diagonally, With Shadow**

```
NEW
100 PRINT CHR$(147); ← Put a semicolon here.
110 FOR J=1 TO 24
120 PRINT "<O>";
130 PRINT "{2 LEFT} {DOWN}";
140 FOR K=1 TO 200:NEXT K
150 NEXT J
160 PRINT CHR$(147)
RUN
```

A diagonal line of creatures appears, beginning in the top left-hand corner. The first creature starts in position 1,1; the second in position 2,2; the third in 3,3; etc.; all the way to position 24,24.





The cursor command to move the creature diagonally is

```
PRINT "{2 LEFT} {DOWN}";
```

We can eliminate the multiple images by printing spaces over previous drawings

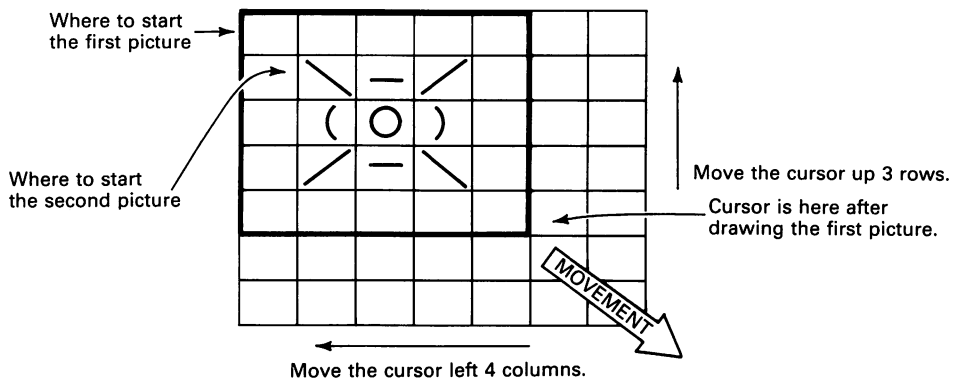
**Program 16-2 Eye Creature Moving Diagonally, Without Shadow**

```
NEW
100 PRINT CHR$(147); ← Put semicolon here.
110 FOR J=1 TO 24
120 PRINT "<O>";
130 PRINT "{3 LEFT}"; ← Moves back to start of picture.
140 FOR K=1 TO 200:NEXT K
150 PRINT "  "; ← 3 spaces
160 PRINT "{2 LEFT} {DOWN}"; ← Moves cursor to beginning of next picture.
170 NEXT J
180 PRINT CHR$(147)
RUN
```

Using the ideas from previous chapters, we can draw any object and move it diagonally. We must be careful, however, not to draw off the screen (columns 1 to 40 and rows 1 to 25).

If the computer prints more than 25 lines, the entire screen moves up, distorting the background scenery (houses, trees, flowers, etc.). When a line exceeds 40 columns, the computer starts to wrap it around to the next row, causing multiple images.

To move the alien creature diagonally, we must use the following diagram:



The cursor command to move the alien diagonally is

```
PRINT "{4 LEFT} {3 UP}";
```

The blank lines above and below the alien let us move it up or down diagonally with no shadow. In fact, we need spaces on all sides. The alien consists of the following lines:

```

1000 PRINT "      "; ← 5 spaces
1020 PRINT " \-/ "; ← Shifted (M, C, N)
1040 PRINT " [O] "; ← Put one space on each side of alien.
1060 PRINT " /-\ "; ← Shifted (N, C, M)
1080 PRINT "      "; ← 5 spaces

```

The cursor commands to move to the next line are

```

1010 PRINT "{5 LEFT} {DOWN}";
1030 PRINT "{5 LEFT} {DOWN}";
1050 PRINT "{5 LEFT} {DOWN}";
1070 PRINT "{5 LEFT} {DOWN}";

```

A little planning and sketching on paper simplifies program writing. We will use GOSUB and RETURN again. You have probably already noticed that using the 1000's for one idea or picture, the 2000's for another, etc., makes it easier to enter, correct, and modify programs. We also get to see the pictures as we enter them.

### Program 16-3 Alien Moving Right Diagonally, With Sound

Remember to turn the volume up on the television set!

```

NEW
55 REM CLEAR SOUND CHIP AND SET VOLUME
60 FOR J=54272 TO 54296
70 POKE J,0
80 NEXT J
90 POKE 54296,15
100 PRINT CHR$(147); ← Put a semicolon here.
105 REM MOVE ALIEN FROM UPPER LEFT TO LOWER RIGHT
110 FOR J=1 TO 20
120 GOSUB 1000
130 PRINT "{4 LEFT} {3 UP}"; ← Picture-moving command.
140 GOSUB 8000
150 NEXT J
160 GOTO 9000
990 REM PICTURE OF ALIEN
1000 PRINT "      "; ← 5 spaces.
1020 PRINT " \-/ "; ← Draw the picture.
1040 PRINT " [O] "; ← 5 spaces.
1060 PRINT " /-\ "; ← 5 spaces.
1080 PRINT "      "; ← 5 spaces.
1010 PRINT "{5 LEFT} {DOWN}";
1030 PRINT "{5 LEFT} {DOWN}";
1050 PRINT "{5 LEFT} {DOWN}";
1070 PRINT "{5 LEFT} {DOWN}";
1090 RETURN
7995 REM ALIEN'S VOICE
8000 POKE 54277,16*0+12:REM MINIMUM ATTACK, HIGH DECAY
8010 POKE 54276,17:REM START WAVEFORM 1
8020 POKE 54273,4
8030 POKE 54272,112 ← Alien's voice.
8040 FOR L=1 TO 250:NEXT L
8050 POKE 54276,16:REM STOP WAVEFORM 1
8060 RETURN
9000 PRINT CHR$(147)
RUN

```

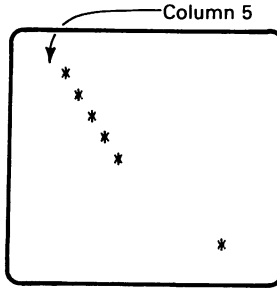
We used FOR J=1 TO 20 to keep the five-line alien on the screen.

To move an object from a column other than #1, we must sketch what we want before writing the program. Let's move an asterisk diagonally, beginning in column 5.

POSITION

1, 5  
2, 6  
3, 7  
4, 8  
5, 9

24, 28



The FOR loop becomes FOR J=1 TO 24. To start the asterisk in column 5, we insert TAB (4) immediately after clearing the screen.

### Program 16-4 Asterisk Moving Diagonally

```
NEW
100 PRINT CHR$(147); ← Clears the screen.
110 PRINT TAB(4); ← Moves cursor to column 5. Note that TAB(4) moves
120 FOR J=1 TO 24 the cursor right 4 columns from column 1.
130 PRINT "*";
140 PRINT "{LEFT}";
150 FOR K=1 TO 200:NEXT K
160 PRINT " "; ← Put 1 space here.
170 PRINT "{DOWN}";
180 NEXT J
190 PRINT CHR$(147)
RUN
```

For the alien, which occupies more than one line, we can combine the ideas of Programs 16-3 and 16-4. Let's start the alien in column 4 by using

```
PRINT TAB(3);
```

Here we need

```
FOR J=1 TO 20
```

to move the figure down.

We can also superimpose two versions of the alien to produce a pulsating figure. The new picture is

```
┌─┐ ← Shifted (P, C, O)
├─┘ ← <, COMMODORE O, >
└─┘ ← Shifted ( @, C, L)
```

The cursor command after the first picture must now put the second picture in the same place. The command is

```
130 PRINT "{5 LEFT} {4 UP}";
```

The command to move the pulsating alien diagonally is the same as line 130 in Program 16-3, that is,

```
PRINT "{4 LEFT} {3 UP}";
```

### Program 16-5 Pulsating Alien Moving Right Diagonally

```
NEW
55 REM CLEAR SOUND CHIP AND SET VOLUME
60 FOR J=54272 TO 54296
70 POKE J,0
80 NEXT J
90 POKE 54296,15
100 PRINT CHR$(147);
110 PRINT TAB(3);
120 FOR J=1 TO 20
125 REM DRAW FIRST ALIEN
130 GOSUB 1000
140 PRINT "{5 LEFT} {4 UP}";
150 GOSUB 8000
155 REM DRAW SECOND ALIEN
160 GOSUB 2000
170 PRINT "{4 LEFT} {3 UP}";
180 GOSUB 7000
190 NEXT J
200 GOTO 9000
995 REM FIRST ALIEN PICTURE
1000 PRINT "    ";
1020 PRINT " \-/ ";
1040 PRINT " [O] ";
1060 PRINT " /-\ ";
1080 PRINT "    ";
1010 PRINT "{5 LEFT} {DOWN}";
1030 PRINT "{5 LEFT} {DOWN}";
1050 PRINT "{5 LEFT} {DOWN}";
1070 PRINT "{5 LEFT} {DOWN}";
1090 RETURN
1995 REM SECOND ALIEN PICTURE
2000 PRINT "    ";
2020 PRINT "  H ";
2040 PRINT " <O> ";
2060 PRINT "  JL ";
2080 PRINT "    ";
2010 PRINT "{5 LEFT} {DOWN}";
2030 PRINT "{5 LEFT} {DOWN}";
2050 PRINT "{5 LEFT} {DOWN}";
2070 PRINT "{5 LEFT} {DOWN}";
2090 RETURN
6995 REM SECOND ALIEN'S VOICE
7000 POKE 54277,16*4+0:REM LOW ATTACK, MINIMUM DECAY
7010 POKE 54276,33:REM START WAVEFORM 2
7020 POKE 54273,18
7030 POKE 54272,209
7040 FOR L=1 TO 250:NEXT L
7050 POKE 54276,32:REM STOP WAVEFORM 2
7060 RETURN
7995 REM FIRST ALIEN'S VOICE
8000 POKE 54277,16*0+12:REM MINIMUM ATTACK, HIGH DECAY
8010 POKE 54276,17:REM START WAVEFORM 1
8020 POKE 54273,4
8030 POKE 54272,112
8040 FOR L=1 TO 250:NEXT L
8050 POKE 54276,16:REM STOP WAVEFORM 1
8060 RETURN
9000 PRINT CHR$(147)
RUN
```

Put a semicolon here.

Moves cursor back to start of the picture.

Picture-moving command.

5 spaces. Same as lines 1080, 2000, and 2080.

Put 1 space on each side of alien.

5 spaces

Shifted (P, C, O)

<, COMMODORE O, >

Shifted (@, C, L)

5 spaces

So far, the creatures have moved diagonally right. We can make one move diagonally left by starting it at the right and changing the cursor command that moves the picture.

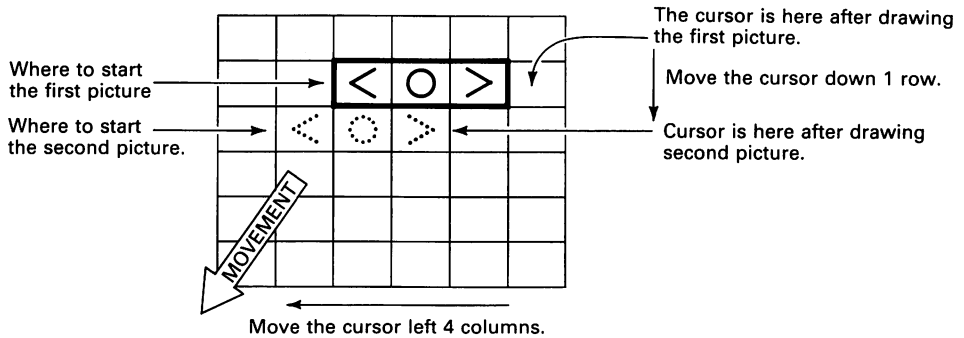
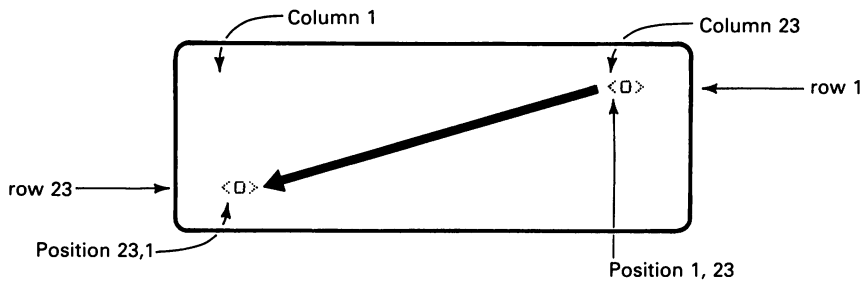
**Program 16-6 Eye Creature Moving Left Diagonally**

```

NEW
100 PRINT CHR$(147);
110 PRINT TAB(22);
120 FOR J=1 TO 23
130 PRINT "<O>";
140 PRINT "{4 LEFT} {DOWN}";
150 FOR K=1 TO 200:NEXT K
160 NEXT J
170 PRINT CHR$(147)
RUN

```

The eye creatures form a diagonal from position 1,23 to 2,22; 3,21; and so on, all the way to 23,1, that is,



We can eliminate the multiple images by printing spaces over old images.



```

170 GOTO 9000
1000 PRINT " ";
1020 PRINT " \-/ ";
1040 PRINT " [O] ";
1060 PRINT " /-\ ";
1080 PRINT " ";
1010 PRINT "{5 LEFT} {DOWN}";
1030 PRINT "{5 LEFT} {DOWN}";
1050 PRINT "{5 LEFT} {DOWN}";
1070 PRINT "{5 LEFT} {DOWN}";
1090 RETURN
7995 REM ALIEN'S VOICE
8000 POKE 54277,16*0+12:REM MINIMUM ATTACK, HIGH DECAY
8010 POKE 54276,17:REM START WAVEFORM 1
8020 POKE 54273,4
8030 POKE 54272,112
8040 FOR L=1 TO 250:NEXT L
8050 POKE 54276,16:REM STOP WAVEFORM 1
8060 RETURN
9000 PRINT CHR$(147)
RUN

```

Annotations in the original image:

- Arrows point from the text "Put 1 space on each side of the alien." to the spaces around the alien's body in lines 1020, 1040, and 1060.
- An arrow points from the text "5 spaces" to the five spaces between the top and bottom parts of the alien in line 1010.

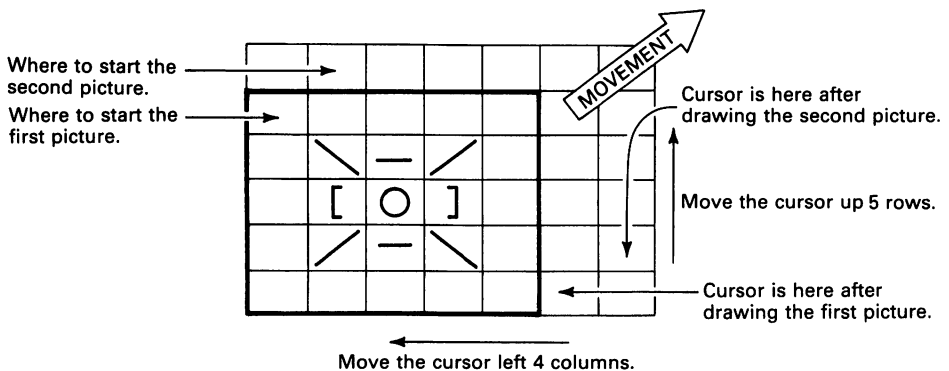
To move the alien from the top right-hand corner, change line 110 to

```
110 PRINT TAB(34);
```

To move an object right diagonally from the bottom, we change line 140 in Program 16-8 to

```
140 PRINT "{4 LEFT} {5 UP}";
```

The change in line 140 comes from the following diagram:



To start the alien near the bottom, replace line 110 of Program 16-8 with

```

110 FOR J=1 TO 20
112 PRINT
115 NEXT J

```

### Program 16-9 Alien Moving Right Diagonally, From Bottom

Do not enter NEW. Leave Program 16-8 in memory.

```
105 REM START ALIEN NEAR BOTTOM, AT LEFT
110 FOR J=1 TO 20
112 PRINT
115 NEXT J
140 PRINT "{4 LEFT} {5 UP}";
RUN
```

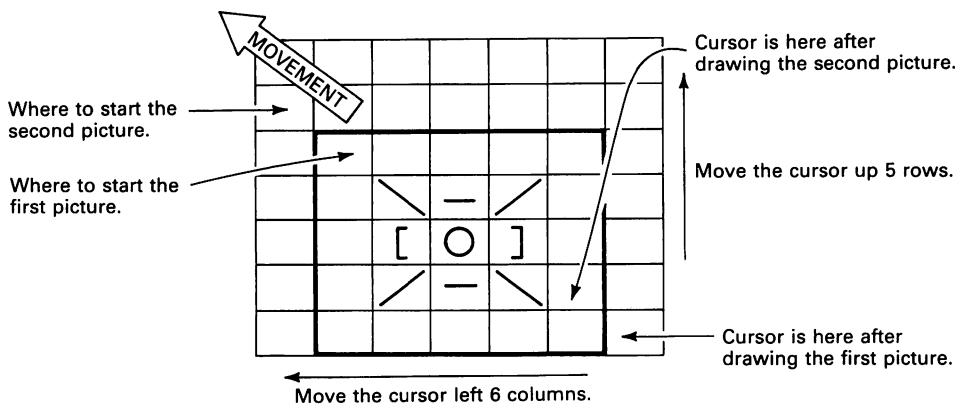
To move the alien left diagonally from the bottom, we must start the cursor further right with

```
105 REM START ALIEN NEAR BOTTOM IN COLUMN 20
117 PRINT TAB(19);
```

This puts the cursor initially in column 20 of row 21. The cursor command to move the picture is

```
140 PRINT "{6 LEFT} {5 UP}";
```

We obtain it from the following diagram:



### Program 16-10 Alien Moving Left Diagonally, From Bottom

Do not enter NEW. Leave Program 16-9 in memory.

```
105 REM MOVE ALIEN FROM LOWER RIGHT TO UPPER LEFT
117 PRINT TAB(19);
140 PRINT "{6 LEFT} {5 UP}";
RUN
```

To start the alien in the lower right-hand corner, use

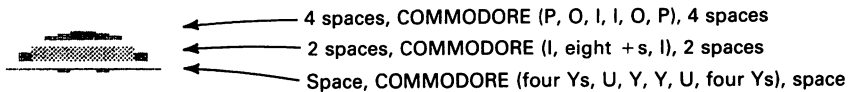
```
117 PRINT TAB(34);
```



# 17

## SPACESHIP GAME

We are now ready to produce more complex scenes by combining programs. For example, we could draw a master spaceship and let it drop a small guided bomb. The master ship looks like



The guided bomb is the symbol **◆** (shifted Z).

The program will let us move the master ship right or left, drop a bomb, and then direct the bomb right or left as it descends. We will use the following control keys:

- E Stop the game (exit).
- L Move master ship or bomb left.
- R Move master ship or bomb right.
- F Drop a bomb.

After the bomb strikes the ground, we can then return to controlling the master ship again.

### ***Program 17-1 Moving the Master Spaceship***

```
NEW
100 PRINT CHR$(147);
195 REM START MASTER SHIP IN COLUMN 13
200 C=12
205 REM DRAW MASTER SHIP
210 PRINT CHR$(19);
220 GOSUB 1000
295 REM LOOK FOR KEYBOARD COMMANDS
300 GET K$
```



```

505 REM L KEY MOVES BOMB LEFT
510 IF K$<>"L" THEN 530
520 IF CB>0 THEN CB=CB-1:PRINT "<LEFT>";
525 REM MOVE BOMB DOWN A LINE
530 PRINT "<LEFT> <DOWN>";
540 NEXT J
550 GOTO 210
RUN

```

Lines 430 through 460 draw and erase the bomb. You can make the bomb move faster by reducing the upper limit in line 450.

The computer reaches line 490 only if the player presses R. That statement moves the cursor right one column if there is room on the screen. Note that line 490 has two statements after THEN, separated by a colon. The computer does both only if CB is less than 38. Line 520 similarly moves the cursor left one column if the player presses L and there is room on the screen. Line 530 is the normal cursor command to move the bomb down a line.

As in Program 17-1, you can initially press L or R to move the master ship. Now, however, pressing F actually drops a bomb rather than just printing a message. As the bomb falls, you can guide it with the L or R key. Note that you get only 21 chances to do this since we use

```
FOR J=1 TO 21
```

to control the bomb's descent. Since the bomb moves slowly, you should be able to hit the targets with ease.

An interesting feature of the Commodore 64 is that it remembers which keys you pressed even if it cannot act on them right away. This feature is generally used to allow it to keep up with a fast typist. In our game, this memory lets you press L or R several times and then watch the bomb move left or right as if by delayed reaction. Be careful, however; if you press L or R repeatedly while the bomb is falling, the program may interpret some commands as spaceship moves. Remember—as soon as the bomb strikes the ground, L and R resume their roles as control keys for the master spaceship.

Note the importance of dividing a large program into small programs (called *subroutines* or *modules*). You can then start with a rough outline of what you want and add more features as you progress. The best approach is to always use the same pattern. Assign each idea a group of line numbers and add one new section at a time to your program. Test each feature you add to make sure it works.

One problem with Program 17-2 is that the bomb falls from the left side of the master ship. To make it fall from under the ship, we must move the cursor right six columns initially. Note, however, that we must also increase CB by six in line 410. The changes thus are

```

400 PRINT TAB(C+6);
410 CB=C+7

```

We can add even more player control to our game. For example, we could let the player determine the bomb's speed. This involves changing the upper limit in the slow-down loop (line 450). Let us call that upper limit M and give it an initial value of 200.

We will assign two new control keys as follows:

- F (fast) makes the bomb move faster.
- S (slow) makes the bomb move slower.

This works as follows:

1. Pressing F makes the computer divide M by 2, thus making the delay between pictures half as long.
2. Pressing S makes the computer multiply M by 2, thus making the delay twice as long.

Program 17-3 contains the revisions required to introduce speed control.

### **Program 17-3 Spaceship Game With Speed Control**

Do not type NEW.

```
412 REM START TIME CONSTANT AT 200
415 M=200
```

```
445 REM VARIABLE WAIT BETWEEN PICTURES
450 FOR K=1 TO M:NEXT K
```

```
471 REM SPEED CONTROL COMMANDS (F DOUBLES SPEED, S HALVES SPEED)
```

```
472 IF K$="F" THEN M=M/2
```

```
473 IF K$="S" THEN M=M*2
```

```
RUN
```

Note that / means divide and \* means multiply.

Run Program 17-3. Press F to drop a bomb and then keep pressing F. The bomb rapidly picks up speed and strikes the ground in an instant. Press F again to drop another bomb; then press S repeatedly. The bomb virtually stops moving. In fact, it looks as though it is suspended in air.

The computer is now spending so much of its time spinning its wheels on line 450 that it hardly ever looks at the keyboard. Once you slow the bomb down, therefore, it is almost impossible to get it moving again. One way to avoid having this happen is to keep the speed within reasonable limits. The following lines keep M between 50 and 400.

```
476 REM RESTRICT SPEED TO REASONABLE VALUES
477 IF M<50 THEN M=50
478 IF M>800 THEN M=800
```

We could produce an explosion when the bomb strikes the ground by inserting lines similar to the ones we used in Program 15-8. We must also add

```
55 REM CLEAR SOUND CHIP AND SET VOLUME
60 FOR J=54272 TO 54296
70 POKE J,0
80 NEXT J
90 POKE 54296,15
```

```

545 REM PRODUCE EXPLOSION
550 PRINT "{UP} {RIGHT}";
560 GOSUB 6000
570 GOTO 100

```

### Program 17-4 Spaceship Game With Speed Control and Explosions

```

NEW
55 REM CLEAR SOUND CHIP AND SET VOLUME
60 FOR J=54272 TO 54296
70 POKE J,0
80 NEXT J
90 POKE 54296,15
100 PRINT CHR$(147);
105 REM DRAW GROUND WITH TARGETS
110 FOR J=1 TO 23
120 PRINT
130 NEXT J
140 PRINT "  X  X  X  X  X  X  X  X  X  X";
195 REM START MASTER SHIP IN COLUMN 13
200 C=12
205 REM DRAW MASTER SHIP
210 PRINT CHR$(19);
220 GOSUB 1000
295 REM LOOK FOR KEYBOARD CONTROL
300 GET K$
305 REM F KEY DROPS GUIDED BOMB
310 IF K$="F" THEN 400
315 REM R KEY MOVES MASTER SHIP RIGHT
320 IF K$="R" THEN C=C+1
325 REM KEEP SHIP ON SCREEN RIGHT
330 IF C>25 THEN C=25
335 REM L KEY MOVES MASTER SHIP LEFT
340 IF K$="L" THEN C=C-1
345 REM KEEP SHIP ON SCREEN LEFT
350 IF C<0 THEN C=0
355 REM E KEY ENDS GAME
360 IF K$="E" THEN 9000
370 GOTO 210
395 REM DROP BOMB
398 REM MOVE CURSOR TO STARTING POSITION OF BOMB
400 PRINT TAB(C+6);
405 REM KEEP TRACK OF BOMB'S COLUMN NUMBER
410 CB=C+7
412 REM START TIME CONSTANT AT 1000
415 M=1000
420 FOR J=1 TO 21
425 REM DRAW AND ERASE BOMB
430 PRINT "◆"; ← Shifted Z
440 PRINT "{LEFT}";
445 REM VARIABLE WAIT BETWEEN PICTURES
450 FOR K=1 TO M:NEXT K
460 PRINT " "; ← 1 space
465 REM LOOK FOR GUIDANCE COMMANDS
470 GET K$
471 REM SPEED CONTROL COMMANDS (F DOUBLES SPEED, S HALVES SPEED)
472 IF K$="F" THEN M=M/2 ← Note that / means divide and * means multiply.
473 IF K$="S" THEN M=M*2 ←
475 REM R KEY MOVES BOMB RIGHT
476 REM RESTRICT SPEED TO REASONABLE VALUES
477 IF M<50 THEN M=50
478 IF M>800 THEN M=800
480 IF K$("<"R" THEN 510
485 REM KEEP BOMB ON SCREEN RIGHT
490 IF CB<38 THEN CB=CB+1:PRINT "{RIGHT}";

```

```

500 GOTO 530
505 REM L KEY MOVES BOMB LEFT
510 IF K$<>"L" THEN 530
520 IF CB>0 THEN CB=CB-1:PRINT "{LEFT}";
525 REM MOVE BOMB DOWN A LINE
530 PRINT "{LEFT} {DOWN}";
540 NEXT J
545 REM PRODUCE EXPLOSION
550 PRINT "{UP} {RIGHT}";
560 GOSUB 6000
570 GOTO 100
995 REM PICTURE OF MASTER SHIP
1000 PRINT TAB(C) "          " "
1010 PRINT TAB(C) " ██████████ " ← Put 1 space on each side of ship.
1020 PRINT TAB(C) " ██████████ "
1030 RETURN

5995 REM EXPLOSION - FRAGMENTS, CLOUD, AND NOISE
5998 REM DESTROY SOME GROUND
6000 PRINT "{2 LEFT}";
6010 PRINT "#####"; ← 3 COMMODORE £s
6015 REM SEND UP SOME DEBRIS
6020 PRINT "{3 LEFT} {UP}";
6030 PRINT "■ ■"; ← COMMODORE F, space, COMMODORE D
6035 REM MAKE SOME EXPLOSION NOISE
6040 GOSUB 7000
6045 REM SEND UP MORE DEBRIS
6050 PRINT "{4 LEFT} {UP}";
6060 PRINT "■ ■ * ■ ■"; ← COMMODORE (F, F, £, D, D)
6065 REM MAKE MORE EXPLOSION NOISE
6070 GOSUB 7000
6075 REM SEND UP A MUSHROOM CLOUD
6080 PRINT "{5 LEFT} {UP}";
6090 PRINT "#####"; ← COMMODORE (£, +, +, +, £)
6095 REM MAKE MORE EXPLOSION NOISE
6100 GOSUB 7000
6110 RETURN
6995 REM PRODUCE EXPLOSION SOUND EFFECTS
7000 POKE 54277,16*4+12:REM LOW ATTACK, HIGH DECAY
7010 POKE 54276,129:REM START WAVEFORM 4
7020 POKE 54273,5
7030 POKE 54272,71
7040 FOR L=1 TO 500:NEXT L
7050 POKE 54276,128:REM STOP WAVEFORM 4
7060 RETURN
9000 PRINT CHR$(147)
RUN

```

# 18

## CONTROLLING MOTION IN ALL DIRECTIONS

Now let us create a game that involves moving an object right, left, up, or down. This requires four control keys instead of two. Let us start with an alien creature that looks the same regardless of which way it is moving. We will control its motion with the following keys:

- W Move up one line.
- A Move left one column.
- S Move right one column.
- Z Move down one line.

Although the letters do not mean anything, these keys form a convenient cluster on the left side of the keyboard. The W (up) key is above the others, A (left) is to the left, S (right) to the right, and Z (down) is below the others. Thus, if you use one finger to control the creature, you should have no trouble remembering how to move it in a particular direction. The mnemonic keys (U for up, L for left, R for right, and D for down) are scattered around the keyboard, making control much more difficult. Furthermore, L is actually to the right and R to the left.

The alien looks like

\-/ ← Shifted (M, C, N)  
[O] ← [, shifted W, ]  
/-\ ← Shifted (N, C, M)

Since it is not even clear whether the alien has a top, bottom, or sides, we can move it freely in any direction.

When we put spaces and blank lines around it to avoid shadows, the alien becomes five lines tall and five columns wide. The command that moves the cursor from the end of its picture back to the beginning is therefore

```
PRINT "{5 LEFT} {4 UP}";
```

We can make the computer move the alien in any direction by simply following this statement with an additional cursor move.

The routine to draw the alien is

```

1000 PRINT "      "; ← Put 1 space on each side of alien.
1020 PRINT " \-/ "; ← 5 spaces
1040 PRINT " [O] "; ← Space, shifted (M, C, N), space
1060 PRINT " /-\ "; ← Space, [, shifted W, ], space
1080 PRINT "      "; ← Space, shifted (N, C, M), space
                             5 spaces
1010 PRINT "{5 LEFT} {DOWN}";
1030 PRINT "{5 LEFT} {DOWN}"; ← Cursor command to move to next line.
1050 PRINT "{5 LEFT} {DOWN}";
1070 PRINT "{5 LEFT} {DOWN}";
1090 RETURN

```

As usual, the E key stops the game.

### **Program 18-1 Alien Moving in Four Directions**

```

NEW
100 PRINT CHR$(147)
105 REM START ALIEN IN ROW 10, COLUMN 18
110 PRINT CHR$(19); ← Put a semicolon here.
120 FOR J=1 TO 9
130 PRINT
140 NEXT J
150 PRINT TAB(17); ← Put a semicolon here.
155 REM DRAW ALIEN
160 GOSUB 1000
165 REM WAIT BETWEEN PICTURES
170 FOR K=1 TO 100:NEXT K
175 REM RETURN CURSOR TO BEGINNING OF PICTURE
180 PRINT "{5 LEFT} {4 UP}";
185 REM LOOK FOR KEYBOARD COMMANDS
190 GET K$
195 REM E KEY EXITS GAME
200 IF K$="E" THEN 9000
205 REM W KEY MOVES ALIEN UP
210 IF K$="W" THEN PRINT "{UP}";
215 REM Z KEY MOVES ALIEN DOWN
220 IF K$="Z" THEN PRINT "{DOWN}";
225 REM A KEY MOVES ALIEN LEFT
230 IF K$="A" THEN PRINT "{LEFT}";
235 REM S KEY MOVES ALIEN RIGHT
240 IF K$="S" THEN PRINT "{RIGHT}";
250 GOTO 160
995 REM PICTURE OF ALIEN
1000 PRINT "      "; ← 5 spaces
1020 PRINT " \-/ ";
1040 PRINT " [O] ";
1060 PRINT " /-\ ";
1080 PRINT "      ";

```



```

1010 PRINT "{S LEFT} {DOWN}";
1030 PRINT "{S LEFT} {DOWN}";
1050 PRINT "{S LEFT} {DOWN}";
1070 PRINT "{S LEFT} {DOWN}";
1090 RETURN
9000 PRINT CHR$(147)
RUN

```

← Cursor command to move to next line.

You can use the W, A, S, and Z keys to move the alien. Keep it away from the screen boundaries, however, since we have not yet restricted its travel. To do that, we must keep track of where the alien's picture starts. We need both the row number R and the column number C. Since we begin the alien in row 10, column 18, we must insert

```

152 R=10
154 C=18

```

Then, before obeying a keyboard command, the program must determine whether the move is reasonable. If not, it simply keeps the alien where it was. The revised command section is as follows:

```

205 REM W KEY MOVES ALIEN UP
210 IF K$<>"W" THEN 230 ← <> means "not equal to."
215 REM MOVE ALIEN UP ONE ROW IF POSSIBLE
220 IF R>1 THEN R=R-1:PRINT "{UP}"; ← Colon between statements
225 REM Z KEY MOVES ALIEN DOWN                                     on the same line.
230 IF K$<>"Z" THEN 250
235 REM MOVE ALIEN DOWN ONE ROW IF POSSIBLE
240 IF R<20 THEN R=R+1:PRINT "{DOWN}";
245 REM A KEY MOVES ALIEN LEFT
250 IF K$<>"A" THEN 270
255 REM MOVE ALIEN LEFT ONE COLUMN IF POSSIBLE
260 IF C>1 THEN C=C-1:PRINT "{LEFT}";
265 REM S KEY MOVES ALIEN RIGHT
270 IF K$<>"S" THEN 160
275 REM MOVE ALIEN RIGHT ONE COLUMN IF POSSIBLE
280 IF C<35 THEN C=C+1:PRINT "{RIGHT}";
290 GOTO 160

```

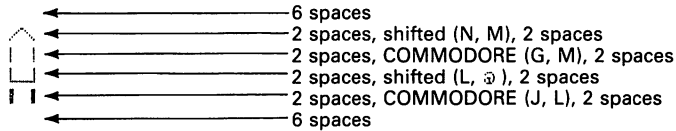
Lines 220, 240, 260, and 280 keep the alien from moving off the screen. Note that each of these lines has two statements after THEN with a colon between them. The computer does both (changing R or C and moving the cursor) only if the condition is true (that is, if the move is reasonable).

As we noted in discussing Program 17-3, you can press a direction key many times without waiting for the computer to respond. The computer will remember what you pressed and will eventually catch up with you. If you do this, however, the computer will not respond to other direction keys until it has finished catching up. The alien may thus appear to have momentum in a particular direction or be stuck in a corner, since it will not change direction until it has followed every command you entered.

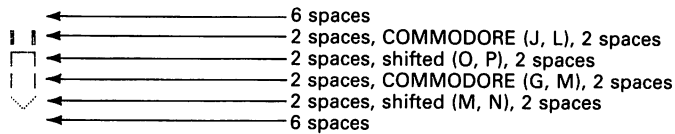
To make sure the modified program works, move the alien all the way to the right, left, top, and bottom. If something goes wrong, you must have made either a typing or a logical error. The computer does exactly what you tell it to do, not necessarily what you intend it to do.

Let's now draw a rocket and guide it in any of four directions. Unlike the alien, the rocket's appearance depends on which way it is moving. Thus, we need the following four pictures:

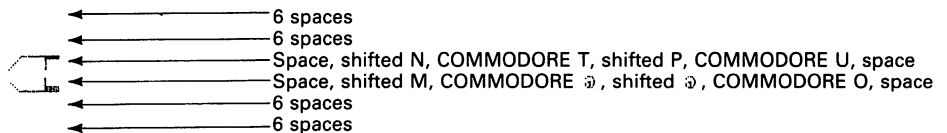
#### ROCKET MOVING UP



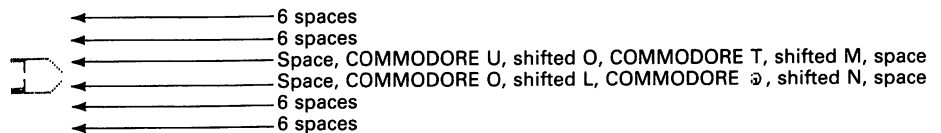
#### ROCKET MOVING DOWN



#### ROCKET MOVING LEFT



#### ROCKET MOVING RIGHT



Now the program not only must move the rocket in the proper direction and keep it on the screen, but must also draw the appropriate picture. Note that the rocket is slightly larger than the alien (six lines by six columns instead of five by five), so we must reduce the upper limits on both R and C by one.

## Program 18-2 Rocket Moving Anywhere

```

NEW
100 PRINT CHR$(147);
105 REM START ROCKET IN ROW 10, COLUMN 18
110 FOR J=1 TO 9
120 PRINT
130 NEXT J
140 PRINT TAB(17);
150 R=10
160 C=18
165 REM DRAW ROCKET MOVING UP INITIALLY
170 GOSUB 1000
175 REM RETURN CURSOR TO START OF PICTURE
180 PRINT "{6 LEFT} {5 UP}";
185 REM WAIT BETWEEN PICTURES
190 FOR K=1 TO 100:NEXT K
195 REM LOOK FOR KEYBOARD COMMANDS
200 GET K$
205 REM E KEY IS EXIT
210 IF K$="E" THEN 9000
215 REM W KEY MOVES ROCKET UP
220 IF K$<>"W" THEN 260
225 REM MOVE ROCKET UP ONE ROW IF POSSIBLE
230 IF R>1 THEN R=R-1:PRINT "{UP}";
235 REM DRAW ROCKET MOVING UP
240 GOSUB 1000
250 GOTO 180
255 REM Z KEY MOVES ROCKET DOWN
260 IF K$<>"Z" THEN 300
265 REM MOVE ROCKET DOWN ONE ROW IF POSSIBLE
270 IF R<19 THEN R=R+1:PRINT "{DOWN}";
275 REM DRAW ROCKET MOVING DOWN
280 GOSUB 2000
290 GOTO 180
300 IF K$<>"A" THEN 340
305 REM MOVE ROCKET LEFT ONE COLUMN IF POSSIBLE
310 IF C>1 THEN C=C-1:PRINT "{LEFT}";
315 REM DRAW ROCKET MOVING LEFT
320 GOSUB 3000
330 GOTO 180
335 REM S KEY MOVES ROCKET RIGHT
340 IF K$<>"S" THEN 190
345 REM MOVE ROCKET RIGHT ONE COLUMN IF POSSIBLE
350 IF C<34 THEN C=C+1:PRINT "{RIGHT}";
355 REM DRAW ROCKET MOVING RIGHT
360 GOSUB 4000
370 GOTO 180

995 REM PICTURE OF ROCKET MOVING UP
1000 PRINT "      ";
1020 PRINT "      ^ ";
1040 PRINT "      | ";
1060 PRINT "      | ";
1080 PRINT "      || ";
1100 PRINT "      ";
1010 PRINT "{6 LEFT} {DOWN}";
1030 PRINT "{6 LEFT} {DOWN}";
1050 PRINT "{6 LEFT} {DOWN}";
1070 PRINT "{6 LEFT} {DOWN}";
1090 PRINT "{6 LEFT} {DOWN}";
1110 RETURN

```

Put semicolon here.

6 spaces

2 spaces, shifted (N, M), 2 spaces

2 spaces, COMMODORE (G, M), 2 spaces

2 spaces, shifted (L, @), 2 spaces

2 spaces, COMMODORE (J, L), 2 spaces

6 spaces

Move cursor to next line of picture — always the same.

```

1995 REM PICTURE OF ROCKET MOVING DOWN
2000 PRINT "      "; ← 6 spaces
2020 PRINT "      |"; ← 2 spaces, COMMODORE (J, L), 2 spaces
2040 PRINT "      |"; ← 2 spaces, shifted (O, P), 2 spaces
2060 PRINT "      |"; ← 2 spaces, COMMODORE (G, M), 2 spaces
2080 PRINT "      V"; ← 2 spaces, shifted (M, N), 2 spaces
2100 PRINT "      "; ← 6 spaces
2010 PRINT "{6 LEFT} {DOWN}";
2030 PRINT "{6 LEFT} {DOWN}";
2050 PRINT "{6 LEFT} {DOWN}";
2070 PRINT "{6 LEFT} {DOWN}";
2090 PRINT "{6 LEFT} {DOWN}";
2110 RETURN
2995 REM PICTURE OF ROCKET MOVING LEFT
3000 PRINT "      "; ← 6 spaces
3020 PRINT "      "; ← 6 spaces
3040 PRINT "      |"; ← Space, shifted N, COMMODORE T, shifted P,
3060 PRINT "      |"; ← COMMODORE U, space
3080 PRINT "      |"; ← Space, shifted M, COMMODORE @, shifted @,
3100 PRINT "      "; ← COMMODORE O, space
3010 PRINT "{6 LEFT} {DOWN}";
3030 PRINT "{6 LEFT} {DOWN}";
3050 PRINT "{6 LEFT} {DOWN}";
3070 PRINT "{6 LEFT} {DOWN}";
3090 PRINT "{6 LEFT} {DOWN}";
3110 RETURN
3995 REM PICTURE OF ROCKET MOVING RIGHT
4000 PRINT "      "; ← 6 spaces
4020 PRINT "      "; ← 6 spaces
4040 PRINT "      |"; ← Space, COMMODORE U, shifted O, COMMODORE T,
4060 PRINT "      |"; ← shifted M, space
4080 PRINT "      |"; ← Space, COMMODORE O, shifted L, COMMODORE @,
4100 PRINT "      "; ← shifted N, space
4010 PRINT "{6 LEFT} {DOWN}";
4030 PRINT "{6 LEFT} {DOWN}";
4050 PRINT "{6 LEFT} {DOWN}";
4070 PRINT "{6 LEFT} {DOWN}";
4090 PRINT "{6 LEFT} {DOWN}";
4110 RETURN
9000 PRINT CHR$(147)
RUN

```

If you use the CRSR keys to enter the rocket subroutines (you should, since so many of the lines are repetitive), be sure to LIST each subroutine separately to check the drawings. It is quite easy (as we know from experience) to accidentally omit a picture line, a cursor command, or a RETURN statement.

Run Program 18-2 and move the rocket around the screen. Move it all the way to the right, left, top, and bottom. It should not budge if you try to move it off the screen.

Note how we got to this point. We started by making things move in different directions. Then we introduced control keys and moved things left and right. Next we moved a single picture of an alien in four directions. Here we have extended the program to handle an object that looks different, depending on which way it is heading. Writing a complex program is obviously much simpler if you divide it into small parts and make each part work. You can then expand the program and add features easily.

Although Program 18-2 is long, it follows naturally from Program 18-1. Once we could move an alien in all four directions, it was relatively easy to extend the idea to drawing four different pictures. Note how much of the main program (the 100 and 200 lines) stayed the same, since we put all the pictures in subroutines.

To add sound, we could use a routine like the one in Program 16-8.

### **Program 18-3 Rocket Moving Anywhere, With Sound**

Do not type NEW since we want to keep Program 18-2.

```
55 REM CLEAR SOUND CHIP AND SET VOLUME
60 FOR J=54272 TO 54296
70 POKE J,0
80 NEXT J
90 POKE 54296,15
190 GOSUB 8000
7995 REM MAKE NOISE
8000 POKE 54277,16*0+15:REM MINIMUM ATTACK, MAXIMUM DECAY
8010 POKE 54276,17:START WAVEFORM 1
8020 POKE 54273,4
8030 POKE 54272,112
8040 FOR L=1 TO 100:NEXT L
8050 POKE 54276,16:REM STOP WAVEFORM 1
8060 RETURN
RUN
```

We can modify Program 9-3 to scatter objects randomly for the rocket to hunt down. We will use left-hand graphics for the random characters and choose one with

```
LH=INT(RND(1)*31)+161
PRINT CHR$(LH)
```

Look at Table 9-1 if you cannot remember how this works. The random position and character selection routines will be another subroutine starting at line 7000.

### **Program 18-4 Rocket Moving Anywhere, With Sound and Targets**

So not type NEW since we want to retain Program 18-3. If the noise annoys you, either turn down the volume on your television set or change line 190 back to

```
190 FOR K=1 TO 100:NEXT K

101 REM PLACE RANDOM TARGETS ON SCREEN
102 GOSUB 7000
103 PRINT CHR$(19);
6995 REM PUT RANDOM LEFT-HAND GRAPHICS IN RANDOM POSITIONS
7000 FOR J=1 TO 10 ← 10 objects
7010 PRINT CHR$(19);
7015 REM MOVE DOWN A RANDOM NUMBER OF ROWS
7020 NR=INT(RND(1)*24) ← Move down 0 to 23 rows.
7025 REM NO MOVEMENT DOWN IF RANDOM NUMBER IS ZERO
7030 IF NR=0 THEN 7070
7040 FOR R=1 TO NR
7050 PRINT
7060 NEXT R
7065 REM MOVE RIGHT A RANDOM NUMBER OF COLUMNS
7070 C=INT(RND(1)*39) ← Move right 0 to 38 columns.
7075 REM GENERATE A LEFT-HAND GRAPHICS CHARACTER
7080 LH=INT(RND(1)*31)+161 ← LH is between 161 and 191.
7090 PRINT TAB(C)CHR$(LH)
7100 NEXT J
7110 RETURN
RUN
```

Run Program 18-4 and see how long it takes you to destroy all the objects. Any resemblance to a famous trademarked video game is purely deliberate. Because of the spaces around the rocket, it does not have to actually strike an object. The object will disappear when the rocket comes near it. The rocket appears to be surrounded by one of those invisible destructive shields made so popular by science-fiction writers.

Note how convenient GOSUB and RETURN are. We can add new features easily from other programs by assigning them some line numbers, adding a RETURN at the end, and putting a GOSUB in the main program. You can, in fact, add more features, such as more sounds or smoke trailing from behind the rocket.

In Program 18-4 we must repeatedly press a key to keep the rocket moving. This is unrealistic, since an actual vehicle has momentum. Once it is moving, it keeps going until you apply thrust in another direction. This is particularly true for a rocket moving outside the atmosphere, with neither gravity nor friction to slow it down.

Let us modify our game so the rocket maintains its momentum between commands. Instead of moving the rocket, the command keys will simply establish its direction DI\$—UP, DOWN, LEFT, or RIGHT. The rocket will then move continuously in the DI\$ direction. Lines 215 on are now

```

215 REM W KEY MAKES DIRECTION UP
220 IF K$="W" THEN DI$="UP"
225 REM Z KEY MAKES DIRECTION DOWN
230 IF K$="Z" THEN DI$="DOWN"
235 REM A KEY MAKES DIRECTION LEFT
240 IF K$="A" THEN DI$="LEFT"
245 REM S KEY MAKES DIRECTION RIGHT
250 IF K$="S" THEN DI$="RIGHT"
255 REM MOVE ROCKET IN APPROPRIATE DIRECTION
260 IF DI$<>"UP" THEN 300
265 REM MOVE ROCKET UP ONE ROW IF POSSIBLE
270 IF R>1 THEN R=R-1:PRINT "{UP}";
275 REM DRAW ROCKET MOVING UP
280 GOSUB 1000
290 GOTO 180
300 IF DI$<>"DOWN" THEN 340
305 REM MOVE ROCKET DOWN ONE ROW IF POSSIBLE
310 IF R<19 THEN R=R+1:PRINT "{DOWN}";
315 REM DRAW ROCKET MOVING DOWN
320 GOSUB 2000
330 GOTO 180
340 IF DI$<>"LEFT" THEN 380
345 REM MOVE ROCKET LEFT ONE COLUMN IF POSSIBLE
350 IF C>1 THEN C=C-1:PRINT "{LEFT}";
355 REM DRAW ROCKET MOVING LEFT
360 GOSUB 3000
370 GOTO 180
380 IF DI$<>"RIGHT" THEN 180
385 REM MOVE ROCKET RIGHT ONE COLUMN IF POSSIBLE
390 IF C<34 THEN C=C+1:PRINT "{RIGHT}";
395 REM DRAW ROCKET MOVING RIGHT
400 GOSUB 4000
410 GOTO 180

```

We must also start the rocket in the upward direction by inserting

```
172 DI$="UP"
```

## Program 18-5 Rocket Game With Momentum

Do not type NEW since we want to keep Program 18-4 in memory.

```
172 DI$="UP"
190 FOR K=1 TO 100:NEXT K

215 REM W KEY MAKES DIRECTION UP
220 IF K$="W" THEN DI$="UP"
225 REM Z KEY MAKES DIRECTION DOWN
230 IF K$="Z" THEN DI$="DOWN"
235 REM A KEY MAKES DIRECTION LEFT
240 IF K$="A" THEN DI$="LEFT"
245 REM S KEY MAKES DIRECTION RIGHT
250 IF K$="S" THEN DI$="RIGHT"
255 REM MOVE ROCKET IN APPROPRIATE DIRECTION
260 IF DI$<>"UP" THEN 300
265 REM MOVE ROCKET UP ONE ROW IF POSSIBLE
270 IF R>1 THEN R=R-1:PRINT "{UP}";
275 REM DRAW ROCKET MOVING UP
280 GOSUB 1000
290 GOTO 180
300 IF DI$<>"DOWN" THEN 340
305 REM MOVE ROCKET DOWN ONE ROW IF POSSIBLE
310 IF R<19 THEN R=R+1:PRINT "{DOWN}";
315 REM DRAW ROCKET MOVING DOWN
320 GOSUB 2000
330 GOTO 180
340 IF DI$<>"LEFT" THEN 380
345 REM MOVE ROCKET LEFT ONE COLUMN IF POSSIBLE
350 IF C>1 THEN C=C-1:PRINT "{LEFT}";
355 REM DRAW ROCKET MOVING LEFT
360 GOSUB 3000
370 GOTO 180
380 IF DI$<>"RIGHT" THEN 180
385 REM MOVE ROCKET RIGHT ONE COLUMN IF POSSIBLE
390 IF C<34 THEN C=C+1:PRINT "{RIGHT}";
395 REM DRAW ROCKET MOVING RIGHT
400 GOSUB 4000
410 GOTO 180

6995 REM PUT RANDOM LEFT-HAND GRAPHICS IN RANDOM POSITIONS
7000 FOR J=1 TO 10 ← 10 objects
7010 PRINT CHR$(19);
7015 REM MOVE DOWN A RANDOM NUMBER OF ROWS
7020 NR=INT(RND(1)*23) ← Move down 0 to 22 rows.
7025 REM NO MOVEMENT DOWN IF RANDOM NUMBER IS ZERO
7030 IF NR=0 THEN 7070
7040 FOR R=1 TO NR
7050 PRINT
7060 NEXT R
7065 REM MOVE RIGHT A RANDOM NUMBER OF COLUMNS
7070 C=INT(RND(1)*38) ← Move right 0 to 37 columns.
7075 REM GENERATE A LEFT-HAND GRAPHICS CHARACTER
7080 LH=INT(RND(1)*31)+161
7090 PRINT TAB(C)CHR$(LH)
7100 NEXT J
7110 RETURN
9000 PRINT CHR$(147)
RUN
```

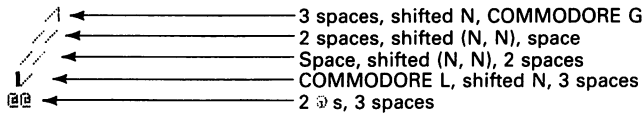
These changes make the game much more difficult, since the rocket moves rapidly between commands. If you ponder your next move too long, the rocket will go right past the objects and stop at an edge. An even more realistic game would force the player to stop the rocket before turning it. That is, if the rocket were moving left, the player would have to apply thrust to the right and stop the rocket before changing its direction.



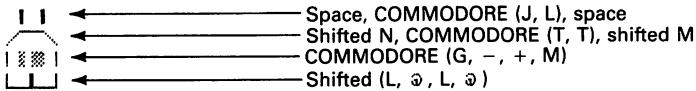
# 19

## TARGET PRACTICE

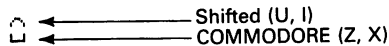
We can combine ideas to create new scenes. Let us draw an artillery gun firing a shell at a factory. The shell first rises at an angle for a while, then falls toward the target. The artillery gun looks like



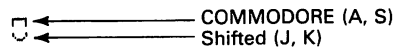
The target (a factory with smokestacks) looks like



The shell looks like



when it is rising and like



when it is falling.

To create the scene, we will combine several routines, each of which performs a single function. Let us start by making the shell rise diagonally from the lower left-hand corner. The required instructions are

```

FOR J=1 TO 19
GOSUB 1000 ← Draw shell.
FOR K=1 TO 200:NEXT K
PRINT "{2 LEFT} {UP}"; ← Move cursor back to start of picture.

GOSUB 3000 ← Erase shell.
PRINT "{LEFT} {2 UP}"; ← Move cursor to start of next picture.

```

The shell drawing (lines 1000 on) is

```

1000 PRINT "O";
1020 PRINT "U";
1010 PRINT "{2 LEFT} {DOWN}"; ← Move cursor to next line.

```

the shell erasure (lines 3000 on) is

```

3000 PRINT " "; ← 2 spaces
3020 PRINT " "; ← 2 spaces
3010 PRINT "{2 LEFT} {DOWN}"; ← Move cursor to next line.

```

Note that we have reserved the 2000 series lines for the picture of the falling shell. We have also left the 100 series line numbers for the background.

### Program 19-1 Shell Rising

```

NEW
100 PRINT CHR$(147)
205 REM START SHELL NEAR BOTTOM OF SCREEN
210 FOR J=1 TO 19
220 PRINT
230 NEXT J
235 REM MOVE SHELL UP DIAGONALLY
240 FOR J=1 TO 19
245 REM DRAW SHELL
250 GOSUB 1000
255 REM WAIT A WHILE
260 FOR K=1 TO 200:NEXT K
265 REM ERASE SHELL
270 PRINT "{2 LEFT} {UP}";
280 GOSUB 3000
285 REM MOVE SHELL UP DIAGONALLY
290 PRINT "{LEFT} {2 UP}";
300 NEXT J
310 GOTO 9000
995 REM PICTURE OF RISING SHELL
1000 PRINT "O"; ← Shifted (U, I)
1020 PRINT "U"; ← COMMODORE (Z, X)
1010 PRINT "{2 LEFT} {DOWN}";
1030 RETURN
2995 REM ERASE SHELL
3000 PRINT " "; ← 2 spaces
3020 PRINT " "; ← 2 spaces
3010 PRINT "{2 LEFT} {DOWN}";
3030 RETURN
9000 PRINT CHR$(147)
RUN

```

Let us now modify the program to make the shell first rise and then fall. This involves an additional picture (falling shell) and a new cursor command. The command to make the shell fall diagonally is

```
PRINT "{LEFT}";
```

Now we have two FOR-NEXT loops, each with J going from 1 to 19. Both use the erasure routine starting at line 3000.

### **Program 19-2 Shell Rising and Falling**

```

NEW
100 PRINT CHR$(147)
205 REM START SHELL AT BOTTOM OF SCREEN
210 FOR J=1 TO 19
220 PRINT
230 NEXT J
235 REM MOVE SHELL UP DIAGONALLY
240 FOR J=1 TO 19
245 REM DRAW SHELL RISING
250 GOSUB 1000
255 REM WAIT A WHILE
260 FOR K=1 TO 200:NEXT K
265 REM ERASE SHELL
270 PRINT "{2 LEFT} {UP}";
280 GOSUB 3000
285 REM MOVE SHELL UP DIAGONALLY
290 PRINT "{LEFT} {2 UP}";
300 NEXT J
305 REM MOVE SHELL DOWN DIAGONALLY
310 FOR J=1 TO 19
315 REM DRAW FALLING SHELL
320 GOSUB 2000
325 REM WAIT A WHILE
330 FOR K=1 TO 200:NEXT K
335 REM ERASE SHELL
340 PRINT "{2 LEFT} {UP}";
350 GOSUB 3000
355 REM MOVE SHELL DOWN ONE STEP DIAGONALLY
360 PRINT "{LEFT}";
370 NEXT J
380 GOTO 9000
995 REM PICTURE OF RISING SHELL
1000 PRINT " ◡ "; ← Shifted (U, I)
1020 PRINT " ◢ "; ← COMMODORE (Z, X)
1010 PRINT "{2 LEFT} {DOWN}";
1030 RETURN
1995 REM PICTURE OF FALLING SHELL
2000 PRINT " ◣ "; ← COMMODORE (A, S)
2020 PRINT " ◤ "; ← Shifted (J, K)
2010 PRINT "{2 LEFT} {DOWN}";
2030 RETURN
2995 REM ERASE SHELL
3000 PRINT "  "; ← 2 spaces
3020 PRINT "  "; ← 2 spaces
3010 PRINT "{2 LEFT} {DOWN}";
3030 RETURN
9000 PRINT CHR$(147)
RUN

```

Now the shell rises diagonally, turns over when it reaches the peak of its path, and then falls diagonally. We could add sound to the picture by replacing lines 260 and 330 with GOSUB 8000 and adding the routine from Chapter 15.

```

7995 REM PRODUCE SHELL SOUND
8000 FOR K=1 TO 5
8010 POKE 54277,16*0+10:REM MINIMUM ATTACK, MEDIUM DECAY
8020 POKE 54276,17:REM START WAVEFORM 1
8030 POKE 54273,23:REM OCTAVE 4, F SHARP
8040 POKE 54272,181
8050 FOR L=1 TO 5:NEXT L
8060 POKE 65276,16:REM STOP WAVEFORM 1
8070 NEXT K
8080 RETURN

```

You must also insert

```

55 REM CLEAR SOUND CHIP
60 FOR J=54272 TO 54296
70 POKE J,0
80 NEXT J
85 REM LOW VOLUME INITIALLY
90 POKE 54296,5

```

to initialize the sound chip. Be sure to turn the volume up on your television set as well.

We are now ready to add the background, the artillery gun, and the target. We must modify the shell's path to extend from the end of the gun to the top of the target. A little experimentation shows that the upward FOR loop should go from 1 to 12, and the downward FOR loop from 1 to 13.

### ***Program 19-3 Artillery Shelling a Factory***

```

NEW
100 PRINT CHR*(147)
105 REM MOVE CURSOR NEAR BOTTOM OF SCREEN
110 FOR J=1 TO 18
120 PRINT
130 NEXT J
135 REM DRAW ARTILLERY GUN
140 GOSUB 4000
145 REM MOVE CURSOR TO START OF TARGET
150 PRINT "{3 UP}";
160 PRINT TAB(30);
165 REM DRAW FACTORY
170 GOSUB 5000
205 REM MOVE CURSOR TO END OF GUN
210 PRINT "{6 UP}" ←
220 PRINT TAB(5); ←
235 REM MOVE SHELL UP DIAGONALLY
240 FOR J=1 TO 12
245 REM DRAW RISING SHELL
250 GOSUB 1000
255 REM WAIT A WHILE
260 FOR K=1 TO 200:NEXT K
265 REM ERASE SHELL
270 PRINT "{2 LEFT} {UP}";
280 GOSUB 3000
285 REM MOVE SHELL UP DIAGONALLY
290 PRINT "{LEFT} {2 UP}";
300 NEXT J

```

No semicolon here.  
Move right to end of gun.  
There is no line 230; lines 235 through 380 are almost identical to those in Program 19-2.

```

305 REM MOVE SHELL DOWN DIAGONALLY
310 FOR J=1 TO 13
315 REM DRAW FALLING SHELL
320 GOSUB 2000
325 REM WAIT A WHILE
330 FOR K=1 TO 200:NEXT K
335 REM ERASE SHELL
340 PRINT "{2 LEFT} {UP}";
350 GOSUB 3000
355 REM MOVE SHELL DOWN DIAGONALLY
360 PRINT "{LEFT}";
370 NEXT J
380 GOTO 9000

995 REM PICTURE OF RISING SHELL
1000 PRINT " ◯ "; ← Shifted (U, I)
1020 PRINT " ◻ "; ← COMMODORE (Z, X)
1010 PRINT "{2 LEFT} {DOWN}";
1030 RETURN
1995 REM PICTURE OF FALLING SHELL
2000 PRINT " ◻ "; ← COMMODORE (A, S)
2020 PRINT " ◯ "; ← Shifted (J, K)
2010 PRINT "{2 LEFT} {DOWN}";
2030 RETURN
2995 REM ERASURE OF SHELL
3000 PRINT "  "; ←
3020 PRINT "  "; ← 2 spaces
3010 PRINT "{2 LEFT} {DOWN}";
3030 RETURN
3995 REM PICTURE OF ARTILLERY GUN
4000 PRINT "  / "; ← 3 spaces, shifted N, COMMODORE G
4020 PRINT " // "; ← 2 spaces, shifted (N, N), space
4040 PRINT "  \ "; ← Space, shifted (N, N), 2 spaces
4060 PRINT "  \ "; ← COMMODORE L, shifted N, 2 spaces
4080 PRINT " @@" ; ← 2 @ s, 3 spaces
4010 PRINT "{5 LEFT} {DOWN}";
4030 PRINT "{5 LEFT} {DOWN}";
4050 PRINT "{5 LEFT} {DOWN}";
4070 PRINT "{5 LEFT} {DOWN}";
4090 RETURN
4995 REM PICTURE OF FACTORY
5000 PRINT " | | "; ← Space, COMMODORE (J, L), space
5020 PRINT " ◡ "; ← Shifted N, COMMODORE (T, T), shifted M
5040 PRINT " |███| "; ← COMMODORE (G, -, +, M)
5060 PRINT " |███| "; ← Shifted (L, @, L, @)
5010 PRINT "{4 LEFT} {DOWN}";
5030 PRINT "{4 LEFT} {DOWN}";
5050 PRINT "{4 LEFT} {DOWN}";
5070 RETURN
9000 PRINT CHR$(147)
RUN

```

Finally, let us add the sounds of the gun firing and the shell traveling up and down. We will also create an explosion in which the factory collapses and debris flies in the air.

**Program 19-4 Artillery Shelling a Factory,  
With Noise and Explosion**

```

NEW
55 REM CLEAR SOUND CHIP
60 FOR J=54272 TO 54296
70 POKE J,0
80 NEXT J
85 REM TURN VOLUME UP FOR GUN FIRING
90 POKE 54296,15
100 PRINT CHR$(147)
105 REM MOVE CURSOR NEAR BOTTOM OF SCREEN
110 FOR J=1 TO 18
120 PRINT
130 NEXT J
135 REM DRAW ARTILLERY GUN
140 GOSUB 4000
145 REM MOVE CURSOR TO START OF TARGET
150 PRINT "{3 UP}";
160 PRINT TAB(30);
165 REM DRAW FACTORY
170 GOSUB 5000
175 REM MAKE SOUND OF GUN FIRING
180 GOSUB 7000
205 REM MOVE CURSOR TO END OF GUN
210 PRINT "{6 UP}" ← No semicolon here.
220 PRINT TAB(5); ← Move right to end of gun.
225 REM TURN VOLUME DOWN FOR SHELL
230 POKE 54296,2
235 REM MOVE SHELL UP DIAGONALLY
240 FOR J=1 TO 12
245 REM DRAW RISING SHELL
250 GOSUB 1000
255 REM SHELL SOUND
260 GOSUB 8000
265 REM ERASE SHELL
270 PRINT "{2 LEFT} {UP}";
280 GOSUB 3000
285 REM MOVE SHELL UP DIAGONALLY
290 PRINT "{LEFT} {2 UP}";
300 NEXT J
305 REM MOVE SHELL DOWN DIAGONALLY
310 FOR J=1 TO 13
315 REM DRAW FALLING SHELL
320 GOSUB 2000
325 REM SHELL SOUND
330 GOSUB 8000
335 REM ERASE SHELL
340 PRINT "{2 LEFT} {UP}";
350 GOSUB 3000
355 REM MOVE SHELL DOWN DIAGONALLY
360 PRINT "{LEFT}";
370 NEXT J
375 REM TURN VOLUME UP FOR EXPLOSION
380 POKE 54296,15
385 REM PRODUCE EXPLOSION
390 GOSUB 6000
400 GOTO 9000

995 REM PICTURE OF RISING SHELL
1000 PRINT "○"; ← Shifted (U, I)
1020 PRINT "└"; ← COMMODORE (Z, X)
1010 PRINT "{2 LEFT} {DOWN}";
1030 RETURN
1995 REM PICTURE OF FALLING SHELL
2000 PRINT "┐"; ← COMMODORE (A, S)
2020 PRINT "○"; ← Shifted (J, K)
2010 PRINT "{2 LEFT} {DOWN}";
2030 RETURN

```

Lines 235 through 380 are almost identical to those in Program 19-2.

```

2995 REM ERASURE OF SHELL
3000 PRINT " "; ←
3020 PRINT " "; ←
3010 PRINT "{2 LEFT} {DOWN}"; ← 2 spaces
3030 RETURN
3995 REM PICTURE OF ARTILLERY GUN
4000 PRINT " "; ←
4020 PRINT " / "; ← 3 spaces, shifted N, COMMODORE G
4040 PRINT " / "; ← 2 spaces, shifted (N, N), space
4060 PRINT " / "; ← Space, shifted (N, N), 2 spaces
4080 PRINT " @ "; ← COMMODORE L, shifted N, 2 spaces
4080 PRINT " @ "; ← 2 @ s, 3 spaces
4010 PRINT "{5 LEFT} {DOWN}";
4030 PRINT "{5 LEFT} {DOWN}";
4050 PRINT "{5 LEFT} {DOWN}";
4070 PRINT "{5 LEFT} {DOWN}";
4090 RETURN
4995 REM PICTURE OF FACTORY
5000 PRINT " | | "; ← Space, COMMODORE (J, L), space
5020 PRINT " | "; ← Shifted N, COMMODORE (T, T), shifted M
5040 PRINT " | "; ← COMMODORE (G, -, +, M)
5060 PRINT " | "; ← Shifted (L, @, L, @)
5010 PRINT "{4 LEFT} {DOWN}";
5030 PRINT "{4 LEFT} {DOWN}";
5050 PRINT "{4 LEFT} {DOWN}";
5070 RETURN

5995 REM EXPLOSION
5998 REM DESTROY CHIMNEYS
6000 PRINT "{DOWN} {RIGHT}";
6010 PRINT " ■ "; ← COMMODORE (I, B)
6015 REM SEND UP SOME DEBRIS
6020 PRINT "{3 LEFT} {2 UP}";
6030 PRINT " ■ ■ "; ← COMMODORE D, space, COMMODORE D
6035 REM MAKE SOME EXPLOSION NOISE
6040 GOSUB 7000
6045 REM CRUMBLE ROOF OF BUILDING
6050 PRINT "{3 LEFT} {3 DOWN}";
6060 PRINT " ^ | "; ← Shifted (F, N, M, F)
6065 REM MAKE MORE EXPLOSION NOISE
6070 GOSUB 7000
6075 REM SEND UP MORE FRAGMENTS
6080 PRINT "{5 LEFT} {4 UP}";
6090 PRINT " ■ ■ ■ ■ "; ← COMMODORE (F, F, £, D, D)
6095 REM MAKE MORE EXPLOSION NOISE
6100 GOSUB 7000
6105 REM COLLAPSE MIDDLE OF BUILDING
6110 PRINT "{4 LEFT} {5 DOWN}";
6120 PRINT " \ / "; ← Shifted M, COMMODORE (£, £), shifted N
6125 REM MAKE MORE EXPLOSION NOISE
6130 GOSUB 7000
6135 REM SEND UP A MUSHROOM CLOUD
6140 PRINT "{5 LEFT} {7 UP}";
6150 PRINT " ■ ■ ■ ■ "; ← COMMODORE (£, +, +, +, £)
6155 REM MAKE MORE EXPLOSION NOISE
6160 GOSUB 7000
6165 REM COLLAPSE REST OF BUILDING
6170 PRINT "{4 LEFT} {8 DOWN}";
6180 PRINT " | \ | "; ← Shifted (B, I, O, B)
6190 GOSUB 7000
6200 RETURN
6995 REM PRODUCE EXPLOSION SOUND EFFECTS
7000 POKE 54277,16*4+12:REM LOW ATTACK, HIGH DECAY
7010 POKE 54276,129:REM START NOISE WAVEFORM
7020 POKE 54273,5
7030 POKE 54272,71
7040 FOR L=1 TO 500:NEXT L
7050 POKE 54276,128:REM STOP NOISE WAVEFORM
7060 RETURN
7995 REM PRODUCE SHELL SOUND
8000 FOR K=1 TO 5

```

```

8010 POKE 54277,16*0+10:REM MINIMUM ATTACK, MEDIUM DECAY
8020 POKE 54276,17:REM START WAVEFORM 1
8030 POKE 54273,23:REM OCTAVE 4, F-SHARP
8040 POKE 54272,181
8050 FOR L=1 TO 5:NEXT L
8060 POKE 54276,16:REM STOP WAVEFORM 1
8070 NEXT K
8080 RETURN
9000 PRINT CHR$(147)
RUN

```

When creating a complex scene like the explosion in lines 6000 through 6200, a paper sketch is generally insufficient. The timing and order are significant, and you simply cannot put those on paper. Furthermore, one incorrect cursor instruction will distort the entire picture and will be difficult to trace on the screen. One approach is to freeze the scene after each printed line. For example, to see if lines 6000 and 6010 are correct, temporarily insert

```
6012 GOTO 6012
```

This statement branches to itself, creating an endless loop; you must press the STOP key to regain control. Either 6012 STOP or 6012 END would actually terminate the program and return control to you. You could then resume it later by typing CONT (continue). Unfortunately, both STOP and END make the computer print a message on the screen, which either distorts the picture or destroys part of it. Remember to remove line 6012 (by typing 6012 and RETURN) after you have checked the scene.

One unrealistic aspect of Program 19-4 is that the shell's velocity is constant. In the real world, of course, gravity slows the shell down as it rises, and speeds it up as it falls. We can model this effect by changing the duration in the shell's sound routine, since that determines its speed. We will introduce an upper limit M in line 8050, and control it as follows:

1. Line 225 makes M initially 5.
2. Line 295 (inside the rising loop) adds 1 to M, thus slowing the shell down as it rises.
3. Line 365 (inside the falling loop) subtracts 1 from M, thus speeding the shell up as it falls.

The new lines are

```

222 REM INITIALIZE SHELL SPEED CONTROL
225 M=3

292 REM REDUCE SHELL SPEED AS IT RISES
295 M=M+1

362 REM INCREASE SHELL SPEED AS IT FALLS
365 M=M-1

8045 REM CONTROL SHELL SPEED WITH LOOP LIMIT
8050 FOR L=1 TO M

```



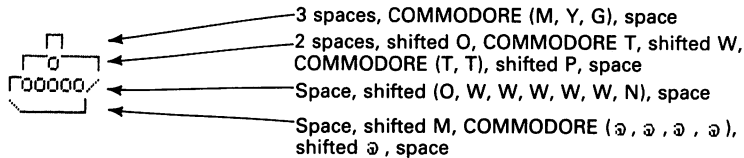
# 20

## SIMULTANEOUS MOTION IN TWO DIRECTIONS

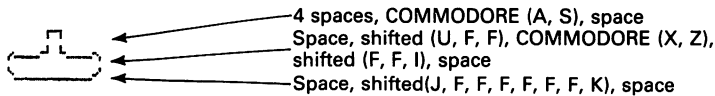
So far, we have allowed motion in only one direction at a time. Now we want to move two objects in different directions simultaneously. We will have a ship travel right on the ocean's surface, while a submarine travels left underneath it. We can move both of them using TABs inside a FOR-NEXT loop controlled by

```
FOR C=1 TO 25
```

TAB(C) in front of each line of the ship moves it right, while TAB(26-C) in front of each line of the submarine moves it left. The ship drawing is



The submarine drawing is



We will start by moving the ship and submarine without any background. The top of the submarine is seven lines below the bottom of the ship. We have grouped functions into sets of line numbers in the main program and left many numbers for later expansion.

## Program 20-1 Ship and Submarine Moving in Different Directions

```

NEW
100 PRINT CHR$(147)
105 REM MOVE CURSOR TO TOP OF SHIP
110 PRINT "{5 DOWN}";
195 REM MOVE SHIP AND SUBMARINE
200 FOR C=1 TO 25
295 REM DRAW SHIP
300 GOSUB 1000
310 FOR K=1 TO 100:NEXT K
395 REM MOVE CURSOR TO TOP OF SUBMARINE
400 PRINT "{7 DOWN}";
405 REM DRAW SUBMARINE
410 GOSUB 2000
420 FOR K=1 TO 100:NEXT K
495 REM MOVE CURSOR BACK TO TOP OF SHIP
500 PRINT "{14 UP}";
600 NEXT C
610 GOTO 9000
995 REM PICTURE OF SHIP
1000 PRINT TAB(C) "  " " 3 spaces
1010 PRINT TAB(C) "  O  " "
1020 PRINT TAB(C) "  OOOO  " "
1030 PRINT TAB(C) "  " "
1040 RETURN
1995 REM PICTURE OF SUBMARINE
2000 PRINT TAB(26-C) "  " " 4 spaces
2010 PRINT TAB(26-C) "  O  " "
2020 PRINT TAB(26-C) "  OOOO  " "
2030 RETURN
9000 PRINT CHR$(147)
RUN

```

We can use graphics characters to simulate rolling ocean waves. Two useful characters are the triangles (the left-hand graphics character on the \* key and the right-hand graphics character on the £ key). We can produce the rolling effect as follows:

1. Print a line consisting of eighteen alternating COMMODORE \*'s and shifted £'s, starting with a COMMODORE \*.
2. Move the cursor up a line.
3. Print a line consisting of eighteen alternating shifted £'s and COMMODORE \*'s, starting with a shifted £.

Enter the wave lines carefully, since you must alternate one finger between COMMODORE and SHIFT while you alternate another between \* and £. You may want to keep count on a piece of paper to ensure that you enter eighteen pairs of characters.

We can also use the following pictures to draw birds with flapping wings:

```

  O  ← 2 spaces, shifted M, COMMODORE @ , shifted W,
  OOO ← COMMODORE @ , shifted N, 2 spaces, shifted (C, W, C)

  O  ← 2 spaces, shifted N, COMMODORE T, shifted W,
  OOO ← COMMODORE T, shifted M, 2 spaces, shifted (U, W, I)

```

## Program 20-2 Ship, Submarine, Ocean Waves, and Birds

Do not type NEW since we want to retain Program 20-1.

```

105 REM MOVE CURSOR TO BIRDS
110 PRINT "{2 DOWN}";
195 REM MOVE SHIP, SUBMARINE, BIRDS, AND WAVES
205 REM DRAW BIRDS WITH WINGS IN ONE POSITION
210 PRINT TAB(10)" \O/ " " " 2 spaces
220 FOR K=1 TO 200:NEXT K No semicolon here.
225 REM DRAW BIRDS WITH WINGS IN ANOTHER POSITION
230 PRINT "{UP}"; 2 spaces
240 PRINT TAB(10)" /O " " " No semicolon here.
245 REM MOVE CURSOR TO TOP OF SHIP
250 PRINT "{2 DOWN}";
315 REM DRAW OCEAN WAVES IN ONE POSITION 18 pairs of COMMODORE *s and
320 PRINT "*****" shifted £s You should see 18
395 REM MOVE CURSOR TO TOP OF SUBMARINE upside-down light blue pyramids.
400 PRINT "{6 DOWN}";
425 REM MOVE CURSOR BACK TO OCEAN SURFACE
430 PRINT "{10 UP}";
435 REM DRAW OCEAN WAVES IN ANOTHER POSITION 18 pairs of shifted £s and
440 PRINT "*****"; COMMODORE *s. You should see
495 REM MOVE CURSOR BACK TO BIRDS 18 dark-blue pyramids.
500 PRINT "{8 UP}";
RUN

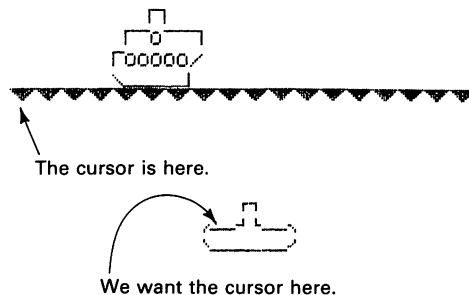
```

In creating such a complex drawing, you may end up with cursor errors even if you work from a good sketch. The result is usually duplicate pictures appearing all over the screen because the computer is starting each iteration in the wrong place. One way to check parts of the drawings, as we noted in the last chapter, is to temporarily insert statements such as

```
305 GOTO 305
```

into the program. This kind of endless loop halts the computer without affecting the picture. You must press STOP and remove the loop (i.e., type 305, then RETURN) before continuing.

Let us now make the submarine fire a projectile when the ship reaches column 13. At that time the screen looks like



The pictures of the ship, submarine, birds, and waves do not change. We must add a new picture for the launch sequence, and make the computer draw it only when C=12. We will start the launch sequence in line 3000. The only addition to the main program is

```
445 REM FIRE PROJECTILE WHEN SHIP REACHES COLUMN 13
450 IF C=12 THEN GOSUB 3000
```

The subroutine that fires the projectile must:

1. Move the cursor to the top left corner of the submarine.
2. Use the logic from Program 7-7 to move the projectile up the screen.

### **Program 20-3 Submarine Firing Projectile**

```
Do not enter NEW.
445 REM FIRE PROJECTILE WHEN SHIP REACHES COLUMN 13
450 IF C=12 THEN GOSUB 3000
2990 REM FIRE PROJECTILE UP THE SCREEN
2995 REM MOVE CURSOR TO TOP LEFT OF SUBMARINE
3000 PRINT "{6 DOWN}";
3010 PRINT TAB(28-C);
3015 REM SEND PROJECTILE UP
3020 FOR R=1 TO 17
3025 REM DRAW PROJECTILE
3030 PRINT "◆"; ← Shifted Z
3040 FOR K=1 TO 200:NEXT K
3045 REM ERASE PROJECTILE
3050 PRINT "{LEFT}";
3060 PRINT " "; ← 1 space here
3065 REM MOVE PROJECTILE UP ONE LINE
3070 PRINT "{LEFT} {UP}";
3080 NEXT R
3085 REM MOVE CURSOR BACK TO OCEAN SURFACE
3090 PRINT "{10 DOWN}"; ← No semicolon here so cursor
3100 RETURN moves to left edge
RUN
```

The submarine now fires a projectile when C is 12. Note how the action stops, since the computer is no longer executing the parts of the program that redraw and thus move the birds, ship, waves, and submarine. The projectile breaks through the ocean waves (leaving a hole in them), scores a direct hit on the center of the ship, rips off the end of a bird's left wing, and flies off the screen. Fortunately nature (and the ship's crew) have marvelous regenerative powers, so the ocean, ship, and bird recover their original shapes immediately and the action continues.

We can easily let the player control when the submarine fires its projectile. We will use the F key for this purpose, as in earlier programs. The additional lines are

```
445 REM CHECK KEYBOARD
450 GET K$
455 REM F KEY FIRES PROJECTILE
460 IF K$="F" THEN GOSUB 3000
```

Now you can make the submarine fire several projectiles. You do not have to rush, since nothing moves while a projectile is on the screen. See if you can score a direct hit on

a bird's body, rather than just ripping off its wing. Be careful, however, not to practice this feat around Audubon Society members.

In fact, you can use the computer's keyboard memory to make the submarine fire projectiles continuously. Simply press F several times. While the computer will not respond to the extra Fs immediately, it will remember them. Once the projectile disappears from the screen, the submarine will move left one column and fire again. It will continue this routine of waiting for a projectile to disappear, moving left one column, and firing until the program ends or until the computer has responded to all the extra Fs.

The complete program with the player controlling the firing is as follows:

**Program 20-4 Ocean, Birds, Ship, and Submarine,  
With Launch**

```

NEW
100 PRINT CHR$(147)
105 REM MOVE CURSOR TO BIRDS
110 PRINT "{2 DOWN}";
195 REM MOVE SHIP, SUBMARINE, BIRDS, AND WAVES
200 FOR C=1 TO 25
205 REM DRAW BIRDS WITH WINGS IN ONE POSITION 2 spaces
210 PRINT TAB(10)" \O_/_ " ← No semicolon here.
220 FOR K=1 TO 100:NEXT K
225 REM DRAW BIRDS WITH WINGS IN ANOTHER POSITION 2 spaces
230 PRINT "{UP}";
240 PRINT TAB(10)" /O_/_ " ← No semicolon here.
245 REM MOVE CURSOR TO TOP OF SHIP
250 PRINT "{2 DOWN}";
295 REM DRAW SHIP
300 GOSUB 1000
310 FOR K=1 TO 100:NEXT K
315 REM DRAW OCEAN WAVES IN ONE POSITION ← 18 pairs of COMMODORE *s and
320 PRINT "*****" ← shifted £s. You should see 18
395 REM MOVE CURSOR TO TOP OF SUBMARINE
400 PRINT "{6 DOWN}";
405 REM DRAW SUBMARINE
410 GOSUB 2000
420 FOR K=1 TO 100:NEXT K
425 REM MOVE CURSOR BACK TO OCEAN SURFACE
430 PRINT "{10 UP}";
435 REM DRAW OCEAN WAVES IN ANOTHER POSITION ← 18 pairs of shifted £s and
440 PRINT "*****" ← COMMODORE *s.
445 REM CHECK KEYBOARD
450 GET K$
455 REM F KEY FIRES PROJECTILE
460 IF K$="F" THEN GOSUB 3000
495 REM MOVE CURSOR BACK TO BIRDS
500 PRINT "{8 UP}";
600 NEXT C
610 GOTO 9000
995 REM PICTURE OF SHIP
1000 PRINT TAB(C)" □ " ← 3 spaces, COMMODORE (M, Y, G), space
1010 PRINT TAB(C)" □ " ← 2 spaces, shifted O, COMMODORE T, shifted W,
COMMODORE (T, T), shifted P, space
1020 PRINT TAB(C)" □00000/ " ← Space, shifted (O, W, W, W, W, W, N), space
1030 PRINT TAB(C)" □ " ← Space, shifted M, COMMODORE ( @, @, @, @),
shifted @, space
1040 RETURN

```

```

1995 REM PICTURE OF SUBMARINE
2000 PRINT TAB(26-C) "      "
2010 PRINT TAB(26-C) "      "
2020 PRINT TAB(26-C) "      "
2030 RETURN
2990 REM FIRE PROJECTILE UPWARD
2995 REM MOVE CURSOR TO TOP LEFT OF SUBMARINE
3000 PRINT "{6 DOWN}";
3010 PRINT TAB(28-C);
3015 REM MOVE PROJECTILE UP
3020 FOR R=1 TO 17
3025 REM DRAW PROJECTILE
3030 PRINT "◆";
3040 FOR K=1 TO 200:NEXT K
3045 REM ERASE PROJECTILE
3050 PRINT "{LEFT}";
3060 PRINT " ";
3065 REM MOVE PROJECTILE UP ONE LINE
3070 PRINT "{LEFT} {UP}";
3080 NEXT R
3085 REM MOVE CURSOR BACK TO OCEAN SURFACE
3090 PRINT "{10 DOWN}";
3100 RETURN
9000 PRINT CHR$(147)
RUN

```

← 4 spaces, COMMODORE (A, S), space  
 ← Space, shifted (U, F, F), COMMODORE (X, Z),  
 ← shifted (F, F, I), space  
 ← Space, shifted (J, F, F, F, F, F, K), space  
 ← Shifted Z  
 ← 1 space here  
 ← No semicolon here, so cursor  
 ← moves to left edge.

As with the shell in Chapter 19, gravity should reduce the projectile's speed. We can simulate gravity by increasing the time delay (line 3040) as the projectile rises. This increases the time between pictures and thus reduces the speed. Let us call the upper limit T, start T at 150, and increase it by 20 after each iteration. The required changes are

```

3011 REM INITIALIZE TIME BETWEEN PICTURES
3012 T=150
3035 REM WAIT BETWEEN PICTURES
3040 FOR K=1 TO T:NEXT K
3072 REM SLOW PROJECTILE BY INCREASING TIME BETWEEN PICTURES
3074 T=T+20

```

This is similar to the way we changed the shell's speed in Chapter 19, although here we are varying the length of a delay loop rather than a sound-generation subroutine.

An additional complication is that the projectile moves from water into air. Clearly, water should exert more drag than air, since water is much thicker (more viscous). We can simulate that difference by reducing the change in T after the projectile reaches the surface (six lines up). To do this, we replace line 3074 with

```

3074 IF R<6 THEN T=T+20
3076 IF R>=6 THEN T=T+10

```

← >= means "greater than or equal to".

We can also introduce sound into the silent ocean scene. To produce an explosion sound when the submarine fires, and a sound to accompany the projectile, add

```

55 REM CLEAR SOUND CHIP
60 FOR J=54272 TO 54296
70 POKE J,0
80 NEXT J

```

```

3016 REM MAKE A LOUD FIRING SOUND
3017 POKE 54296,15
3018 GOSUB 6000
3019 POKE 54296,2:REM REDUCE VOLUME
3035 REM MAKE NOISE
3040 GOSUB 7000

5995 REM SOUND OF SUBMARINE FIRING PROJECTILE
6000 POKE 54277,16*4+12:REM LOW ATTACK, HIGH DECAY
6010 POKE 54276,129:REM START NOISE WAVEFORM
6020 POKE 54273,5
6030 POKE 54272,71
6040 FOR L=1 TO 500:NEXT L
6050 POKE 54276,128:REM STOP NOISE WAVEFORM
6060 RETURN

6995 REM PROJECTILE SOUND
7000 FOR K=1 TO 5
7010 POKE 54277,16*0+10:REM MINIMUM ATTACK, MEDIUM DECAY
7020 POKE 54276,17:REM START WAVEFORM #1
7030 POKE 54273,23:REM OCTAVE 4, F-SHARP
7040 POKE 54272,181
7050 FOR L=1 TO 5:NEXT L
7060 POKE 54276,16:REM STOP WAVEFORM #1
7070 NEXT K
7080 RETURN

```

You may want to modify the projectile sound routine so that it can be used to simulate the effects of gravity and water's resistance to motion.

# 21

## MUSICAL INTERLUDE

The computer can produce music as well as noise to accompany pictures. We already know how to create musical notes using the values in Appendix A. The following program plays the first three notes (C, D, and E) in octave 3.

### *Program 21-1 Musical Quarter Notes*

```
NEW
55 REM CLEAR SOUND CHIP
60 FOR J=54272 TO 54296
70 POKE J,0
80 NEXT J
90 POKE 54296,15
100 PRINT CHR$(147)
110 PRINT TAB(5)"FIRST THREE NOTES IN OCTAVE 3"
120 POKE 54277,16*0+10:REM MINIMUM ATTACK, MEDIUM DECAY
130 POKE 54276,17:REM START WAVEFORM 1
135 REM PLAY C FROM OCTAVE 3
140 POKE 54273,8
150 POKE 54272,97
160 FOR L=1 TO 250:NEXT L
170 POKE 54276,16:REM STOP WAVEFORM 1
175 REM PLAY D FROM OCTAVE 3
180 POKE 54276,17:REM START WAVEFORM 1
190 POKE 54273,9
200 POKE 54272,104
210 FOR L=1 TO 250:NEXT L
220 POKE 54276,16:REM STOP WAVEFORM 1
225 REM PLAY E FROM OCTAVE 3
230 POKE 54276,17:REM START WAVEFORM 1
240 POKE 54273,10
250 POKE 54272,143
260 FOR L=1 TO 250:NEXT L
270 POKE 54276,16:REM STOP WAVEFORM 1
280 PRINT CHR$(147)
RUN
```



Only the high and low frequency POKEs change for different notes. Thus we need a way to repeat all the POKEs but bring in different high- and low-frequency values each time. To do this requires two new BASIC statements: DATA, which holds information until the program needs it, and READ, which obtains information from a DATA statement.

DATA statements simply consist of a sequence of values (either numbers or characters) separated by commas. For example, to hold the HF and LF values from Program 21-1, we would use

```
DATA 8,97,9,104,10,143
```

READ statements take values from DATA statements in the usual left-to-right order. For example,

```
READ HF,LF
```

at the start of a program would take the first two values from DATA statements and call them HF and LF. If the first DATA statement were the one above, these values would be 8 for HF and 97 for LF. The next READ statement would take up where this one left off. That is, the next

```
READ HF,LF
```

would set HF to 9 and LF to 104. The following program shows how this works:

```
NEW
100 PRINT CHR$(147)
110 FOR J=1 TO 6
120 READ V
130 PRINT V
140 NEXT J
150 DATA 8,97,9,104,10,143
RUN
```

This program reads and prints the six values consecutively, one at a time. The FOR NEXT loop determines how many times the computer executes READ V. The following version READs and PRINTs two items in each iteration.

```
NEW
100 PRINT CHR$(147)
110 FOR J=1 TO 3
120 READ HF,LF
130 PRINT HF,LF
140 NEXT J
150 DATA 8,97,9,104,10,143
RUN
```

This program READs the DATA statement three times, taking two values (HF and LF) from it each time.

Now let us use READ and DATA to play the eight notes in the third octave; note how much shorter this program is than Program 21-1.

## Program 21-2 Musical Scale in Octave 3

```
NEW
55 REM CLEAR SOUND CHIP AND SET VOLUME
60 FOR J=54272 TO 54296
70 POKE J,0
80 NEXT J
90 POKE 54296,15
100 PRINT CHR$(147)
110 PRINT TAB(10)"THE NOTES IN OCTAVE 3"
120 POKE 54277,16*0+10:REM MINIMUM ATTACK, MEDIUM DECAY
130 FOR J=1 TO 8
140 POKE 54276,17:REM START WAVEFORM 1
145 REM READ HIGH AND LOW FREQUENCY FOR NOTE
150 READ HF,LF
160 POKE 54273,HF
170 POKE 54272,LF
175 REM LENGTH IS QUARTER NOTE
180 FOR L=1 TO 250:NEXT L
190 POKE 54276,16:REM STOP WAVEFORM 1
200 NEXT J
210 DATA 8,97,9,104,10,143,11,48,12,143,14,24,15,210,16,195
RUN
```

Note that we borrowed the ending C (16,195) from octave 4 in Appendix A to complete the octave. We have put the DATA statement at the end to keep it out of the way, but it can actually go anywhere in the program. The computer simply skips over DATA statements when it reaches them.

To repeat the notes, we must make the computer return to the beginning of the DATA statement. The statement that does this is RESTORE. Thus, to play the same eight notes over and over, add

```
220 RESTORE
230 GOTO 130
```

If you forget RESTORE, the program will stop and report

```
?OUT OF DATA ERROR IN 150
```

To produce a different sound, change the waveform in lines 140 and 190—that is

```
140 POKE 54276,33:REM START WAVEFORM 2
190 POKE 54276,32:REM STOP WAVEFORM 2
```

To hear the effects of duration, change the length of each note by changing the upper limit in line 180 to 1000 (a whole note). To hear the effects of attack and decay, change the value in line 120 to  $16*8+8$  (medium attack, medium decay), then to  $16*15+0$  (maximum attack, minimum decay). Experimenting with these values will give you a practical understanding of how to create music on the Commodore 64.

Now let us consider the classic song "Old MacDonald's Farm." We will split it into two parts, to be played sequentially. Appendix B contains a summary of music for those who cannot read it.

**PART A**



**PART B**



Using Appendix B, we convert the notes to the letters A through G. Let's look at the first four measures. The  $\flat$  at the left tells us which notes are flat (a half-tone lower than the natural note). Here all Bs and Es are flat unless specifically marked otherwise. A minus sign (-) following a note's letter designation means it is flat.



Notes                                    E - E - E - B -    C C B -            G G F F    E -    B -

After converting the notes from a score to the proper letters, we then find the length of each note



1/4 1/4 1/4 1/4    1/4 1/4 1/2            1/4 1/4 1/4 1/4    1/2    1/4

We next find the octave for each note



Octave                                    3 3 3 2    3 3 2            3 3 3 3    3    2

We now have all the information we need to program these four measures. The music looks like



Octave	3 3 3 2	3 3 2	3 3 3 3	3	2
Notes	E- E- E- B-	C C B-	G G F F	E-	B-
Length	1/4 1/4 1/4 1/4	1/4 1/4 1/2	1/4 1/4 1/4 1/4	1/2	1/4

Using Appendix A, the score becomes



HF values	9 9 9 7	8 8 7	12 12 11 11	9	7
LF values	247 247 247 119	97 97 119	143 143 48 48	247	119
Length	4 4 4 4	4 4 2	4 4 4 4	2	4

We have inverted the lengths to make the values easier to enter. That is, a value of 4 represents a length of  $\frac{1}{4}$ , a value of 2 a length of  $\frac{1}{2}$ , etc. The do-nothing loop that determines duration must divide 1000 (a full note) by the inverted length; that is, the loop will be

```
FOR L=1 TO 1000/LE:NEXT L
```

To hold the frequency and length values, we need the following DATA statements:

```
DATA 9,247,4,9,247,4,9,247,4,7,119,4
DATA 8,97,4,8,97,4,7,119,2
DATA 12,143,4,12,143,4,11,48,4
DATA 11,48,4,9,247,2,7,119,4
```

Since the computer handles all DATA statements as a unit, we can divide the values into many lines. The computer will proceed to the next DATA statement as soon as it finishes the previous one.

The READ statement that obtains an inverted (reciprocal) length value as well as frequency values is

```
READ HF,LF,LE
```

Let's look at the next four measures.



Octave	3	3	3	2	3	3	2	3	3	3	3	3	0
Notes	E-	E-	E-	B-	C	C	B-	G	G	F	F	E-	0
Length	1/4	1/4	1/4	1/4	1/4	1/4	1/2	1/4	1/4	1/4	1/4	3/4	1/4

The  $\}$  is called a *rest*, that is, a pause in the music. Here, the symbol means to pause for one fourth of a second (same time given a quarter note) before continuing. To include the rest, we must add

- 0
- Indicates a pause by POKEing zeros into the sound chip.
- 4 Inverted time value.

at the end.

Using Appendix A, we get



HF values	9	9	9	7	8	8	7	12	12	11	11	9	0
LF values	247	247	247	119	97	97	119	143	143	48	48	247	0
Length	4	4	4	4	4	4	2	4	4	4	4	1.33	4

This produces the DATA statements

```
DATA 9,247,4,9,247,4,9,247,4
DATA 7,119,4,8,97,4,8,97,4
DATA 7,119,2,12,143,4,12,143,4
DATA 11,48,4,11,48,4,9,247,1.33,0,0,4
```

Since the song has 26 notes (including the rest), the FOR NEXT loop to read the DATA statements becomes

```
FOR J=1 TO 26
```

## Program 21-3 "Old MacDonald's Farm"

```

NEW
55 REM CLEAR SOUND CHIP AND SET VOLUME
60 FOR J=54272 TO 54296
70 POKE J,0
80 NEXT J
90 POKE 54296,15
100 PRINT CHR$(147)
110 PRINT TAB(10)"OLD MACDONALD'S FARM"
120 POKE 54277,16*0+10:REM MINIMUM ATTACK, MEDIUM DECAY
130 FOR J=1 TO 26
140 POKE 54276,17:REM START WAVEFORM 1
145 REM OBTAIN HIGH FREQUENCY, LOW FREQUENCY, RECIPROCAL LENGTH
150 READ HF,LF,LE
160 POKE 54273,HF
170 POKE 54272,LF
175 REM CONVERT RECIPROCAL LENGTH TO DELAY
180 FOR L=1 TO 1000/LE:NEXT L
190 POKE 54276,16:REM STOP WAVEFORM 1
200 NEXT J
295 REM FIRST 4 MEASURES
300 DATA 9,247,4,9,247,4,9,247,4,7,119,4
310 DATA 8,97,4,8,97,4,7,119,2
320 DATA 12,143,4,12,143,4,11,48,4
330 DATA 11,48,4,9,247,2,7,119,4
335 REM SECOND 4 MEASURES
340 DATA 9,247,4,9,247,4,9,247,4
350 DATA 7,119,4,8,97,4,8,97,4
360 DATA 7,119,2,12,143,4,12,143,4
370 DATA 11,48,4,11,48,4,9,247,1.33,0,0,4
380 PRINT CHR$(147)
RUN

```

Be sure to press RETURN at the end of this line, even though the cursor has already gone down to the next empty line. This is a problem case because it is exactly 40 characters long. If you have entered line 300 correctly, there will be a blank line between it and line 310 in the listing.

To increase the tempo, change 1000 in line 180 to 500 or 250. By experimenting with this value, you can speed the song up or slow it down. A high value such as 2000 produces a good version to play at Old MacDonald's foreclosure auction. Changing the attack and decay values or the waveform produces a different rendition of the tune.

The American classic "Yankee Doodle Dandy" has the musical score

### PART A



### PART B



If we treat “Yankee Doodle Dandy” the same way we did “Old MacDonald,” we get



3	3	3	3	3	3	3	3	3	3	3	3	3	3
E-	E-	F	G	E-	G	F	D	E-	E-	F	G	E-	D
1/4	1/4	1/4	1/4	1/4	1/4	1/4	1/4	1/4	1/4	1/4	1/4	1/2	1/4

This produces the values



9	9	11	12	9	12	11	9	9	9	11	12	9	9
247	247	48	143	247	143	48	104	247	247	48	143	247	104
4	4	4	4	4	4	4	4	4	4	4	4	2	4

for the first four measures.

The second four measures are



giving



3	3	3	3	3	3	3	3	3	2	2	3	3	3
E-	E-	F	G	A-	G	F	E-	D	B-	C	D	E-	E-
1/4	1/4	1/4	1/4	1/4	1/4	1/4	1/4	1/4	1/4	1/4	1/4	1/2	1/2

These are translated into



```

          9 9 11 12      13 12 11 9      9 7 8 9      9 9
          247 247 48 143    78 143 48 247    104 119 97 104    247 247
          4 4 4 4      4 4 4 4      4 4 4 4      2 2
  
```

The DATA statements then become

```

DATA 9,247,4,9,247,4,11,48,4
DATA 12,143,4,9,247,4,12,143,4
DATA 11,48,4,9,104,4,9,247,4
DATA 9,247,4,11,48,4,12,143,4
DATA 9,247,2,9,104,4,0,0,4
  
```

for the first four measures and

```

DATA 9,247,4,9,247,4,11,48,4
DATA 12,143,4,13,78,4,12,143,4
DATA 11,48,4,9,247,4,9,104,4
DATA 7,119,4,8,97,4,9,104,4
DATA 9,247,2,9,247,2
  
```

for the second four.

There are 29 notes in the tune, so we use

```
FOR J=1 TO 29
```

to read the DATA statements.

### **Program 21-4 "Yankee Doodle Dandy"**

```

NEW
55 REM CLEAR SOUND CHIP AND SET VOLUME
60 FOR J=54272 TO 54296
70 POKE J,0
80 NEXT J
90 POKE 54296,15
100 PRINT CHR$(147)
110 PRINT TAB(9)"YANKEE DOODLE DANDY"
120 POKE 54277,16*0+8:REM MINIMUM ATTACK, MEDIUM DECAY
130 FOR J=1 TO 29
140 POKE 54276,33:REM START WAVEFORM 2
145 REM READ HIGH AND LOW FREQUENCY, INVERTED LENGTH
150 READ HF,LF,LE
160 POKE 54273,HF
170 POKE 54272,LF
175 REM FIND LENGTH OF EACH NOTE
180 FOR L=1 TO 300/LE:NEXT L
190 POKE 54276,32:REM STOP WAVEFORM 2
200 NEXT J
  
```

Divide reciprocal length into 300  
to produce a fast tempo.





```

995 REM DRAW FIGURE WITH BATON HELD OUT
1000 PRINT "  O "; ← 2 spaces, shifted W, 2 spaces
1020 PRINT "  | "; ← 2 spaces, shifted -, 2 spaces
1040 PRINT "  + "; ← Space, COMMODORE F, shifted -, COMMODORE (D, G)
1060 PRINT "  | "; ← 2 spaces, shifted -, 2 spaces
1080 PRINT "  / "; ← Space, shifted N, COMMODORE T, shifted M,
1100 PRINT "  \ "; ← space
1030 PRINT "{5 LEFT}{DOWN}";
1050 PRINT "{5 LEFT}{DOWN}";
1070 PRINT "{5 LEFT}{DOWN}";
1090 RETURN
RUN

```

To accompany animated figures with music, we must first draw the figures, then intersperse the music in the drawing program. For example, we can make a stick figure dance by drawing the following four pictures in the same place. Each shows a different stage of dancing.

#### PICTURE #1

```

O ← 2 spaces, shifted W, 2 spaces
| ← 2 spaces, shifted -, 2 spaces
+ ← Space, shifted (U, +, I), space
| ← 2 spaces, shifted -, 2 spaces
/ ← Space, shifted N, COMMODORE T, shifted M, space
\

```

#### PICTURE #2

```

O ← 2 spaces, shifted W, 2 spaces
| ← 2 spaces, shifted -, 2 spaces
+ ← Space, shifted (F, +, F), space
| ← 2 spaces, shifted -, 2 spaces
/ ← Space, shifted @, COMMODORE T, shifted M, space
\

```

#### PICTURE #3

```

O ← 2 spaces, shifted W, 2 spaces
| ← 2 spaces, shifted -, 2 spaces
+ ← Space, shifted (F, +, K), space
| ← 2 spaces, shifted -, 2 spaces
/ ← Space, shifted N, COMMODORE T, shifted L, space
\

```

#### PICTURE #4

```

O ← 2 spaces, shifted W, 2 spaces
| ← 2 spaces, shifted -, 2 spaces
+ ← Space, shifted (J, +, F), space
| ← 2 spaces, shifted -, 2 spaces
/ ← Space, shifted N, COMMODORE T, shifted M, space
\

```

Note that only the third and fifth lines of the pictures (the arms and legs) change; lines one, two, and four are the same in all four pictures. In fact, lines two and four are identical. All the repetitions and similarities mean that you can use the CRSR keys heavily when entering the pictures.

## Program 21-6 Dancing Stick Figure

```

NEW
PRINT CHR$(147)
105 REM DRAW STAGE FLOOR
110 PRINT "{10 DOWN}";
120 PRINT " "
125 REM MOVE CURSOR TO TOP OF STICK FIGURE
130 PRINT "{6 UP}";
135 REM MAKE STICK FIGURE DANCE
140 FOR J=1 TO 15
150 GOSUB 1000
160 PRINT "{5 LEFT} {4 UP}";
170 FOR K=1 TO 100:NEXT K
180 GOSUB 2000
190 PRINT "{5 LEFT} {4 UP}";
200 FOR K=1 TO 100:NEXT K
210 GOSUB 3000
220 PRINT "{5 LEFT} {4 UP}";
230 FOR K=1 TO 100:NEXT K
240 GOSUB 4000
250 PRINT "{5 LEFT} {4 UP}";
260 FOR K=1 TO 100:NEXT K
270 NEXT J
280 GOTO 9000

995 REM PICTURE #1 OF DANCING STICK FIGURE
1000 PRINT " 0 "; ← 2 spaces, shifted W, 2 spaces
1020 PRINT " | "; ← 2 spaces, shifted -, 2 spaces
1040 PRINT " + "; ← Space, shifted (U, +, I), space
1060 PRINT " | "; ← 2 spaces, shifted -, 2 spaces
1080 PRINT " ^ "; ← Space, shifted N, COMMODORE T, shifted M, space
1010 PRINT "{5 LEFT} {DOWN}";
1030 PRINT "{5 LEFT} {DOWN}";
1050 PRINT "{5 LEFT} {DOWN}";
1070 PRINT "{5 LEFT} {DOWN}";
1090 RETURN

1995 REM PICTURE #2 OF DANCING STICK FIGURE
2000 PRINT " 0 "; ← 2 spaces, shifted W, 2 spaces
2020 PRINT " | "; ← 2 spaces, shifted -, 2 spaces
2040 PRINT " +- "; ← Space, shifted (F, +, F), space
2060 PRINT " | "; ← 2 spaces, shifted -, 2 spaces
2080 PRINT " ^ "; ← Space, shifted @, COMMODORE T, shifted M, space
2010 PRINT "{5 LEFT} {DOWN}";
2030 PRINT "{5 LEFT} {DOWN}";
2050 PRINT "{5 LEFT} {DOWN}";
2070 PRINT "{5 LEFT} {DOWN}";
2090 RETURN

2995 REM PICTURE OF #3 DANCING STICK FIGURE
3000 PRINT " 0 "; ← 2 spaces, shifted W, 2 spaces
3020 PRINT " | "; ← 2 spaces, shifted -, 2 spaces
3040 PRINT " + "; ← Space, shifted (F, +, K), space
3060 PRINT " | "; ← 2 spaces, shifted -, 2 spaces
3080 PRINT " ^ "; ← Space, shifted N, COMMODORE T, shifted L, space
3010 PRINT "{5 LEFT} {DOWN}";
3030 PRINT "{5 LEFT} {DOWN}";
3050 PRINT "{5 LEFT} {DOWN}";
3070 PRINT "{5 LEFT} {DOWN}";
3090 RETURN

3995 REM PICTURE #4 OF DANCING STICK FIGURE
4000 PRINT " 0 "; ← 2 spaces, shifted W, 2 spaces
4020 PRINT " | "; ← 2 spaces, shifted -, 2 spaces
4040 PRINT " + "; ← Space, shifted (J, +, F), space
4060 PRINT " | "; ← 2 spaces, shifted -, 2 spaces
4080 PRINT " ^ "; ← Space, shifted N, COMMODORE T, shifted M, space
4010 PRINT "{5 LEFT} {DOWN}";

```

```

4030 PRINT "{5 LEFT} {DOWN}";
4050 PRINT "{5 LEFT} {DOWN}";
4070 PRINT "{5 LEFT} {DOWN}";
4090 RETURN
9000 PRINT CHR$(147)
RUN

```

To make the stick figure dance across the stage, all we must do is insert

```

262 REM MOVE STICK FIGURE ONE COLUMN RIGHT
265 PRINT "{RIGHT}";

```

To produce music for the stick figure, we must intersperse the statement

```

READ HF,LF,LE

```

throughout Program 21-6. since there are 29 notes, we must READ 29 values from the DATA statement, using

```

140 FOR J=1 TO 7
170 READ HF,LF,LE
171 GOSUB 7000

200 READ HF,LF,LE
201 GOSUB 7000

230 READ HF,LF,LE
231 GOSUB 7000

260 READ HF,LF,LE
261 GOSUB 7000

270 NEXT J

275 READ HF,LF,LE
276 GOSUB 7000

```

The 7000 routine to play the song is

```

7000 POKE 54277,16*0+10:REM MINIMUM ATTACK, MEDIUM DECAY
7010 POKE 54276,33:REM START WAVEFORM 2
7020 POKE 54273,HF
7030 POKE 54272,LF
7040 FOR L=1 TO 1000/LE:NEXT L
7050 POKE 54276,32
7060 RETURN

```



You may note that the applause is not exactly overwhelming. In fact, it sounds like there's only one person clapping (perhaps it's the dancer's mother). If you change the decay in line 8000 to 8, it sounds like more people, but now they're all banging on the doors, trying desperately to escape before the encore.

### **Program 21-8 Stick Figure Dancing to Song, With Bow and Applause**

Do not type NEW; leave Program 21-7 in memory.

```

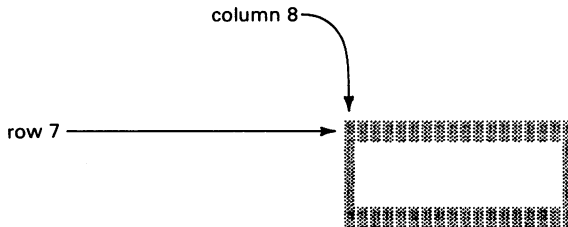
279 REM STAND UP IN TRIUMPH
280 GOSUB 5000
290 GOSUB 8000
300 PRINT "{5 LEFT} {4 UP}";
305 REM BOW RIGHT
310 GOSUB 2000
320 GOSUB 8000
330 PRINT "{5 LEFT} {4 UP}";
335 REM UP AGAIN
340 GOSUB 5000
350 GOSUB 8000
360 PRINT "{5 LEFT} {4 UP}";
365 REM BOW LEFT
370 GOSUB 3000
380 GOSUB 8000
390 PRINT "{5 LEFT} {4 UP}";
395 REM UP AGAIN
400 GOSUB 5000
410 GOSUB 8000
420 GOTO 9000
4995 REM PICTURE #5 OF DANCING STICK FIGURE
5000 PRINT "  O  "; ← 2 spaces, shifted W, 2 spaces
5020 PRINT "  |  "; ← 2 spaces, shifted -, 2 spaces
5040 PRINT "  +  "; ← Space, shifted (J, +, K), space
5060 PRINT "  !  "; ← 2 spaces, shifted -, 2 spaces
5080 PRINT "  ^  "; ← Space, shifted N, COMMODORE T, shifted M, space
5010 PRINT "{5 LEFT} {DOWN}";
5030 PRINT "{5 LEFT} {DOWN}";
5050 PRINT "{5 LEFT} {DOWN}";
5070 PRINT "{5 LEFT} {DOWN}";
5090 RETURN
7995 REM PRODUCE A ROUND OF APPLAUSE FOR THE DANCER
8000 POKE 54277,16*0+2:REM MINIMUM ATTACK, LOW DECAY
8010 FOR K=1 TO 5
8020 POKE 54276,129:REM START NOISE WAVEFORM
8030 POKE 54273,11
8040 POKE 54272,48
8050 FOR L=1 TO 100:NEXT L
8060 POKE 54276,128:REM STOP NOISE WAVEFORM
8070 NEXT K
8080 RETURN
RUN

```

# 22

## RACE CAR GAME

We can use the symbol `COMMODORE +` to draw a simple rectangular racetrack for a car. The track begins in column 8 of row 7 and looks like



We make these `PRINT` lines 7000 through 7070 so that we can later use them as a subroutine in a racing game.

### **Program 22-1 Beginner's Racetrack**

```
NEW
6995 DRAW RACE TRACK
6998 REM START DOWN 7 LINES
7000 PRINT "{7 DOWN}";
7005 REM DRAW TOP
7010 PRINT TAB(8) "*****"
7015 REM DRAW SIDES
7020 PRINT TAB(8) " |" SPC(16) " |"
7030 PRINT TAB(8) " |" SPC(16) " |"
7040 PRINT TAB(8) " |" SPC(16) " |"
7050 PRINT TAB(8) " |" SPC(16) " |"
7055 REM DRAW BOTTOM
7060 PRINT TAB(8) "*****"
PRINT CHR$(147)
RUN
```

18 `COMMODORE +`s. Line 7060 is the same.

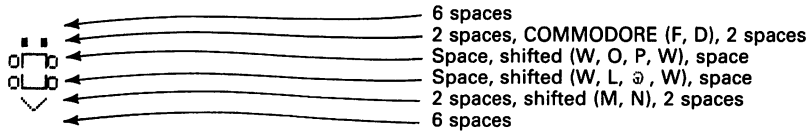
`SPC(N)` skips N spaces forward; that is, it moves the cursor N columns right.

`COMMODORE +`. Lines 7030, 7040, and 7050 are the same.

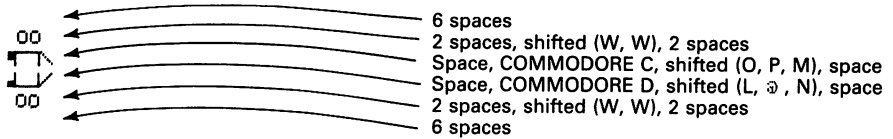
18 `COMMODORE +`s. Be sure to type `SHIFT` and 2, not `COMMODORE` and 2 to enter the closing quotation mark.

To race a car around the track, we need the following pictures of it moving left, right, up, and down.

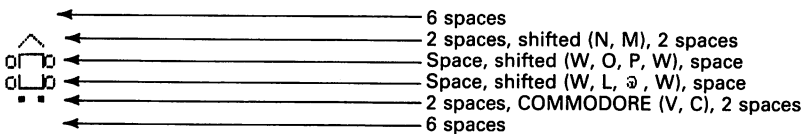
#### CAR MOVING DOWN



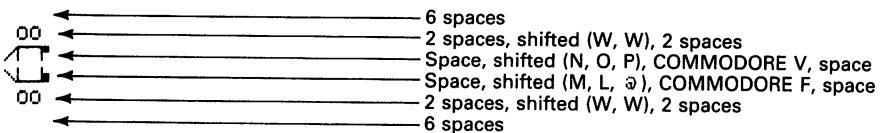
#### CAR MOVING RIGHT



#### CAR MOVING UP



#### CAR MOVING LEFT



We will use the same control keys as in Program 18-1, that is,

- A — move car left
- S — move car right
- W — move car up
- Z — move car down
- E — end game

Like the rocket in Program 18-5, the car continues in its current direction until the player presses a different direction key. The car starts moving left from line 0, column 20 (top center).



## Program 22-2 Car Racing Around Track

```

NEW
100 PRINT CHR$(147); ← Put semicolon here.
105 REM DRAW TRACK
110 GOSUB 7000
120 PRINT CHR$(19); ← Put semicolon here.
125 REM START CAR IN FIRST ROW, 20TH COLUMN
130 R=1
140 C=20
150 PRINT TAB(19);
160 DI$="LEFT"
170 GOSUB 3000
175 REM RETURN CURSOR TO START OF PICTURE
180 PRINT "{6 LEFT} {5 UP}";
190 FOR K=1 TO 100:NEXT K
195 REM EXAMINE KEYBOARD
200 GET K$
205 REM E KEY ENDS GAME
210 IF K$="E" THEN 9000

215 REM W, A, S, AND Z KEYS CHANGE DIRECTIONS
218 REM W KEY MAKES CAR MOVE UP
220 IF K$="W" THEN DI$="UP"
225 REM Z KEY MAKES CAR MOVE DOWN
230 IF K$="Z" THEN DI$="DOWN"
235 REM A KEY MAKES CAR MOVE LEFT
240 IF K$="A" THEN DI$="LEFT"
245 REM S KEY MAKES CAR MOVE RIGHT
250 IF K$="S" THEN DI$="RIGHT"
255 REM MOVE CAR IN APPROPRIATE DIRECTION
260 IF DI$<>"UP" THEN 300
265 REM MOVE CAR UP ONE ROW IF POSSIBLE
270 IF R>1 THEN R=R-1:PRINT "{UP}";
275 REM DRAW CAR MOVING UP
280 GOSUB 1000
290 GOTO 180
300 IF DI$<>"DOWN" THEN 340
305 REM MOVE CAR DOWN ONE ROW IF POSSIBLE
310 IF R<19 THEN R=R+1:PRINT "{DOWN}";
315 REM DRAW CAR MOVING DOWN
320 GOSUB 2000
330 GOTO 180
340 IF DI$<>"LEFT" THEN 380
345 REM MOVE CAR LEFT ONE COLUMN IF POSSIBLE
350 IF C>1 THEN C=C-1:PRINT "{LEFT}";
355 REM DRAW CAR MOVING LEFT
360 GOSUB 3000
370 GOTO 180
380 IF DI$<>"RIGHT" THEN 180
385 REM MOVE CAR RIGHT ONE COLUMN IF POSSIBLE
390 IF C<34 THEN C=C+1:PRINT "{RIGHT}";
395 REM DRAW CAR MOVING RIGHT
400 GOSUB 4000
410 GOTO 180

995 REM PICTURE OF CAR MOVING UP
1000 PRINT " "; ← 6 spaces
1020 PRINT " "; ← 2 spaces, shifted (N, M), 2 spaces
1040 PRINT "  O  O "; ← Space, shifted (W, O, P, W), space
1060 PRINT "  O  O "; ← Space, shifted (W, L, @, W), space
1080 PRINT "  .  . "; ← 2 spaces, COMMODORE (V, C), 2 spaces
1100 PRINT " "; ← 6 spaces
1010 PRINT "{6 LEFT} {DOWN}";
1030 PRINT "{6 LEFT} {DOWN}";
1050 PRINT "{6 LEFT} {DOWN}";
1070 PRINT "{6 LEFT} {DOWN}";
1090 PRINT "{6 LEFT} {DOWN}";
1110 RETURN

```



yourself frantically pressing Q or X or accidentally pushing SHIFT LOCK down; if you have SHIFT LOCK down, none of the control keys will work.

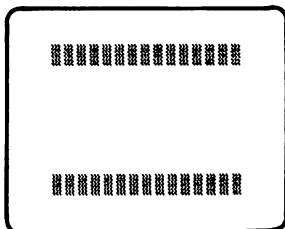
We can also draw a more complex racetrack that requires more maneuvering skill. All we must change is the track-drawing subroutine. The new version is

### **Program 22-3 Intermediate Car Race**

Do not type NEW; retain Program 22-2 in memory.

```
6995 REM DRAW INTERMEDIATE RACE TRACK
6998 REM START DOWN 7 LINES
7000 PRINT "{7 DOWN}";
7005 REM DRAW UPPER WALL
7010 PRINT TAB(8) "████████████████████████████████████████" ← 15 COMMODORE +s. Line 7030 is the
7015 REM DRAW LOWER WALL                                         same.
7020 PRINT "{7 DOWN}";
7030 PRINT TAB(8) "████████████████████████████████████████"
7040 RETURN
RUN
```

The screen will look like



Take a test drive around the track. Try maneuvering between the walls. Since the car has momentum, you must prepare to turn it before it reaches an opening. Put your finger on the key you plan to press, but don't push down too early. If the car goes too far, you do have the luxury of being able to turn it around and go back. Real race-car drivers don't have this option—imagine how many head-on collisions would occur if they did.

We can add sound as follows:

### **Program 22-4 Car Race, With Sound**

Do not type NEW since we want to keep Program 22-3.

```
55 REM CLEAR SOUND CHIP AND SET VOLUME
60 FOR J=54272 TO 54296
70 POKE J,0
80 NEXT J
90 POKE 54296,15
285 GOSUB 8000
325 GOSUB 8000
365 GOSUB 8000
405 GOSUB 8000
```

```

7995 REM PRODUCE SOUND
8000 FOR K=1 TO 5
8010 POKE 54277,16*0+12:REM MINIMUM ATTACK, HIGH DECAY
8020 POKE 54276,33:REM START WAVEFORM 2
8025 REM PLAY G SHARP IN OCTAVE 2
8030 POKE 54273,6
8040 POKE 54272,167
8050 FOR L=1 TO 5:NEXT L
8060 POKE 54276,32:REM STOP WAVEFORM 2
8070 NEXT K
8080 RETURN
RUN

```

A more complex racetrack has several openings. Since the car is six lines tall and six columns wide, all openings must be at least that large for it to pass through. The advanced track looks like

```

#####      #####      #####

#####      #####      #####

```

We can add numbers to designate the car's route.

```

#####  1  #####  6  #####

      2

#####  5  #####  3  #####

      4

```

### Program 22-5 Car Race, With Advanced Racetrack

Do not type NEW; leave Program 22-3 or 22-4 in memory.

```

6995 REM DRAW ADVANCED RACE TRACK
6998 REM START DOWN 6 LINES
7000 PRINT "{6 DOWN}";
7005 REM DRAW TOP BARRIERS
7010 PRINT "##### 1 ##### 6"
7020 PRINT "{3 DOWN}";
7030 PRINT TAB(7)"2";
7040 PRINT "{3 DOWN}";
7045 REM DRAW BOTTOM BARRIERS
7050 PRINT "##### 5 ##### 3"
7060 PRINT "{3 DOWN}";
7070 PRINT TAB(9)"4";
7080 RETURN
RUN

```

9 COMMODORE + s  
 3 spaces  
 7 COMMODORE + s  
 3 spaces  
 6 COMMODORE + s  
 4 spaces

See if you can maneuver the car through the numbers in the proper order without destroying any track, backing up, or hitting the edge of the screen. Tricky, isn't it?

You can easily devise other geometries for the track-drawing subroutine. Give the driver a challenge, but also give him or her a reasonable chance to avoid crashing. One approach is to make the track more difficult (say, by narrowing the openings in Program 22-5) each time the driver completes a circuit without crashing. We could also simplify the track each time the driver fails, thus keeping a beginner from getting discouraged. Many arcade games become more difficult as the player becomes more skillful; this practice holds the player's interest and keeps the quarters coming.

# 23

## ADD A LITTLE COLOR TO YOUR LIFE

So far, we have barely mentioned the fascinating subject of color. Of course, those of you with black and white television sets will have to content yourselves with watching variations in gray scale.

The easiest way to change the Commodore 64's printing color is by pressing the CTRL key and one of the following keys simultaneously:

Key	Color
1	Black
2	White
3	Red
4	Cyan (Greenish-blue)
5	Purple
6	Green
7	Blue
8	Yellow

When you do this, the cursor will appear in the selected color. So will everything you type until you select another color, press STOP and RESTORE together, or turn the computer off. Note that the computer always starts out printing in light blue. We can use this simple approach to vary the color of a moving figure.

### ***Program 23-1 Colorful Alien Moving Right***

```
NEW
100 PRINT CHR$(147);
110 FOR C=1 TO 25
120 GOSUB 1000
125 REM MOVE ALIEN RIGHT
130 PRINT "{4 LEFT} {2 UP}";
```

```

140 FOR K=1 TO 200:NEXT K
150 NEXT C
160 GOTO 9000
995 REM PICTURE OF ALIEN
1000 PRINT " \-/ ";
1020 PRINT " [O] "; ← Put a space on each side of alien.
1040 PRINT " /-\ ";
1010 PRINT "{5 LEFT} {DOWN}";
1030 PRINT "{5 LEFT} {DOWN}";
1050 RETURN
9000 PRINT CHR$(147)
RUN

```

To determine the alien's color, simply press CTRL and a color key before entering RUN. For example, press CTRL and 1 (BLK) to make the alien black, or CTRL and 3 (RED) to make it red. What happens if you press CTRL and 7 (BLU)? Blue printing on a blue screen is like writing in invisible ink. The cursor disappears, and you cannot see what you are typing or what the computer is displaying. This combination is well suited to intelligence agencies, gossip columnists, and paranoids. To return the printing color to light blue, press STOP and RESTORE simultaneously.

RUN Program 23-1 with each of the eight colors. Be sure to adjust the controls on your television set. Note that the computer keeps working in the color you choose until you select another color. You may want to change back to cyan (CTRL and 4) or white (CTRL and 2) before continuing.

We can also place color-change instructions in PRINT statements just like cursor-moving instructions. Here again, we need special notation, since what appears on the screen is meaningless. The following list shows our notation for color change instructions and what you will actually see.

Color produced	Keys pressed	Program notation	Screen symbol
Black	CTRL and 1	{BLK}	■
White	CTRL and 2	{WHT}	□
Red	CTRL and 3	{RED}	■
Cyan	CTRL and 4	{CYN}	■
Purple	CTRL and 5	{PUR}	■
Green	CTRL and 6	{GRN}	■
Blue	CTRL and 7	{BLU}	■
Yellow	CTRL and 8	{YEL}	■

One problem with displaying colors is that they show up so poorly against the computer's normal dark blue background. Fortunately, we can change the background (screen) color to white with

```
POKE 53281,1
```

at the beginning of a color program. We can then return it to blue with

```
POKE 53281,6
```

at the end.

The following program prints the name of the colors, each in its own color. Be sure to press CTRL to enter a color change instruction, not SHIFT or COMMODORE. Watch the screen carefully; having to choose between CTRL, SHIFT, and COMMODORE leads to many typing errors.

### ***Program 23-2 Color Names in Color***

```
NEW
100 PRINT CHR$(147);
105 REM MAKE SCREEN WHITE
110 POKE 53281,1
115 REM PRINT COLOR NAMES
120 PRINT TAB(15)"{2 DOWN}{BLK}BLACK"
130 PRINT TAB(15)"{2 DOWN}{WHT}WHITE"
140 PRINT TAB(15)"{2 DOWN}{RED}RED"
150 PRINT TAB(15)"{2 DOWN}{CYN}CYAN"
160 PRINT TAB(15)"{2 DOWN}{PUR}PURPLE"
170 PRINT TAB(15)"{2 DOWN}{GRN}GREEN"
180 PRINT TAB(15)"{2 DOWN}{BLU}BLUE"
190 PRINT TAB(15)"{2 DOWN}{YEL}YELLOW"
200 GOTO 200
RUN
```

You will see all the color names except WHITE, but there will be a mysterious gap between BLACK and RED. The problem is that white printing is invisible against a white background. If you leave the background its normal dark blue by omitting line 110, the white line will appear but the blue line will vanish.

The program will run forever, since line 200 branches to itself. What happens when you press STOP? The background stays white. Furthermore, READY and the cursor are yellow, since the last color command is {YEL}. To restore the dark blue background, enter

```
POKE 53281,6
```

without a line number.

We can use different colors in different parts of a picture. For example, we could revise Program 23-1 to make the alien's lines red, green, and black from top to bottom. To obtain a white background for easier viewing, enter

```
POKE 53281,1
```

without a line number before RUNning Program 23-3. You may then want to use black (CTRL and 1) or dark blue (CTRL and 7) as the printing color.

### ***Program 23-3 Multicolored Alien Moving Right***

```
NEW
100 PRINT CHR$(147);
110 PRINT "{5 DOWN}"
120 FOR C=1 TO 25
130 GOSUB 1000
135 REM MOVE ALIEN RIGHT
140 PRINT "{4 LEFT} {2 UP}";
```





```

2050 FOR K=1 TO 200:NEXT K
2060 PRINT "{5 UP}";
2070 RETURN
2995 REM PICTURE #3 OF HORSE
3000 PRINT " " ← 8 spaces
3010 PRINT " !---\ "
3020 PRINT " (---)\ "
3030 PRINT " /! /! !" ← Shifted B
3040 PRINT " /! /! ! 0"
3050 FOR K=1 TO 200:NEXT K
3060 PRINT "{5 UP}";
3070 RETURN
9000 PRINT CHR$(147)
RUN

```

By changing line 170, you can make the horse any color you want. You could even make the horse change colors, much as the famous "horse of a different color" did in the motion picture "The Wizard of Oz."

### **Program 23-5 Horse of a Different Color**

Do not type NEW; leave Program 23-4 in memory.

```

192 IF J=3 THEN PRINT "{RED}";
194 IF J=5 THEN PRINT "{PUR}";
196 IF J=7 THEN PRINT "{CYN}";
198 IF J=9 THEN PRINT "{YEL}";
RUN

```

We can also make an alien change color as it moves. The following program prints three pictures in the original color (normally light blue), four in red, four in green, and four in yellow.

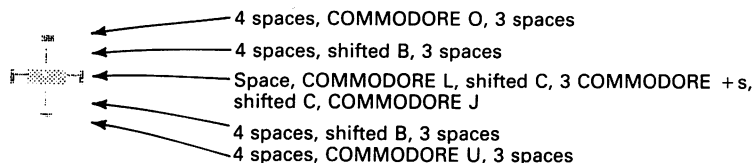
### **Program 23-6 Moving Alien With Varying Colors**

```

NEW
100 PRINT CHR$(147);
110 PRINT "{4 DOWN}";
120 FOR C=1 TO 25
130 IF C=4 THEN PRINT "{RED}";
140 IF C=8 THEN PRINT "{GRN}";
150 IF C=12 THEN PRINT "{YEL}";
160 GOSUB 1000
165 REM MOVE ALIEN RIGHT
170 PRINT "{4 LEFT} {2 UP}";
180 FOR K=1 TO 200:NEXT K
190 NEXT C
200 GOTO 9000
995 REM PICTURE OF ALIEN
1000 PRINT " \-/ ";
1020 PRINT " (0) "; ← Put a space on each side.
1040 PRINT " /-\ ";
1010 PRINT "{5 LEFT} {DOWN}";
1030 PRINT "{5 LEFT} {DOWN}";
1050 RETURN
9000 PRINT CHR$(147)
RUN

```

We can easily draw a space station that changes color as it moves. The station looks like



In the following program, the station takes on seven different colors; it might be passing through a huge gas cloud or reflecting light from many different stars.

### ***Program 23-7 Space Station Changing Color***

```

NEW
100 PRINT CHR$(147)
105 REM MOVE NEAR CENTER OF SCREEN
110 FOR J=1 TO 9
120 PRINT
130 NEXT J
135 REM DRAW MOVING SPACE STATION
140 FOR J=1 TO 4
145 REM FIRST BLACK
150 PRINT "{BLK}";
160 GOSUB 1000
165 REM SECOND RED
170 PRINT "{RED}";
180 GOSUB 1000
185 REM THIRD CYAN
190 PRINT "{CYN}";
200 GOSUB 1000
205 REM FOURTH PURPLE
210 PRINT "{PUR}";
220 GOSUB 1000
225 REM FIFTH GREEN
230 PRINT "{GRN}";
240 GOSUB 1000
245 REM SIXTH BLUE
250 PRINT "{BLU}";
260 GOSUB 1000
265 REM SEVENTH YELLOW
270 PRINT "{YEL}";
280 GOSUB 1000
290 NEXT J
300 GOTO 9000
995 REM MOVE SPACE STATION RIGHT
998 REM DRAW SPACE STATION
1000 PRINT "      █      ";
1020 PRINT "      |      ";
1040 PRINT "  ┌───┴───┐ ";
1060 PRINT "      |      ";
1080 PRINT "      ─      ";
1010 PRINT "{8 LEFT}{DOWN}";
1030 PRINT "{8 LEFT}{DOWN}";
1050 PRINT "{8 LEFT}{DOWN}";
1070 PRINT "{8 LEFT}{DOWN}";
1085 REM WAIT BETWEEN PICTURES
1090 FOR K=1 TO 100:NEXT K
1095 REM MOVE STATION RIGHT
1100 PRINT "{7 LEFT}{4 UP}";
1110 RETURN
9000 PRINT CHR$(147)
RUN

```

We can also make a colorful dot move in a circle. The dot changes from red to green to yellow to cyan as it moves.

**Program 23-8 Multicolored Dot Moving in a Circle**

```

NEW
100 PRINT CHR$(147);
105 REM MOVE NEAR CENTER OF SCREEN
110 FOR J=1 TO 10
120 PRINT
130 NEXT J
140 PRINT TAB(17);
145 REM ESTABLISH TIME CONSTANT
150 T=100
155 REM CIRCULAR MOTION
160 FOR J=1 TO 20
170 PRINT "{RED}";
175 REM MOVE RIGHT AND DOWN
180 PRINT "{2 DOWN} {RIGHT}";
190 GOSUB 1000
200 PRINT "{GRN}";
205 REM MOVE RIGHT AND UP
210 PRINT "{2 UP} {RIGHT}";
220 GOSUB 1000
230 PRINT "{YEL}";
235 REM MOVE LEFT AND UP
240 PRINT "{2 UP} {3 LEFT}";
250 GOSUB 1000
260 PRINT "{CYN}";
265 REM MOVE LEFT AND DOWN
270 PRINT "{2 DOWN} {3 LEFT}";
280 GOSUB 1000
290 NEXT J
300 GOTO 9000
995 REM DRAW DOT, WAIT, ERASE IT
1000 PRINT "■";
1010 FOR K=1 TO T:NEXT K
1020 PRINT "{LEFT}";
1030 PRINT " ";
1040 RETURN
9000 PRINT CHR$(147)
RUN

```

Annotations for Program 23-8:

- Line 100: Put semicolon here.
- Line 140: Put semicolon here.
- Line 170: Red dot
- Line 200: Green dot
- Line 230: Yellow dot
- Line 260: Cyan dot
- Line 1000: Shifted Q
- Line 1030: 1 space

The value of T in line 150 controls the dot's speed. Setting T to 250 slows the dot down enough so you can see it hop from one position to the next. We can produce many unusual effects with slight variations on this program. Try the following:

1. Change line 1030 to

```

PRINT "□";

```

Shifted W

This gives the effect of twinkling lights as the dot appears to move from one empty circle to the next.

2. Change line 1030 to

```

PRINT "♥";

```

Shifted S

Now you see four beating hearts. The value of T determines the pulse rate.

3. Change line 210 to

```
PRINT "{2 DOWN} {RIGHT}";
```

Now the cursor never returns to its original position, so the pattern of dots keeps expanding. You can control the direction of the growth by changing the cursor-moving commands. Experiment with lines 180, 210, 240, and 270 to see what effects you can create.

We can also extend our random art program (Program 9-3) to include color. The following program uses four colors, (red, green, yellow, and cyan) and prints 10 random characters before changing colors.

### **Program 23-9 Random Art With Color**

```
NEW
100 PRINT CHR$(147);
105 REM PRODUCE 10 CHARACTERS IN RED
110 PRINT "{RED}";
120 GOSUB 1000
125 REM PRODUCE 10 CHARACTERS IN GREEN
130 PRINT "{GRN}";
140 GOSUB 1000
145 REM PRODUCE 10 CHARACTERS IN YELLOW
150 PRINT "{YEL}";
160 GOSUB 1000
165 REM PRODUCE 10 CHARACTERS IN CYAN
170 PRINT "{CYN}";
180 GOSUB 1000
190 GOTO 110
995 REM PRODUCE 10 RANDOM CHARACTERS
1000 FOR J=1 TO 10
1010 PRINT CHR$(19);
1015 REM MOVE DOWN TO A RANDOM ROW
1020 NR=INT(RND(1)*23)+1 ←————— The computer makes up a number
1030 FOR R=1 TO NR                                     between 1 and 23.
1040 PRINT
1050 NEXT R
1055 REM MOVE RIGHT TO A RANDOM COLUMN
1060 C=INT(RND(1)*39) ←————— C is between 0 and 38.
1065 REM GENERATE A RANDOM SYMBOL
1070 S=INT(RND(1)*64)+64 ←————— S is between 64 and 127.
1080 PRINT TAB(C)CHR$(S); ←————— Puts the random symbol in a random
1090 NEXT J                                             column.
1100 RETURN
RUN
```

The program will run forever, producing 10 random characters in each color. To stop the action, first press RUN/STOP, then SHIFT and CLR/HOME together to clear the screen, and finally CTRL and 4 to produce a cyan printing color. You can vary the character set by using Table 9-1.

Although the Commodore 64's keyboard has markings for only eight colors, the computer can actually produce 16. To obtain the other eight, press COMMODORE (instead of CTRL) and a color key. This results in the following colors:

Color produced	Keys pressed	Program notation	Screen symbol
Orange	COMMODORE and 1	{ORN}	☉
Brown	COMMODORE and 2	{BRN}	☼
Light red	COMMODORE and 3	{LRD}	☽
Gray 1 (dark gray)	COMMODORE and 4	{GY1}	☾
Gray 2 (medium gray)	COMMODORE and 5	{GY2}	☿
Light green	COMMODORE and 6	{LGR}	☺
Light blue	COMMODORE and 7	{LBL}	☻
Gray 3 (light gray)	COMMODORE and 8	{GY3}	☼

The following program prints the names of the additional colors, each in its own color. Be sure to use COMMODORE here to enter the color-change instruction, not CTRL or SHIFT.

### ***Program 23-10 Additional Color Names in Color***

```

NEW
100 PRINT CHR$(147);
105 REM MAKE SCREEN WHITE
110 POKE 53281,1
115 REM PRINT COLOR NAMES
120 PRINT TAB(15)"{2 DOWN}{ORG}ORANGE"
130 PRINT TAB(15)"{2 DOWN}{BRN}BROWN"
140 PRINT TAB(15)"{2 DOWN}{LRD}LIGHT RED"
150 PRINT TAB(15)"{2 DOWN}{GY1}GRAY 1 (DARK)"
160 PRINT TAB(15)"{2 DOWN}{GY2}GRAY 2 (MEDIUM)"
170 PRINT TAB(15)"{2 DOWN}{LGR}LIGHT GREEN"
180 PRINT TAB(15)"{2 DOWN}{LBL}LIGHT BLUE"
190 PRINT TAB(15)"{2 DOWN}{GY3}GRAY 3 (LIGHT)"
200 GOTO 200
RUN

```

Note that light blue is the computer's start-up printing color. You can thus restore that color at any time by pressing COMMODORE and 7 simultaneously.

While we have used only dark blue and white backgrounds, we can actually change the screen (background) and outside color to any of the 16 colors. Table 23-1 lists the values needed to change the border color with

```
POKE 53280,BC
```

and the screen with

```
POKE 53281,SC
```

For example, the command that makes the border green is

```
POKE 53280,5
```

and the command that makes the background light red is

```
POKE 53281,10
```

These POKES do not affect the printing color. Of course, you will not be able to distinguish the border if you make it the same color as the screen.

**TABLE 23-1 SCREEN AND BORDER COLOR VALUES**

0	Black	4	Purple	8	Orange	12	Gray 2
1	White	5	Green	9	Brown	13	Light Green
2	Red	6	Blue	10	Light Red	14	Light Blue
3	Cyan	7	Yellow	11	Gray 1	15	Gray 3

Be careful when using colors. Characters will be difficult to read if the printing color and the screen color are similar, and invisible if they are identical.

The following program produces a variety of screen and border colors.

### ***Program 23-11 Kaleidoscope of Screen and Border Colors***

```
NEW
100 PRINT CHR$(147);
110 FOR CB=0 TO 15
120 FOR CS=0 TO 15
125 REM CHANGE BORDER COLOR
130 POKE 53280,CB
135 REM CHANGE THE SCREEN COLOR
140 POKE 53281,CS
150 FOR K=1 TO 100:NEXT K
160 NEXT CS
170 NEXT CB
RUN
```

Program 23-11 starts with black as the screen color and displays all 16 possible borders. It then continues with the next screen color. After the program ends, enter

```
POKE 53281,6
```

to restore the normal blue screen and

```
POKE 53280,14
```

to restore the normal light blue border.

We can also produce a red, yellow, and blue neon sign with moving lights. We will put the sign in the center of a solid white screen (white border and white background).

## Program 23-12 Neon Sign With Moving Lights

```

NEW
100 PRINT CHR$(147);
105 REM MAKE BORDER WHITE
110 POKE 53280,1
115 REM MAKE BACKGROUND WHITE
120 POKE 53281,1
125 REM MOVE TO CENTER OF SCREEN
130 FOR J=1 TO 10
140 PRINT
150 NEXT J
155 REM PRINT BLUE MESSAGE IN MIDDLE
160 PRINT "{2 DOWN}{BLU}";
170 PRINT TAB(11)"GRAND OPENING SALE"
180 PRINT "{3 UP}";
185 REM CREATE MOVING YELLOW LIGHTS
190 PRINT "{YEL}";
200 FOR J=1 TO 50
205 REM DRAW LIGHTS ONE WAY
210 PRINT TAB(10)"●○○○○○○○○○○○○○○○○○○" ← 10 pairs of shifted Q,
                                                    shifted W
220 PRINT TAB(10)"○";TAB(29)"●" ← Shifted W, shifted Q
230 PRINT TAB(10)"●";TAB(29)"○" ← Shifted Q, shifted W
240 PRINT TAB(10)"○";TAB(29)"●" ← Shifted W, shifted Q
250 PRINT TAB(10)"●○○○○○○○○○○○○○○○○○○" ← 10 pairs of shifted Q,
                                                    shifted W. Same as line 210.

260 FOR K=1 TO 50:NEXT K
270 PRINT "{5 UP}";
275 REM DRAW LIGHTS OTHER WAY
280 PRINT TAB(10)"○○○○○○○○○○○○○○○○○○" ← 10 pairs of shifted W,
                                                    shifted Q
290 PRINT TAB(10)"●";TAB(29)"○" ← Shifted Q, shifted W
300 PRINT TAB(10)"○";TAB(29)"●" ← Shifted W, shifted Q
310 PRINT TAB(10)"●";TAB(29)"○" ← Shifted Q, shifted W
320 PRINT TAB(10)"○○○○○○○○○○○○○○○○○○" ← 10 pairs of shifted W,
                                                    shifted Q. Same as line 280.
330 FOR K=1 TO 50:NEXT K
340 PRINT "{5 UP}";
350 NEXT J
RUN

```

To make GRAND OPENING SALE flash periodically, add

```

321 REM BLANK MESSAGE FOR A WHILE
322 PRINT "{3 UP}";
323 PRINT TAB(11)" " ← 18 spaces
324 PRINT "{UP}";
325 FOR K=1 TO 25:NEXT K
326 REM THEN RESTORE MESSAGE IN BLUE
327 PRINT "{BLU}";TAB(11)"GRAND OPENING SALE"
328 PRINT "{YEL}{2 DOWN}";
330 FOR K=1 TO 25:NEXT K

```

You can adjust the wait loops on lines 260, 325, and 330 to change the rates at which the lights move and the sign flashes.

Feel free to change the sign's colors or message. Note the use of two different patterns of characters to create the illusion of moving lights.

We can even use a combination with the same printing and screen colors to hide a message. Here is the latest way to give James Bond his orders. All the agent must do is press the British pound sign (£) key.



### **Program 23-13 James Bond's Secret Orders**

```
NEW
100 PRINT CHR$(147);
105 REM MAKE BORDER WHITE
110 POKE 53280,1
115 REM MAKE BACKGROUND WHITE
120 POKE 53281,1
125 REM MAKE PRINTING COLOR WHITE
130 PRINT "{WHT}";
140 PRINT "{2 DOWN}";
150 PRINT "TO: JAMES BOND"
160 PRINT
170 PRINT "FROM: M."
180 PRINT
190 PRINT "TAKE THE NEXT FLIGHT TO TIMBUKTU"
200 PRINT
210 PRINT "CARRY NO LUGGAGE EXCEPT A RED SCARF"
220 PRINT
230 PRINT "YOU WILL BE MET AT THE AIRPORT"
240 PRINT
250 PRINT "THIS MISSION IS EXTREMELY DANGEROUS"
260 PRINT
270 PRINT "BE SURE YOUR WILL IS UP TO DATE"
280 PRINT
290 PRINT "DON'T FORGET YOUR AMERICAN EXPRESS CARD"
300 PRINT
310 PRINT "READ FAST. THIS MESSAGE SELF-DESTRUCTS"
315 REM WAIT FOR BOND TO PRESS BRITISH POUND SIGN KEY
320 GET K$
330 IF K$("<>") = "£" THEN 320
335 REM TURN BACKGROUND DARK BLUE SO MESSAGE APPEARS
340 POKE 53281,6
345 REM LET MESSAGE SHOW FOR A WHILE
350 FOR K=1 TO 4000:NEXT K
355 REM THEN BLANK SCREEN
360 PRINT CHR$(147)
365 REM AND DESTROY THE ENTIRE PROGRAM
370 NEW
RUN
```

Be sure to save this program before typing RUN. The message appears in white on a dark blue background, then the screen goes blank. Note the use of NEW in line 370 to make the program self-destruct.

# 24

## REVERSING THE ODDS

Besides changing the Commodore 64's color scheme, we can also reverse its character and screen colors. This makes the printing look like an X ray or a photographic negative, since it appears in the screen color against a background of the normal character color. If, for example, the computer is printing its usual light blue characters on a blue background, reversal results in blue characters on a light blue background.

To reverse colors, all you must do is press CTRL and the RVS ON (9) key simultaneously; we will refer to this as {RVS} in programs. Be careful when you do this. Nothing happens on the screen, so you only know whether reverse is on when you type. To return to normal, press CTRL and RVS OFF (0) simultaneously; we will refer to this in programs as {OFF}. Reverse the Commodore 64's standard colors and type THIS IS REVERSE PRINTING to see how reversed characters look. Note that pressing RETURN turns the reverse off.

We can put reversal instructions in programs just like color changes and cursor moves. All we must do is type {RVS} or {OFF} inside quotation marks in PRINT statements. When you type {RVS} (CTRL and 9) inside quotation marks, a reversed R appears on the screen. {OFF} shows up as a reversed shifted R.

The following program prints your name in reverse near the center of the screen. Note how distinctive it looks, as compared to ordinary printing.

### ***Program 24-1 Your Name in Reverse***

```
NEW
100 PRINT CHR$(147);
110 FOR J=1 TO 10
120 PRINT
130 NEXT J
140 PRINT TAB(18) "{RVS}TONY{OFF}"
RUN
```

Press the CTRL and RVS ON keys simultaneously.

Put your name here.

Press the CTRL and RVS OFF keys simultaneously. You can omit this since RETURN turns the reverse off anyway.

Adjust the TAB in line 140 to center your name. Note that the {RVS} and {OFF} characters appear in the listing, making the printing in line 140 look two columns longer than it actually is. Like cursor moves and color changes, {RVS} and {OFF} do not appear in the printed output.

We can also make the computer print a staircase of reversed names.

### ***Program 24-2 Your Name in Reverse, Moving Left***

```

NEW
100 PRINT CHR$(147)
110 INPUT "WHAT IS YOUR NAME";N$
120 PRINT CHR$(147);
130 FOR C=1 TO 15
140 PRINT TAB(C)"{RVS}"N$
150 FOR K=1 TO 100:NEXT K
160 NEXT C
RUN

```

Remember to press CTRL and RVS ON (9) to enter (RVS).

Reversed printing lets us flash words on the screen. We simply have the computer alternate reversed and normal printing of the same characters. The next program flashes your name in the middle of the screen. You can adjust the TAB as in Program 24-1.

### ***Program 24-3 Your Name Flashing***

```

NEW
100 PRINT CHR$(147);
110 INPUT "WHAT IS YOUR NAME";N$
120 PRINT CHR$(147);
125 REM MOVE DOWN TO CENTER OF SCREEN
130 FOR J=1 TO 10
140 PRINT
150 NEXT J
155 REM FLASH NAME
160 FOR J=1 TO 20
165 REM PRINT NAME REVERSED
170 PRINT TAB(15)"{RVS}"N$
180 FOR K=1 TO 200:NEXT K
190 PRINT "{UP}";
195 REM PRINT NAME NORMAL
200 PRINT TAB(15)N$
210 FOR K=1 TO 200:NEXT K
220 PRINT "{UP}";
230 NEXT J
RUN

```

Flashing can direct people's attention to important messages such as ENEMY APPROACHING, REACTOR CRITICAL, or ABANDON SHIP. The following program emphasizes its message with flashing.

## Program 24-4 Abandon Ship

```
NEW
100 PRINT CHR$(147);
105 REM MOVE DOWN TO CENTER OF SCREEN
110 FOR J=1 TO 10
120 PRINT
130 NEXT J
135 REM FLASH MESSAGE
140 FOR J=1 TO 20
145 REM PRINT MESSAGE REVERSED
150 PRINT TAB(13)"(RVS)ABANDON SHIP"
160 FOR K=1 TO 200:NEXT K
170 PRINT "{UP}";
175 REM PRINT MESSAGE NORMAL
180 PRINT TAB(13)"ABANDON SHIP"
190 FOR K=1 TO 200:NEXT K
200 PRINT "{UP}";
210 NEXT J
RUN
```

The reversal command makes the printing appear longer on line 150 than on line 180, although they are actually the same length.

We can easily modify the program to let the user select the message. The only problem is centering a message of unknown length. To do this, we use the computer's built-in LEN (length) function. LEN finds the length of a string, including embedded (interior) spaces but not ones at either end. Once we use LEN to determine the message's length, L, we know how many spaces to indent. Since the entire screen is 40 characters wide, we should indent

```
INT(20-L/2)
```

spaces to center the message. INT drops the fraction if the length is odd.

## Program 24-5 Message Flasher

```
NEW
100 PRINT CHR$(147);
105 REM LET USER ENTER MESSAGE
110 PRINT "ENTER MESSAGE"
120 PRINT "39 CHARACTERS MAXIMUM INCLUDING SPACES."
130 PRINT "NO COMMAS OR COLONS, PLEASE!"
140 INPUT M$
145 REM CHECK IF LENGTH IS PROPER
150 L=LEN(M$)
155 REM REJECT STRING IF TOO LONG OR NOTHING IN IT
160 IF L=0 OR L>40 THEN 100 ← OR means "either one or both."
170 PRINT CHR$(147);
175 REM MOVE DOWN TO CENTER LINE
180 FOR J=1 TO 10
190 PRINT
200 NEXT J
205 REM DETERMINE HOW FAR TO INDENT FOR CENTERING
210 INDENT=INT(20-L/2)
215 REM FLASH MESSAGE
220 FOR J=1 TO 20
225 REM PRINT MESSAGE REVERSED
230 PRINT TAB(INDENT)"(RVS)"M$
240 FOR K=1 TO 200:NEXT K
```

```

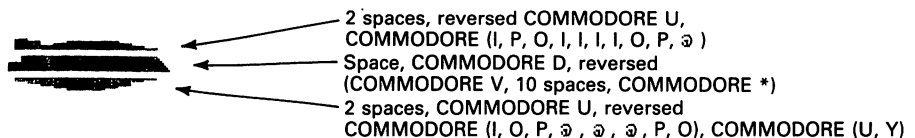
250 PRINT "{UP}";
255 REM PRINT MESSAGE NORMAL
260 PRINT TAB(INDENT)M$
270 FOR K=1 TO 200:NEXT K
280 PRINT "{UP}";
290 NEXT J
300 PRINT CHR$(147)
RUN

```

Note the restrictions on the messages. Valid entries include I WANT TO BE ALONE, DO NOT DISTURB, and LOOK OUT BELOW. What happens if you start or end your message with a space?

The program rejects messages that are too long for the Commodore 64's screen. What happens if you enter STOP THE WORLD, I WANT TO GET OFF or HELP, I'M TRAPPED IN THE COMPUTER? The computer thinks the first comma marks the end of the entry. You can, however, enter these messages if you put quotation marks around them. This approach also allows you to enter messages with colons (e.g., "DESTINATION: MARS") or commands (e.g., "{RED} RED ALERT{CYN}" or "{GRN}PRESERVE OUR FORESTS{CYN}")

We can also draw a blimp and have it flash a message as it moves. The blimp looks like



We used reversed graphics characters to form the blimp's fin, nose, and bottom, and reversed spaces (solid squares) to form the bulk of its body. Note that you must enter {RVS} by pressing CTRL and 9 before typing the characters you want reversed. You must enter {OFF} by pressing CTRL and 0 if subsequent characters on the same line are to appear in the normal form. Thus we need {OFF} in lines 1 and 3 of the blimp, but not in line 2 since it ends with reversed characters. Remember that RETURN always turns the reverse off automatically.

For example, to enter the top line of the blimp, proceed as follows after typing the opening quotation mark:

1. Press the space bar twice.
2. Enter {RVS} by pressing CTRL and RVS ON (9).
3. Enter COMMODORE U.
4. Enter {OFF} by pressing CTRL and RVS OFF (0).
5. Complete the line by holding COMMODORE down and typing I, P, O, I, I, I, O, P, and @. Then type an ending quotation mark and RETURN.

While reversed graphics characters are quite useful in drawing pictures, they also introduce new problems:

1. Pictures look different when you RUN the programs. This is because the reversed characters do not appear in the listing. Determining how a picture involving both

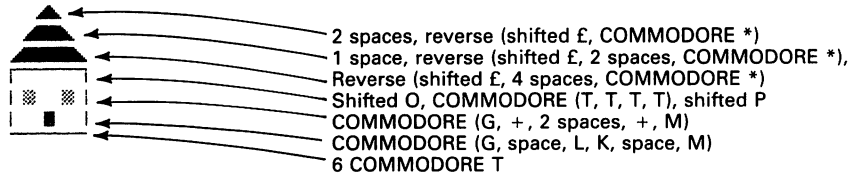




To add a fourth tank to the squadron, insert

```
155 GOSUB 1000
```

We can also use reversed characters to draw a house with a solid roof. We form the diagonal edges from reversed triangles (shifted British pound sign and Commodore \*) and the main part of the roof from reversed spaces. The house looks like



### Program 24-9 Little House on the Prairie

```
NEW
100 PRINT CHR$(147);
105 REM MOVE DOWN TO CENTER OF SCREEN
110 FOR J=1 TO 10
120 PRINT
130 NEXT J
135 REM DRAW SOLID ROOF
140 PRINT " {RVS}▲"
150 PRINT " {RVS}▲"
160 PRINT " {RVS}▲"
165 REM DRAW HOUSE
170 PRINT "| |"
180 PRINT "| | | |"
190 PRINT "| | ■ |"
200 PRINT "| |"
RUN
```

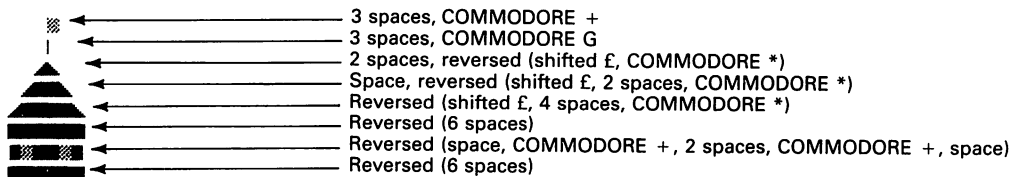
Diagram illustrating the construction of the roof in the program. The roof is drawn using reversed characters. The components are defined by the following Commodore commands:

- 2 spaces
- 4 spaces

Note that the {RVS} characters in lines 140, 150, and 160 make the roof appear out of alignment with the house below it. To make the roof brown and the background yellow, insert

```
106 REM MAKE PRINTING COLOR BROWN
107 PRINT "{BRN}"; ← Press COMMODORE and 2.
108 REM MAKE BACKGROUND YELLOW
109 POKE 53281,7
```

We can produce a more elaborate picture by changing the colors. Here's a little red schoolhouse with a green roof and a flag.





## Program 24-10 Little Red Schoolhouse

```

NEW
100 PRINT CHR$(147);
110 PRINT "{5 DOWN}";
115 REM DRAW PURPLE FLAG
120 PRINT "{PUR}";
130 PRINT "  █"
140 PRINT "  |"
145 REM DRAW GREEN ROOF
150 PRINT "{GRN}";
160 PRINT "  ▲{RVS}▼"
170 PRINT "  {RVS}▼  ▲"
180 PRINT "  {RVS}▼  ▲"
185 REM DRAW RED SCHOOLHOUSE
190 PRINT "{RED}";
200 PRINT "{RVS}";
210 PRINT "  {RVS}  █"
220 PRINT "  {RVS}  |"
RUN

```

Enter

```
POKE 53281,1
```

to make the background white before you run this picture.

We could combine the color changes (lines 120, 150, and 190) with the subsequent PRINT statements, but separating them makes it easier to see which lines they control. Add

```
80 POKE 53280,1
90 POKE 53281,1
```

to produce a scene on a totally white screen.

A flashing character (such as the computer's own cursor) provides a conspicuous indicator of where you are working on the screen. The following program constructs a maze from your keyboard commands—U for up, D for down, R for right, L for left, and E for exit. The flashing diamond always indicates the active position, even if you backtrack.

## Program 24-11 Maze Builder

```

NEW
100 PRINT CHR$(147);
105 REM START CURSOR IN TOP LEFT-HAND CORNER
110 R=1
120 C=1
125 REM FLASH CURSOR
128 REM PRINT NORMAL CURSOR
130 PRINT "  █";
140 FOR K=1 TO 50:NEXT K
150 PRINT "{LEFT}";
155 REM PRINT CURSOR REVERSED
160 PRINT "  {RVS}◊{OFF}";
170 FOR K=1 TO 50:NEXT K
180 PRINT "{LEFT}";
185 REM CHECK FOR KEYBOARD COMMAND
190 GET K$
195 REM E KEY MEANS EXIT

```

```

200 IF K$="E" THEN 9000
205 REM U KEY MEANS UP (IF POSSIBLE)
210 IF K$="U" AND R>1 THEN R=R-1:PRINT "{UP}";
215 REM D KEY MEANS DOWN (IF POSSIBLE)
220 IF K$="D" AND R<24 THEN R=R+1:PRINT "{DOWN}";
225 REM R KEY MEANS RIGHT (IF POSSIBLE)
230 IF K$="R" AND C<40 THEN C=C+1:PRINT "{RIGHT}";
235 REM L KEY MEANS LEFT (IF POSSIBLE)
240 IF K$="L" AND C>1 THEN C=C-1:PRINT "{LEFT}";
250 GOTO 130
9000 PRINT CHR$(147)
RUN

```

In lines 210, 220, 230, and 240, AND between the two conditions makes the combination true only if both are true. Thus line 210 causes the computer to reduce R by 1 and to move the cursor up if the key entry is U *and* R is greater than 1.

Draw a few patterns so you can see the cursor move. If you try to move it off the screen or into the bottom row, it will simply hold still and flash.

The reversed characters can produce more realistic ocean waves than the regular characters we used in Chapter 20. Reversing the triangles (shifted British pound sign and Commodore \*) makes the waves roll on top of the surface instead of below it.

### Program 24-12 Rolling Ocean Waves

```

NEW
75 REM TURN SCREEN AND BORDER WHITE
80 POKE 53280,1
90 POKE 53281,1
100 PRINT CHR$(147);
105 REM MOVE DOWN NEAR CENTER OF SCREEN
110 FOR J=1 TO 10
120 PRINT
130 NEXT J
135 REM MAKE OCEAN WAVES ROLL
140 FOR J=1 TO 20
145 REM DRAW WAVES IN ONE DIRECTION
150 PRINT "{RV$}";
160 FOR K=1 TO 200:NEXT K
165 REM DRAW WAVES IN ANOTHER DIRECTION
170 PRINT "{UP}";
180 PRINT "{RV$}";
190 FOR K=1 TO 200:NEXT K
200 PRINT "{UP}";
210 NEXT J
220 PRINT CHR$(147)
225 REM RETURN SCREEN TO ITS NORMAL COLORS
230 POKE 53280,14
240 POKE 53281,6
250 PRINT "{LBL}";
RUN

```

18 pairs of COMMODORE \* and shifted £s. You should see 18 upside-down light blue pyramids.

18 pairs of shifted £ and COMMODORE \*s. You should see 18 dark blue pyramids.

Press COMMODORE and 7.

Enter the wave lines (150 and 180) carefully. Remember that you must move your fingers between Commodore and SHIFT as well as between the British pound sign and \* each time. Count the different screen colors to determine how many pairs of characters to enter. Experiment with the different screen colors to produce different color combinations.

The waves roll left continuously. Let us now use inverted spaces to draw some solid ocean underneath them. We will make the water red to represent the Red Sea.

## Program 24-13 Rolling Waves on the Red Sea

```

NEW
75 REM TURN SCREEN AND BORDER WHITE
80 POKE 53280,1
90 POKE 53281,1
100 PRINT CHR$(147);
105 REM MOVE DOWN NEAR CENTER OF SCREEN
110 FOR J=1 TO 10
120 PRINT
130 NEXT J
135 REM PAINT SEA RED
140 PRINT "{RED}";
145 REM CREATE A SOLID EXPANSE OF OCEAN
147 REM SKIP WAVE LINE
150 PRINT
155 REM PRINT THREE LINES OF SOLID OCEAN
160 FOR J=1 TO 3
170 PRINT "{RVS}
180 NEXT J
185 REM MOVE CURSOR BACK TO OCEAN SURFACE
190 PRINT "{4 UP}";
195 REM MAKE OCEAN WAVES ROLL
200 FOR J=1 TO 20
205 REM DRAW WAVES IN ONE DIRECTION
210 PRINT "{RVS}
220 FOR K=1 TO 200:NEXT K
225 REM DRAW WAVES IN ANOTHER DIRECTION
230 PRINT "{UP}";
240 PRINT "{RVS}
250 FOR K=1 TO 200:NEXT K
260 PRINT "{UP}";
270 NEXT J
280 PRINT CHR$(147)
285 REM RETURN WITH CYAN PRINTING ON BLUE SCREEN
290 PRINT "{CYN}";
300 POKE 53280,14
310 POKE 53281,6
RUN

```

36 spaces

18 pairs of COMMODORE \*s and shifted £s. You should see 18 upside-down blue pyramids.

18 pairs of shifted £s and COMMODORE \*s. You should see 18 pairs of white pyramids.

What's next? Did you guess? After all, as long as we have drawn the Red Sea, we might as well show you how to part it.

## Program 24-14 Parting the Red Sea

```

35 REM CLEAR SOUND CHIP AND SET VOLUME
40 FOR J=54272 TO 54296
50 POKE J,0
60 NEXT J
70 POKE 54296,15

202 REM NORMAL SEA EXCEPT DURING BREAK TIME (J=6 TO 9)
205 IF J>5 AND J<10 THEN 270
262 REM DONE WITH NORMAL WAVE ACTION
265 GOTO 450
268 REM PRINT ROLLING WAVES WITH PART
270 PRINT "{RVS}
275 REM CONTINUE UNLESS SEA MUST BE PARTED
280 IF J>6 THEN 340
285 REM PART ENTIRE SEA
290 FOR L=1 TO 3
300 PRINT TAB(15)" "
310 NEXT L

```

8 pairs of reversed COMMODORE \*s and shifted £s, 2 spaces, then 9 more reversed pairs. You should see a total of 17 upside-down light blue pyramids.

2 spaces

```

315 REM RETURN TO OCEAN SURFACE
320 PRINT "{3 UP}";
325 REM ANNOUNCE PARTING WITH NOISE
330 GOSUB 7000
335 REM ROLL WAVES SLOWER WHILE PARTING IN PROGRESS
340 FOR K=1 TO 400:NEXT K
350 PRINT "{UP}";
355 REM DRAW PARTED OCEAN WAVES IN ANOTHER POSITION
360 PRINT "{RVS}{(OFF) (RVS)}"
370 FOR K=1 TO 400:NEXT K
375 REM CONTINUE UNLESS SEA MUST BE RECONNECTED
380 IF J<9 THEN 440
385 REM REFILL THE ENTIRE SEA
390 FOR L=1 TO 3
400 PRINT TAB(15)"{RVS} "
410 NEXT L
415 REM RETURN TO OCEAN SURFACE
420 PRINT "{3 UP}";
425 REM ANNOUNCE RECONNECTION WITH NOISE
430 GOSUB 7000
440 PRINT "{UP}";
450 NEXT J
460 GOTO 9000

6995 REM BOOMING NOISE TO ANNOUNCE PARTING, RECONNECTION OF SEA
7000 POKE 54277,16*4+12:REM LOW ATTACK, HIGH DECAY
7010 POKE 54276,129:REM START NOISE WAVEFORM
7020 POKE 54273,34
7030 POKE 54272,75
7040 FOR L=1 TO 1000:NEXT L
7050 POKE 54276,128:REM STOP NOISE WAVEFORM
7060 RETURN
8995 REM RETURN SCREEN TO DARK BLUE WITH CYAN PRINTING
9000 PRINT CHR$(147)
9010 POKE 53280,14
9020 POKE 53281,6
9030 PRINT "{CYN}"
RUN

```

8 pairs of reversed shifted £s and COMMODORE \*s, 2 spaces, then 9 more reversed pairs. You should see a total of 17 dark blue pyramids.

" 2 spaces

Clearly, parting the Red Sea is a complicated business. You must create a gap in the waves and in the underlying ocean. After the Israelites have passed through, you must then fill in the ocean and the waves. The moral here is that programmers should not attempt to play God.

# Appendix A

## Music Note Values\*

Musical Note	High Frequency	Low Frequency	
<i>Octave 2</i>			
NATURAL			
C	4	48	
D	4	180	
E	5	71	
F	5	152	
G	6	71	
A	7	12	
B	7	233	
SHARP			
C#	4	112	
D#	4	251	
F#	5	237	There is no E sharp
G#	6	167	
A#	7	119	There is no B sharp
FLAT			
C -	3	244	Same as B natural in octave 1
D -	4	112	
E -	4	251	
F -	5	71	Same as E natural
G -	5	237	
A -	6	167	
B -	7	119	

*Octave 3*

NATURAL		
C	8	97
D	9	104
E	10	143
F	11	48
G	12	143
A	14	24
B	15	210
SHARP		
C #	8	225
D #	9	247
F #	11	218
G #	13	78
A #	14	239
FLAT		
C -	7	233
D -	8	225
E -	9	247
F -	10	143
G -	11	218
A -	13	78
B -	14	239

There is no E sharp

There is no B sharp

Same as B natural in octave 2

Same as E natural

*Octave 4*

NATURAL		
C	16	195
D	18	209
E	21	31
F	22	96
G	25	30
A	28	49
B	31	165
SHARP		
C #	17	195
D #	19	239
F #	23	181
G #	26	156
A #	29	233
FLAT		
C -	15	210
D -	17	195
E -	19	239
F -	21	31
G -	23	181
A -	26	156
B -	29	233

There is no E sharp

There is no B sharp

Same as C natural in octave 3

Same as E natural

Octave 5

NATURAL			
C	33	135	
D	37	162	
E	42	62	
F	44	193	
G	50	60	
A	56	99	
B	63	75	
SHARP			
C#	35	134	
D#	39	223	There is no E sharp
F#	47	107	
G#	53	57	
A#	59	190	There is no B sharp
FLAT			
C-	31	165	Same as B natural in octave 4
D-	35	134	
E-	39	223	
F-	42	62	Same as E natural
G-	47	107	
A-	53	57	
B-	59	190	

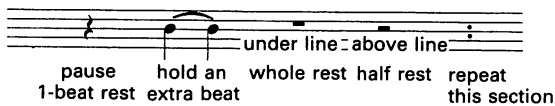
---

\*Note that to complete an octave (eight notes) you must end it with the first note of the next octave.

# APPENDIX B

## MUSIC REVIEW

The following pages contain a brief summary of music terms.



### **OCTAVES**

(On the Commodore 64 octave 3 begins at middle C).






# MUSIC REVIEW


Scale





 = Treble clef


$\frac{4}{4}$  = (4) beats per measure  
 $\frac{4}{4}$  = (4) quarter note receives the beat


 = whole note (4 counts)


 = whole rest


 = half note (2 counts)


 = half rest


 = quarter note (1 count)


 = quarter rest

 = eighth note ( $\frac{1}{2}$  count)


 = sixteenth note ( $\frac{1}{4}$  count)

 = eighth rest

 = repeat sign

 = dotted quarter note (3 counts)  
 The value of the dot is  $\frac{1}{2}$  the value of the note to which it is added.

  
 ↑ 1 measure      ↑ end of section

 = tied notes

# APPENDIX C

## COMMODORE 64 DISC OPERATIONS

The model 1541 disk drive for the Commodore 64 is a small box with rectangular slits in the front. It is like a record player except that it plays thin, flexible disks (hence the name *floppy disk*). The disks, 5-1/4 in. (about 13 cm) in diameter, are often called *diskettes*, *minidisks*, or *minifloppies*. We just call them *disks*.

When turning on the computer that has a printer or disk drive, turn the individual items on in the following order:

1. Printer
2. Disk drive
3. Monitor (television set)
4. Computer

To start operating the Commodore 64, load the disk drive with a disk that has been prepared for computer use (or *formatted*). You may compare formatting to preparing a grade book, scorecard, or ledger sheet for later use. Put the formatted disk in the slot, with the label up and toward you. Pull the drive door (a small black lever in the center) down until it clicks into place. Be careful, you must push in and then up to open the door once it has been closed. Don't force anything! Once you have the disk in the drive and the door closed, the Commodore 64 is ready to operate.

We will now summarize some disk commands. Consult your owner's manual for more details.

### **FORMATTING A DISK**

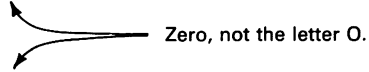
You must prepare (or *format*) a new disk before using it. You can also format old disks if you no longer want the programs on them.

WARNING: FORMATTING A DISKETTE DESTROYS ALL PROGRAMS ON IT.

To format a disk, enter OPEN 15,8,15, "NO:NAME,ID" where NAME and ID are any names you care to use. For example, you might use

```
OPEN 15,8,15,"NO:DISK1,D1"
```

for the first disk,



```
OPEN 15,8,15,"NO:DISK2,D2"
```

for the second disk etc. The name ID can only be two letters long. Formatting takes the Commodore about 30 to 45 seconds. You should label your disks after formatting them. Put the date you first used the disk on the label as part of the identification information.

## **OTHER DISK OPERATIONS**

1. To see what programs are on a disk (i.e., to load its directory), type

```
LOAD "$",8
```

followed by LIST. The listing may move by rapidly; you can use the CTRL key to slow it down or RUN/STOP to stop it completely. Note that loading a directory erases any program in memory.

2. To save a program on a disk (after entering it into the computer), type SAVE followed by the program's name (say P02-04 for Program 4 of Chapter 2) and the number 8. For example, enter

```
SAVE "P02-04",8
```

to save Program 4 of Chapter 2 for later use.

3. To load a program from a disk (you must have saved it earlier under its name), type LOAD, followed by the program's name and the number 8. For example, use

```
LOAD "P02-04",8
```

to load Program 4 of Chapter 2 from a disk.

4. To erase a program from a disk, enter

```
OPEN 15,8,15,"SO:NAME"
```

where NAME is the name of the program to be deleted. For example, to erase Program P02-05, enter

OPEN 15,8,15,"S0:P02-05"

Zero, not the letter O.

5. To stop the red light on the disk drive from blinking, enter

OPEN 15,8,15,"I"

6. To use the disk drive after you see the error message "FILE ALREADY OPEN", enter

CLOSE 15

7. To use the disk drive after you see the error message "DEVICE NOT AVAILABLE", enter

OPEN 15,8,15,"I"

If this instruction does not work, remove the disk, then turn both the computer and the disk drive off. Turn them back on (disk drive first), and reinsert the disk.

WARNING: NEVER TURN THE DISK DRIVE OFF WITH A DISK IN IT.

8. To SAVE a new version of a program (say P03-05), enter

SAVE "S0:P03-05",8

This saves the program under the name P03-05 and destroys the old program with that name.

# INDEX

- Advanced car race, 186**
- Advanced race track, 186
- Airplane, 26, 32, 123–25
- Alien, blinking, 45, 101
  - color, 190–91
  - moving right, 53
  - pulsating, 130
- Alien face, 27
- Alien spacecraft, 40, 60, 102
- Alphabetic characters, 73–75, 83
- Animation, 42–48
- Answering questions, 81–88
- Applause, sound, 179
- Applications, 1
- Areas on screen, 13
- Arithmetic quiz, 87
- Arrow drawing, 33
- Arrow keys (cursor control), 5, 8, 43, 49–50
- Art, random, 68, 195
- Artillery gun, 32, 151
- Asking questions, 81–88
- Attack, in sound, 95
- ♭ (flat in music), 169
- Back and forth motion, 27–28
- Background for pictures, 36
- Backward FOR-NEXT statements, 19, 24
- Backward index, 19
- Ball, 35–36
- Ball game, 37
- BASIC, 7
- Baton, and stick figure, 175
- Bee drawing, 34–35
- Birds flying, 160
- Birthday card, 77
- Blank line, 23
- Blanks around drawing, 36, 119, 127
- Blastoff, 40
- Blimp, 203–4
- Bomb, 116–21
  - Bomb, guided, 135
  - Bomb game, 116–25
- Border color, 197
- Box, 58
  - graphics, 58
- Branches, 87
- Buffer, entering, 137, 163
- Bug, 28, 42–43, 111–13
- Bumble bee, 34–35
- Capital letters, 5–6, 10**
- Car, 26, 33, 60–61, 110–11, 182–84
- Cassette, saving on, 16–17
- Cassette tape player, 3
- Cat, 78–79
- Catalog (*see* LOAD, a directory)
- Catch (game), 37
- Central Processing Unit, 3
- Characters, 3
- CHR\$, 67, 72
  - table, 72
- CHR\$(19), 23–24
- CHR\$(147), 22–23, 36
- Christmas card, 76
- Christmas tree, 76
- Circular motion, 194
- Clearing a program, 10
- Clearing the screen, 22, 25, 36, 68
- CLR/HOME, 22, 25
- Colon, 23
- Color, 5–6, 47–48, 188–99
- Color, notation, 189, 196

- Color keys, 5–6, 10
- Column (right-most column on screen), 32, 65–66, 75, 81, 121
- Column numbers (on screen), 4
- Commas in PRINT statements, 12–14
- COMMODORE key, 5–6, 10, 58
- Computer languages, 7
- Condition (in IF-THEN statement), 86
- Controlling keyboard, 89–93, 118–19, 162–63
- Controlling left or right motion, 119–20
- Controlling movement, 119–20
- Controlling speed, 138, 158
- Correcting errors, 9
- Countdown to launch, 40
- Cow's face, 29
- CPU, 3
- Creature, 45
- CRSR keys, 5, 8, 43, 49–50
- CRSR keys (to shorten typing), 23
- CTRL key, 5–7, 10, 43, 188
- Cursor, 4
- Cursor control, 52
- Cursor moving keys, 50
- Dancing stick figure, 176–80**
- DATA statement, 267–72
- Decay, in sound, 95
- Deleting, 7–9
  - character, 8–9
  - line, 7
- Diagonal motion:
  - left-to-right, 18, 126–30
  - right-to-left, 131–32
  - up-and-down, 126–34, 152–53
- Diagonal printing, 25, 37
- Dice, 64
- Digits, random, 64–68
- Directory (*see* LOAD, a directory)
- Diskette, 3
- Disk drive, 3
- Display character set, 72
- Division, 138
- Doctor's assistant, 86
- Dollar sign (in text names), 83
- Door, 34
- (DOWN), 50
- Easter bunny, 80**
- Easter card, 79
- Editing a program, 23
- END, 158
- Endless loop, 76, 161
- Entering a program, 8
- Entering CRSR keys, 51
- Enter key (*see* RETURN key)
- Equal sign (=), 13
- Erasing line, 7–8
- Errors, 9–10
- Errors, correcting, 9–10
- Exploding star, 47, 69
- Explosion (drawing), 124, 140, 157
  - sound, 124, 138, 156–57
- Eyes in drawings, 61
- Face, graphics, 61
- Factory, 151–58
- Fast (controlling speed), 138
- File names (standard), 16
- Firing command (submarine), 162
- Flat, in music, 169
- Flower drawing, 35
- Flying, 38
- FOR-NEXT statement, 13
  - backward, 19, 24
  - multiple, 27–29
  - nested, 29, 62
  - slowdown loop, 23
  - STEP, 19
- Four-letter words, 71
- Frequency, in music, 167
- Funny faces, 45–46
- Geography quiz, 88**
- GET, 89–93
- Gorilla drawing, 25
- GOSUB statement, 42–47
  - convenience, 148
- GOTO statement, 76, 161
- Graphics:
  - entering, 58
  - left-hand, 6, 10
  - mode, 6, 10–11
  - right-hand, 6, 10
- Graphics characters, 33
  - table, 59
- Graphics keys, 5–6, 10
- Grass, 46, 191
- Gravity, 164
- Greater than (>), 114, 121
- Grid, 4, 51
- Grid layout:
  - for alien, 52, 127, 132–34
  - for face, 54–55
- Ground (in drawings), 115
- Halloween card, 78**
- Halloween cat, 78
- Hard copy, 3
- Hardware, 2
- Heart, 76
- Homework notice, 82
- Homing the cursor, 23
- Horse grazing, 46, 191
- Houses, 206
- IF-THEN statement, 86–88**
  - examples, 87
  - statements after, 86–88, 119–21
- Illegal quantity error, 121
- Index, 13
- Infestation of bugs, 112
- INPUT statement, 73
- Inserting characters, 8–9
- INST/DEL, 5, 8, 10, 35
- Interfaces, 3
- Interview, 85
- INT statement, 64
- Inverse characters, 52, 85
- Inverse video, 6, 200–210
- Invisible characters, 6–7

**Kaleidoscope of color, 197**  
 Keeping drawings on screen, 32, 65–66, 75, 81, 121, 127  
 Keyboard, picture, 5  
 Keyboard, reading, 89–93  
 Keyboard control of programs, 89–93  
**Language of computer, 7**  
 (LEFT), 50  
 Left-hand graphics, 34  
 Left-to-right motion, 24, 37  
 Legal notice, 83  
 Less than (<), 121  
 Letters, random, 68  
 Limits of screen, 32, 65–66, 75, 81, 105  
 Line numbering, 7–8  
 LIST, 8, 10, 40  
     variations, 40  
 Listing a program, 40  
 Living art, 68, 195  
 LOAD, a directory, 18  
 Loading a program:  
     from cassette, 17  
     from diskette, 17–18  
 Location on screen, 4  
 Loop, 13–14, 62  
 Loop variable, 13  
 Lowercase letters, 6, 10  
     random, 68  
 Lunar station, 56, 89–93  
**Making faces, 45–46**  
 Mars warship, 41  
 Maze builder, 207–8  
 Measure, in music, 166  
 Memory, 3  
 Message board, 83  
 Message flashing, 202  
 Minus sign (–), in music, 169  
 Modules, 137  
 Monster, 25  
 Mother's Day card, 79  
 Motion picture, 42  
 Movement:  
     anywhere on screen, 144–50  
     back and forth, 27–28  
     diagonal, 126–34  
     in all directions, 141–50  
     left to right, 24, 37  
     right-left, 25, 37  
     to bottom of screen, 105, 111, 116–17  
     up-down, 113–14, 152–53  
 Moving up and down, 113–14, 152–53  
 Muffler, sound, 111  
 Multiple statements on line, 23  
 Multiplication, 138  
 Music, 166–80  
 Musical notes, 166, 168  
**Naming programs, 16**  
 Neon sign, 198  
 Nested loops, 29, 62  
 NEW command, 8, 10, 12  
 New Year's card, 77  
 NEXT statement, 13  
 Noise, 94–104  
 Nomenclature, 50  
 Normal screen, 48  
 Notation for color, 181, 196  
 Not equal to (<>), 118  
 Notes, 94  
     converting, 168–84  
     finding, 168–84  
 Null string, 90  
 Numerical answers, 83  
 Numeric variables, 83  
**Ocean waves, 160**  
 Octave, 166  
 Old Macdonald's farm, 168–72  
 On/off, 3  
 Out of data error, 168  
**Paratroopers, 114–15**  
 Pcc-nn (form), 16  
 Police car, 102  
 POKE (for color), 189–91, 197  
 POKE (for sound), 94–104, 160–67  
 Power supply, 3  
 PRINT CHR\$(19), 23–24, 34, 55  
 PRINT CHR\$(147), 22–23, 36  
 Printer, 3  
 PRINT statement, 11–15  
     commas, 12–14  
     quotation marks, 12–13  
     semicolons, 12–13, 15  
     to print a blank line, 23  
 Program, 7  
 Program names, 16  
 Program stub, 136  
 Program writing methods, 128, 137, 146, 158  
 Pulsating object, 129–30  
 Pumpkin, 78  
**Question mark, 23**  
 Questions, asking, 73–88  
 Quizzes, 87–88  
 Quotation marks, 12–13, 84  
**Rabbit, 80**  
 Race car, 110–11, 182–84  
 Race car game, 183–84  
 Race track, drawing, 181–87  
 Random art, 195  
 Randomizing, 71  
 Random numbers, 64–68  
 Random placement on screen, 64–66  
 READ statement, 167–72  
 READY, 12  
 Rectangular waveform, 95  
 REDO FROM START, 83  
 REM (remark) statement, 70  
 Repeating characters, 8  
 Rest, in music, 171  
 RESTORE key, 7, 9, 48  
 RESTORE statement, 168  
 Restoring screen, 77  
 RETURN key, 7, 9, 12  
 RETURN statement, 42–47  
 Reversed characters, 52, 85  
 Reverse video, 6, 200–10

(RIGHT), 50  
 Right-hand graphics, 6  
 Right-to-left motion, 25, 37  
 RND (random number), 64  
 Rocket, 31, 38, 144–50  
   graphics, 60  
   launch, 38, 98, 164  
   sound, 98  
 Rolling waves, 161, 208–9  
 Roulette wheel, 64  
 Row numbers on screen, 4, 64  
 Rows, 4, 64–65  
 Rows, bottom most, 4, 64–65, 105  
 RUN, 8, 12, 22  
 RUN/STOP key, 9, 29, 76  
 RVS OFF, 6–7, 10, 85, 200–10  
 RVS ON, 6–7, 10, 85, 200–10  
**Sailboat, 26**  
 SAVE, 16–17  
 Saving a program:  
   on cassette, 16–17  
   on diskette, 17  
 Sawtooth, in sound, 95  
 Schoolhouse, 206–7  
 Screen, 3, 13, 197  
 Screen areas, 3  
 Screen color, 197  
 Scrolling screen, 65–66, 105, 127  
 Secret orders, 199  
 Semicolon, 12–13, 15, 50, 67  
 Shadow elimination, 36, 119, 127  
 Shadows in pictures, 36, 119  
 Shell, 151  
 Shifted (, 76–77  
 SHIFT key, 5–6, 10  
 SHIFT LOCK, 6, 10, 22  
 Ship drawing, 159  
 Shortcut:  
   in PRINT, 23  
   in writing programs, 27–28  
 SID chip, 96  
 Simultaneous movement, 159–65  
 Siren, 102  
 Slow (controlling speed), 138  
 Slowdown, speedup, 138  
 Slowing screen down, 8, 10, 43  
 Slowdown:  
   FOR-NEXT, 65  
   using delays, 23, 41, 67  
   using sound, 158  
 Smoke drawing, 39  
 Snowfall, 65  
 Sound, 94–104  
   adding to program, 147, 154, 164–65  
   slowdown, 112, 158  
 Sound effects, 110, 112, 117  
   explosion, 154  
 Spacecraft, 40  
 Spaceship, 135–40  
 Spaceship game, 135–40  
 Space station, 193  
 Speed control, 164  
 Speed object up, 138  
 Spider drawing, 107–10  
 Spider web, 108–9  
 Standard name for programs, 16  
 Star, exploding, 47–69  
 Starting the Commodore 64, 11  
 Statement, 7  
 STEP (in FOR-NEXT statement), 19, 24  
 Stick figure, 36, 44, 175  
   dancing, 176–80  
**STOP, 158**  
 Stopping sound, 96, 113  
 Stopping the computer, 9–10, 77  
 STOP/RUN, 7  
 Stories, 81–82  
 String, alphabetic, 73–75, 83  
 String variable, 73–75, 83  
 Stub, program, 136  
 Submarine drawing, 159  
 Subroutine, 42, 137  
 Supernova, exploding, 69  
 Syntax error, 9  
**TAB (0), 19**  
 TAB statement, 18, 32, 49  
 Tank, 27, 30–31, 62, 103, 205–6  
 Tank command, 103  
 Targets, 33, 136  
   factory, 151–58  
   random location, 147  
 Tax collector, 86  
 Television display, 3  
 Tempo, in music, 172  
 Testing programs, 136–37, 143  
 Text mode, 11  
 Thrust, 150  
 Top-down design, 136  
 Town, 33  
 Track, racing, 181–87  
 Trampoline analogy (to FOR-NEXT statement),  
   13  
 Tree, 76  
 Triangular waveform, 95  
 Truck, 62–63, 204–5  
 Turning computer on, 6  
**(UP), 50**  
 Uppercase characters, 10  
**Valentine heart, 76**  
 Video display, 2  
 Volume, 94  
**Waiting for any key, 89–93**  
 Walking, 42–44  
   bug, 42  
   stick figure, 44  
 Warship, 41  
 Wave, ocean, 160, 208–10  
 Waveform, in sound, 95  
 Web, spider, 108–9  
 Window, 34  
 Word answers, 81–86  
 Wrap around, 32, 65–66, 75, 81, 121, 127  
**X, giant, 20**  
**Yankee Doodle Dandy, song, 172–80**





# TONY FABBRI ANIMATION, GAMES, & SOUND FOR THE

# 64 COMMODORE

*SAILBOAT MOVING LEFT. . . BEE FLYING RIGHT TO FLOWER. . . EASTER BUNNY. . . OCEAN, BIRDS, SHIP, AND SUBMARINE, WITH LAUNCH. . . OLD MACDONALD'S FARM. . . CAR RACE WITH ADVANCED RACE TRACK. . . BLIMP WITH BLINKING MESSAGE*

These are just a few examples of the dozens of fascinating programs featured in Tony Fabbri's new book. It will show you how to draw pictures and figures, make them move, and create game action with keyboard control, moving targets, noise, and explosions. The author guides you step by step through every aspect of simple arcade games; even if you have no background in computers or programming, you will soon be making sailboats and tanks move, launching rockets and spaceships, making eyes blink and stars explode, creating a game of catch, producing your own greeting cards, and devising your own colorful action games.

### **Some special features:**

- includes over 180 complete programs you can use to create new games and handle practical applications;
- lets you do things and see the results rather than just read about concepts or methods;
- provides simple, straightforward examples that you can easily expand;
- teaches the programming skills you need to solve business, educational, engineering, management, and scientific problems;
- shows you how to create many of the effects you have seen in arcade and computer games, on television, and in movies;
- assumes no special background, requires no extra equipment, and does everything in elementary BASIC.

PRENTICE-HALL, INC., Englewood Cliffs, N.J. 07632

ISBN 0-13-037375-3