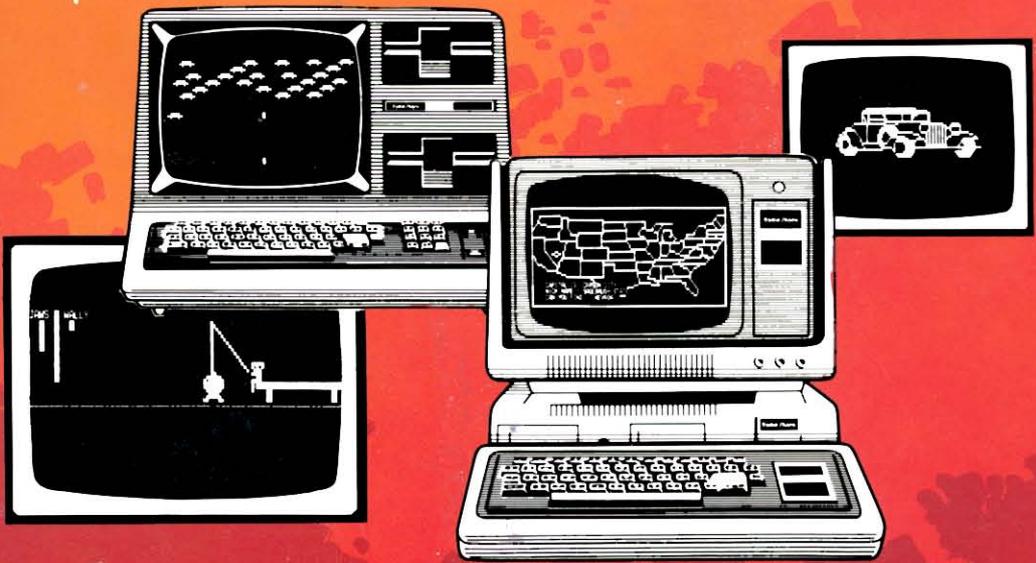


**Tom Dempsey**

# **GAMES and GRAPHICS FOR THE TRS-80®**

## **a BEGINNERS' GUIDE**



**For Models I & III**

**JG**



**Tom Dempsey**

# **GAMES and GRAPHICS FOR THE TRS-80<sup>®</sup>**

---

## **a BEGINNERS GUIDE**

Charles Trapp – Editor

D. J. Smith – Cover Design and Graphics

First Edition  
First Printing  
April, 1983  
Printed in the United States of America

Copyright ©1983 by IJG Inc.

ISBN 0 936200 10 1

All rights reserved. No Part of this book may be reproduced by any means without the express written permission of the publisher. Example programs are for personal use only. Every reasonable effort has been made to ensure accuracy throughout this book, but neither the author or publisher can assume responsibility for any errors or omissions. No liability is assumed for any direct, or indirect, damages resulting from the use of information contained herein.



Published by  
**IJG Inc**  
1953 West  
11th Street  
Upland, CA  
91786 / 714/  
946-5805

Radio Shack and TRS-80 are registered trademarks of the Tandy Corporation

# **IMPORTANT**

## **Read This Notice**

Any software information is used at your own risk. Neither the PUBLISHER nor the AUTHOR assumes any responsibility or liability for loss or damages caused or alleged to be caused, directly or indirectly, by applying any information or alteration to software described in this book, including but not limited to: any interruption of service, loss of business, anticipatory profits or consequential damages resulting from the use or operation of such software information or alterations. Also, no patent liability is assumed with respect to the use of the information contained herein. While every precaution has been taken in the preparation of this book, the PUBLISHER and the AUTHOR assume no responsibility for errors or omissions. The reader is the sole judge of his or her skill and ability to use the information contained in this book.

---

# Reading the Programs

---

The BASIC listings in this book were printed using the program DOCLIST/BAS from *Basic Faster and Better* by Lewis Rosenfelder (available from your local computer store, bookseller, direct from IJG Inc., or Radio Shack, cat. no. 62-1002). DOCLIST/BAS puts each statement on a separate line, inserts spaces between each key word and indents IF-THEN statements and FOR-NEXT loops. In addition, DOCLIST prints a solid underline after each section in the logic, and a dotted underline is used to highlight breaks in the logic.

Also, DOCLIST has a heading which shows the program name and page number, and it correctly processes programs which use the down arrow to provide a line feed to the next line. Blanks which may be in the program are ignored unless they are within quotes or remark statements. We believe DOCLIST enhances the readability of BASIC listings as well as making the program logic more transparent.

Below is a sample program listing before using DOCLIST/BAS.

---

```
51 X=5712:Y=0
52 C=PEEK(X):IFC>127THENPRINT@544,RW$(Y),:Y=Y+1:IFY>123THEN55ELS
ERW$(Y)=CHR$(CANDNOT128):GOTO54
53 RW$(Y)=RW$(Y)+CHR$(C)
54 X=X+1:GOTO52
55 RW$(16)=RW$(16)+""
```

---

Here is the same program after using DOCLIST/BAS.

```

51 X = 5712:
    Y = 0
52 C = PEEK (X):
    IF C > 127
    THEN PRINT @544,RW$(Y),:
        Y = Y + 1:
        IF Y > 123
        THEN 55: .....
        ELSE RW$(Y) = CHR$(C AND NOT 128):
            GOTO 54 .....
53 RW$(Y) = RW$(Y) + CHR$(C)
54 X = X + 1:
    GOTO 52
-----
55 RW$(16) = RW$(16) + ""

```

Note the solid underline, separating lines 54 and 55, which indicates that the logic never falls directly through from 54 to 55 in this program. The dotted lines in line 52's IF-THEN statement indicate possible logic breaks, but the absence of a solid underline before line 53 shows that conditions exist in which the logic will fall through from line 52 to 53.

When typing in programs from this book, data statements should not have spaces inserted where DOCLIST has broken the lines for readability.

# Preface

---

This book is especially designed for the new computer owner, as well as those who have been programming for years.

If you enjoy playing computer games, watching good graphics or perhaps would like to learn a few programming tricks, then this book is for you. If you also happen to have children who love computer games, the games in this book will keep them busy for days.

The microcomputer can do a great many things — keeping track of expenses, keeping your records straight, accounting, designing, estimating, educational uses, or even dialing your telephone. But among the many, many uses for a home computer, the most enjoyable is just playing games. The computer can be programmed to play any game you can think of.

The object of this book is not only to give you program listings that you can enter and run, but, more importantly, to give you ideas on how to make your own computer games using animation and graphics.

If you have been programming for awhile but would like some new ideas on how to create realistic-looking graphics and animation for your games, the programs in this book will help you. Each listing contains new ideas and methods for giving animation and graphics to your games.

In the game **SLOT MACHINE**, you'll see fast graphics animation using only **SET** and **RESET**. And the game **SPEED RACER** will show you new ways of displaying text or instructions on the screen. We'll investigate the use of machine-language subroutines to speed up the graphics, as well as ways to let the computer do some of the work for us by using some simple programming utilities.

As we write our games, we will be concentrating on a few (but very powerful) routines that are available in **BASIC** and especially useful for graphics:

1. **Memory Usage** — how programs are stored in memory, special address locations and the video display.

## Acknowledgment

2. Graphic Codes — using the 64 graphic codes and alphanumeric graphics.
3. Programming Graphics — making the most out of SET and RESET, letting the computer figure out the PRINT@ and POKE positions, and using string packing so our graphics characters are in the program lines to increase the speed of the graphics. At the same time, we don't have to reserve or clear any space in our programs for these graphics because they are actually contained in the program lines themselves.
4. Block Move Routines — which will move our graphics from the program lines and place them on the video screen.
5. Drawing and Animation — programming utilities that we can use for creating our graphics characters and giving them animation.
6. Ideas and Suggestions — tips on how to create your own games by using the utilities in this book, to help you turn your ideas into reality.

I hope you enjoy reading and using the ideas in this book as much as I have enjoyed writing and playing the games with my family.

## ACKNOWLEDGMENTS

I would like to thank:

My wife Judy, who let me quit 15 years of servicing office machines and supported our family while I sat in front of my computer writing games.

Bob Jones (Interface Age), who looked at my first manuscript and instead of telling me to forget it, gave me the encouragement I needed to keep going.

Vince Oehrlein, who introduced me to Electric Pencil, which ended up changing my method of earning a living.

And Harv Pennington, whose creative genius and compassion for his fellow man helped make this book a reality.

Thomas Dempsey

---

# Table of Contents

---

<b>Reading the Programs:</b> .....	3
<b>Preface:</b> .....	5
<b>Acknowledgements:</b> .....	6
<b>Why:</b> .....	11
<b>Introduction</b> .....	13
<b>Chapter 1:</b>	
Eight Bits Equal One Byte .....	15
Point Me In The Right Direction .....	15
<b>Chapter 2:</b>	
What Kind Of Character Are You? .....	19
Divide And Conquer .....	19
A Different Kind Of Graphics .....	21
A Little Imagination Goes A Long Way .....	22
<b>Chapter 3:</b>	
How To Get Ideas .....	25
Creating A Character .....	25
The Right Background .....	25
Where's The Action? .....	26
Giving Your Characters Some Character .....	26
A Game Called JAWS .....	26
The Object Of The Game .....	26
Programming Tricks .....	27
<b>Chapter 4:</b>	
Methods of Display Using PRINT @ .....	35
ROBOT BUILDER .....	35
NEWS LETTER Program .....	40
Operating Commands for NEWSLETTER .....	40

## Contents

<b>Chapter 5:</b>	
Methods of Display Using SET/RESET .....	47
SLOT MACHINE .....	47
Program Explanation .....	51
SHOOTING COWBOY .....	51
Program Explanation .....	55
<b>Chapter 6:</b> .....	57
Methods of Display Using FOR NEXT .....	57
TANK CHASE .....	57
Program Explanation .....	59
TWITS IN WATER .....	59
Program Explanation .....	61
HORSE RACE .....	61
Program Explanation .....	64
RACETRAC .....	65
Operating Instructions .....	65
Program Explanation .....	66
<b>Chapter 7:</b>	
Methods of Display Using Block Moves .....	77
FIND THE STATES .....	77
Program Explanation .....	85
Las Wages POKER .....	85
Program Explanation .....	93
Computer YAHTZEE .....	94
<b>Chapter 8:</b>	
Working With Strings .....	109
AUTO/GRAPHICS Compiler .....	110
How to use AUTO/GRAPHIICS .....	111
Program Explanation .....	112
DATA/GRAPHICS Compiler .....	115
<b>Chapter 9:</b>	
MAKE A MAZE .....	123
Load GRAPHICS PACKER .....	123
Finding The Starting Address .....	124
Running GRAPHICS PACKER .....	124
Moving Around the Screen .....	126
Testing the Packed String\$ .....	126
Testing the Sound .....	126
Program Control .....	127
LISA .....	133
Program Explanation .....	135
<b>Chapter 10:</b>	
Graphic Art and Computers .....	137

1932 FORD ROADSTER .....	137
VIDEOHEX .....	139
Program Explanation .....	142
<b>Chapter 11:</b>	
Educational Games .....	143
MORSE CODE .....	146
SUGGESTED CHANGES .....	146
SILVER WEIGHT CONTENT .....	147
MEMORY MAGIC .....	148
<b>Chapter 12:</b>	
Puzzle Games .....	153
LAST STRAW .....	156
<b>Chapter 13:</b>	
Arcade Games .....	159
SAUCER INVASION .....	159
Explanation Of SAUCER INVASIONS .....	161
SAUCER WARS .....	162
SPEED SHIFTER .....	166
<b>Chapter 14:</b>	
Pushing BASIC to Its Limits .....	169
Defining the Robots .....	169
Marching Robots .....	169
Checking the Robots .....	169
Moving and Shooting .....	170
I'm Working as Fast as I Can! .....	170
Easy Does It .....	177
SERVICE Records Program .....	177
Program Explanation .....	178
<b>Chapter 15:</b>	
Graphic Utility Programs .....	181
LSET 1K .....	181
STRING/FIX .....	182
Packed Strings to CHR\$ Converter .....	184
BASIC LINE MOD .....	186
LINEFINDER .....	187
DISKFILE .....	188
ENLARGE .....	192
COPY .....	194
How to Use COPY .....	194
Program Execution .....	196
PCOPY .....	196
How To Use PCOPY .....	196

**Chapter 16:**

Teaching Your Computer New Tricks ..... 199

Programming Tip 1 ..... 199

Programming Tip 2 ..... 199

Programming Tip 3 ..... 199

Programming Tip 4 ..... 199

Make Your Computer Do It ..... 199

Convert PRINT @ to X,Y Positions ..... 200

Convert X,Y to POKE Positions ..... 200

Convert POKE Positions to PRINT @ ..... 200

Find the Start of the BASIC Program ..... 200

Find the End of the BASIC Program ..... 200

Find the Computer Memory Size ..... 200

Making Programs Unlistable ..... 200

Tape or Disk? ..... 200

Unpacking Packed Strings ..... 200

Machine-language Subroutine Packer ..... 201

**Chapter 17:**

Machine-language Subroutines ..... 203

There Must Be a Better Way ..... 203

One Step at a Time ..... 204

The Merging of BASIC and Machine-Code ..... 205

Machine Language Subroutine Packer ..... 206

LDVIDEO ..... 207

REVIDEO ..... 207

SOUND ..... 208

RSCROLL ..... 208

SCRMEM48 ..... 208

MEMSCR48 ..... 208

SCRMEM16 ..... 208

MEMSCR16 ..... 208

LOADTAPE ..... 208

SAVETAPE ..... 208

**Chapter 18:**

Computer Animation ..... 209

Character Generators ..... 209

The Graphics Generator ..... 209

STRING WRITER ..... 210

**Chapter 19:**

Compu/Mation ..... 215

What's in a Name? ..... 215

Operating Instructions ..... 216

**List of Games and Graphics ..... 235**

---

# Why

---

I was born in Chicago, Illinois on May 4, 1943 and lived there for about 2 years — just long enough to fall off the kitchen table and break my left arm. I guess that has a little to do with why I can do things just as well with either hand. Everything but writing, that is — my writing is so bad I have to print everything so I can read it *myself!*

My family moved to a small, 1-acre chicken farm in Downers Grove, a suburb of Chicago. I always called it a chicken farm because there were more chickens than stray cats running around, although at times it was hard to be sure.

I look back at that stage of my life as perhaps the best and worst times of growing up. At that time I had three brothers, one sister, about 100 chickens (99 chickens — a chicken hawk carried another one away last night), 40 cats, 1 dog, 5 ducks, 4 turkeys and one old work horse.

The chickens provided us with eggs most of the time, and I could write a book on all the ways our dear mother found to cook chicken. The cats gave us kids the most enjoyment. We used to throw them up in the air just to see if they would always land on their feet — they did, except for the one we threw over the house. That one never came back!

The cats always teased the German Shepherd until the day he got smart — instead of running out to the full length of his chain he started stopping about a foot short. I don't have to tell you the rest of the story . . .

The ducks were cute, the turkeys we saved for that special day, and the old horse walked around the post he was tied to until he just fell over and died.

I think every child should be able to experience growing up on a farm with no running water, a coal stove that has to be fed in the middle of the night so you won't wake up frozen, and a little house about 100 feet from the main house that we used to visit in the middle of the night with a flashlight in one hand and a newspaper in the other to chase the cat off the seat. This kind of life tends to give one a much deeper appreciation of the finer things in life. I enjoyed it as a kid, but I don't think I would ever go back.

After about 11 years on the farm, fighting asthma and hayfever so bad that I had to sleep sitting straight up, we moved to Lombard, Illinois, another suburb of Chicago. We were really living in style now — we even had running water! I lived there until I was out of school. In my first year of working for a living I held over 18 jobs, which I think must be some kind of record!

While I was busy finding myself, I met a woman, and we decided to get married three months later. We're going on 20 years now — and they said it would never last! I kept trying out different jobs at the rate of about 10 per year, and then I slowed down. I'm not sure whether I got tired or I just ran out of places, but one time when I was out of work I walked into a company called Smith-Corona, where I repaired office equipment for about 20 years.

One thing I noticed was that off and on during my office-machine repairing days I always kept going back to artistic types of jobs. I worked for Sears & Roebuck as a paste-up and layout artist for one of their 17 vice presidents. I painted oil paintings, — one of my famous paintings is hanging in my mother's house.

My technical training was on IBM, Royal, Smith-Corona, Remington, Adler and Olivetti typewriters, as well as Smith-Corona, Royal, Adler, Remington, Minolta, Olivetti and 3-M copy machines. One of my most enjoyable jobs was as a regional service instructor for Graphic Communications, but I had to leave that position when I discovered I didn't like flying.

Then it happened! I was out of work for about 3 months one time (I thought of it as early retirement) and I was walking past a Radio Shack store, when I looked through the window and saw a computer.

I went into the store and asked the owner a few questions like, "Just how much can I do with this computer?" The owner told me he had his own machine and that it was possible to do just about anything you could imagine! I thought to myself, "Wow, with an imagination like mine that means just about anything would be possible!"

I must have stood in front of that computer for 3 hours just thinking about what commands I could give that machine, when a little light turned on inside my head and I said to myself, "This machine can take my creative ideas and turn them into real pictures." It's like fixing office machines, it's like building skyscrapers with Tinker Toy parts and never running out of parts, it's like painting on a canvas and never running out of paint . . .

Well, the rest of the story is just beginning. I took out my life savings (about \$600.00), told my wife I might as well learn something new while I was out of work and told the owner of the store to gift wrap it! That was three years ago, and for the last three years I've spent no less than 14 hours a day, 7 days a week in front of this extension of my mind. I wonder if there are computers in heaven?

# Introduction

---

What is it that makes games (or to be more specific, — games using graphics) so enjoyable to so many people?

The human brain is a very hungry thing, and even more interesting is the fact that we can never fill it or use it to its full capacity. I can even see a small similarity between the brain and one of the computer spelling programs I have.

Each time I tell the program to remember a new word, the program saves this new word on the disk, and as the program keeps learning new words, it keeps pushing the old words that I don't use very much farther and farther back in the file until they disappear.

Our minds work in a similar manner, except that they don't ever completely forget anything. The mind keeps pushing information farther back too, but it saves all these bits of information and also has the ability to combine and compact this code when needed, — or as much as is needed at any given moment.

Whenever we need to restore this information, we might only need to take a quick refresher course on an old subject and BINGO — our minds bring all this old information back into the main working-memory area. Sounds like I'm talking about a computer with disk drives, doesn't it? I doubt very much that we will ever come close to making a computer as good as the human brain.

I guess what I'm trying to say is that the mind never seems to get enough information. People are not satisfied with just hearing music, — they also want to see music with their eyes and feel it. This urge for new ways to experience different things in the world is what I believe is the key to what makes games using graphics and sound so popular. The more graphics, sound, and animation we get, the more these games become extensions of the world we live in.

Games and graphics are all around us. Everything we look at is graphics, and I won't even try to mention all the different types of games people play, not only with other people or computers, but with themselves also.

## Introduction

Sometimes a game might create a world we would like to live in and experience, but we can only do so through our computers. I think the day is coming when people will no longer be satisfied with merely watching movies or computer simulations.

The future will bring people who want to actually live out experiences in real-life fantasy worlds which man will create. Someday, we may go to a place like Disneyland, sit down in the cockpit of a spaceship fighter-simulator and fly out to other worlds, shooting our laser-beam guns and conquering other solar systems.

---

---

# 1

---

## Eight Bits Equal One Byte

A shave and a haircut used to cost 1/4 of a byte. Today it's more like a sting than a byte! The TRS-80 saves numbers in memory as bytes.

The largest number the computer can save in one byte (or one memory location) is 255. If we want to save a number like 15360 in memory, we have to break this large number into two smaller numbers. The method we use to get these two numbers is shown below:

$$\begin{aligned} \text{MS} &= \text{INT}(15360/256) \\ \text{LS} &= 15360 - \text{MS}*256 \end{aligned}$$

LS is just a variable we use to store the Least Significant Byte, and MS contains the Most Significant Byte. The Least Significant Byte is saved first in memory, and the Most Significant Byte is saved in the following memory location. So our number 15360, saved in memory starting at (let's say) memory location 17129, would look like this in memory:

```
17129 0
17130 60
```

## Point Me in the Right Direction

The following is a BASIC program and how it looks in memory:

---

Figure 1.1 A BASIC Program in Memory

```
10 CLS
20 PRINT"HELLO"
30 END
```

```
27206 76      LS AND
27207 106     MS OF STARTING ADDRESS OF LINE 20
27208 10      LS AND
```

**Point Me in the Right Direction**

```
27209 0      MS OF LINE NUMBER 10
27210 132    CLS
27211 0      END OF LINE
27212 89     LS AND
27213 106    MS OF STARTING ADDRESS OF LINE 30
27214 20     LS AND
27215 0      MS OF LINE NUMBER 20
27216 178    PRINT
27217 34     "
27218 72     H
27219 69     E
27220 76     L
27221 76     L
27222 79     O
27223 34     "
27224 0      END OF LINE
27225 95     LS AND
27226 106    MS OF STARTING ADDRESS OF NEXT LINE
27227 30     LS AND
27228 0      MS OF LINE NUMBER 30
27229 128    END
27230 0      END OF LINE
27231 0      LS AND
27232 0      MS = NULL END OF PROGRAM
```

---

Right about now you're probably saying okay, but how does the computer know how to find the first line in the program? That's easy, — it saves the starting address of the program in memory locations 16548 and 16549, and it saves the end of the program in locations 16633 and 16634.

Any time we need to know where our program starts, we can use the following command:

```
PRINTPEEK(16549)*256+PEEK(16548)
```

If we need to know where the program ends in memory, we can use this command:

```
PRINTPEEK(16634)*256+PEEK(16633)
```

To find out how much memory we have left to use, we can use this command:

```
PRINTMEM
```

If we want to find out what size machine our program is running in, we can use this command:

```
PRINTPEEK(16561)+PEEK(16562)*256
```

To check whether the computer has tape or disk, we can use this command:

```
IF PEEK(16396) = 201 THEN IT'S TAPE  
ANY OTHER NUMBER MEANS DISK
```

If we want to PEEK or POKE into memory locations higher than 32767, we can use the following:

```
A = -1*(65536 - ADDRESS) : PRINTA
```

The following short program will search through memory and print out a list of keywords for you:

---

Figure 1.2 *Keyword Search Program*

```
10 CLS  
20 FORX=5711TO6175  
30 P=PEEK(X) : IFP>90THENP=P-128 : PRINTA$, : A$=""  
40 A$=A$+CHR$(P) : NEXTX
```

---

# NOTES

# 2

---

## What Kind of Character Are You?

The TRS-80 uses memory locations 15360 through 16383 for the video monitor. This 1K of memory is what we will be working with most in our games or graphics programming. You have to get to know and respect it in order to get the most out of it.

Some of its don't rules are listed below:

1. Don't PRINT @ 1023.
2. Don't POKE Lower than 15360.
3. Don't POKE higher than 16383.
4. Don't SET X coordinates lower than 0 or higher than 127.
5. Don't SET Y coordinates lower than 0 or higher than 47.

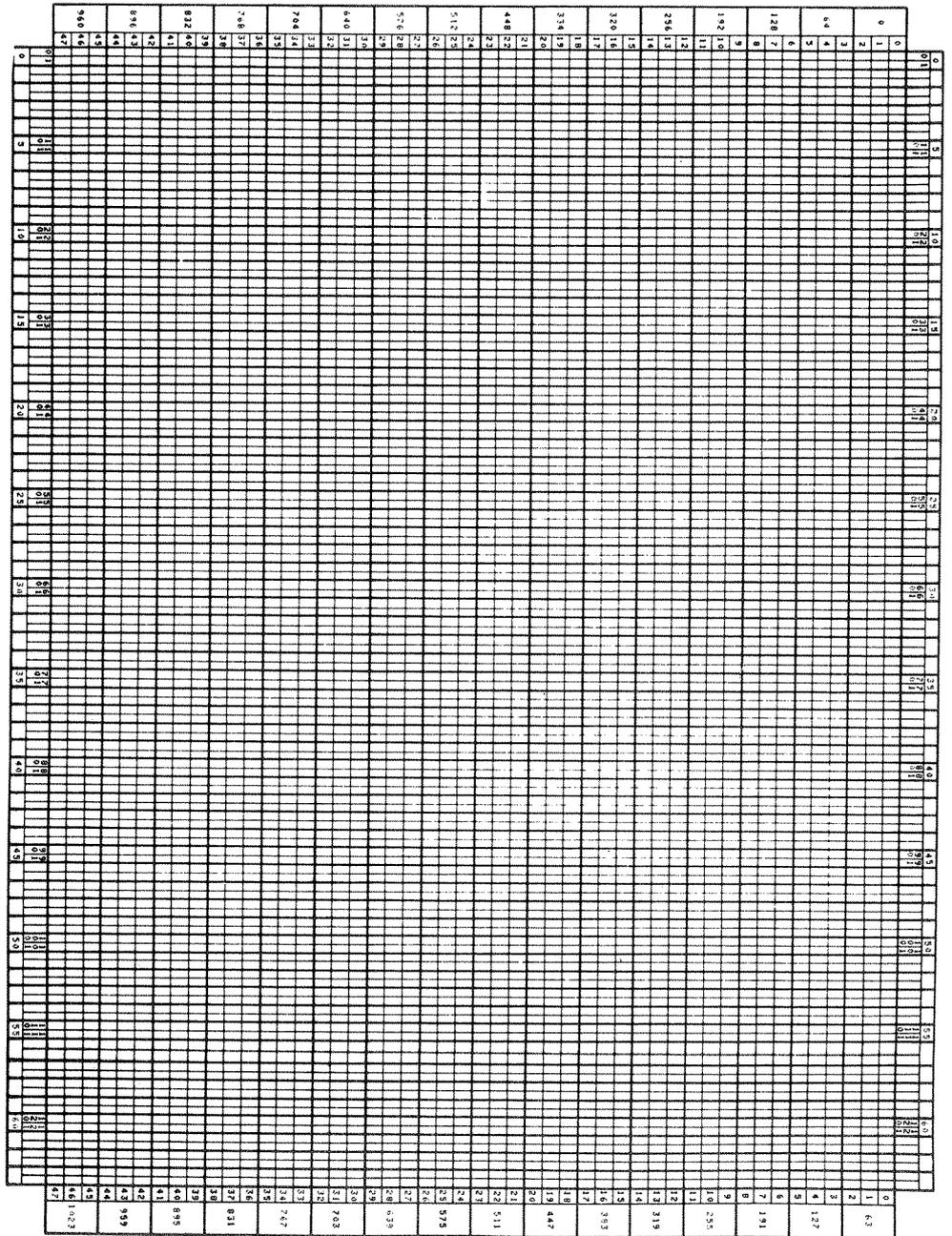
## Divide and Conquer

The screen is divided into 64 PRINT positions across by 16 rows down, and it is further divided by 127 SET positions across and 47 SET positions down.

We have 64 graphics characters to work with, as well as the alphanumeric characters which we can use when we need higher resolution for certain graphic effects.

Divide and Conquer

Figure 2.1 Video Graphics Worksheet



The following short program will display the 64 graphics characters on your screen.

---

Figure 2.2 Graphics Display Program

```

10 CLS
20 FORX=128TO191:PRINTX;" ";CHR$(X);
30 T=T+1:IFT=9THENPRINT" ":T=0
40 NEXT
50 GOTO 50

```

---

This program was originally a one-liner, but for demonstration purposes I kept it nice and neat.

### A Different Kind of Graphics

The following program is called MACHINE PARTS and is a good example of the use of alphanumeric graphics.

As a fellow computer-lover, I hear quite often from other programmers that the screen is not very good for doing any kind of graphics that requires finely-detailed pictures. I believe the graphics are there, but it's just a matter of learning how to express them in a new way. For this reason, I wrote MACHINE PARTS GRAPHICS. The computer is able; it's just a matter of learning either how to work around a few things or new ways to write the program. We are not limited to 64 graphics characters. We also have all our letters and numbers to make use of.

This program does just that. I think if it's possible to draw machine parts, it should be possible to express any graphics picture. The important thing to remember is this: It's possible for our eyes to see things that are not really there or do not exist at all! With this fact in mind, all we must do is create a close resemblance of what we want the players to see (the *appearance* of motion) and let their eyes and minds do the rest. The human eye is much slower than the computer graphics; our brain retains a picture of what the eye sees long after it has gone from view.

This program uses these rules to our advantage. There are many more ways to take advantage of tricks on the eyes. The main idea I'm trying to convey in this short program is this: Don't neglect to make the most of what you have to work with!

In MACHINE PARTS GRAPHICS, I have used the combination of graphics characters and alphanumeric characters to form a picture which we can interpret as machine or mechanical parts.

### A Little Imagination Goes a Long Way

You don't have to be an artist to make use of these principles. What you must do is learn to see things and shapes in their simplest forms.

First, look at the object you want to duplicate as a complete unit. Then draw a mental outline of the complete picture. The outline must contain the important features of the picture. Let's take, for example, a picture of a horse.

Now at first, you might not think of a horse as either being very mechanical or having the least resemblance to a machine, but the beauty and speed of a horse is a very great achievement. First, we have a very long body with two legs at each end, — very long and skinny legs for such a large body. And, much to our amazement, this animal can run very fast. Not only that, but it can also do this while carrying a rider on its back!

Okay, we have thought about this horse long enough. Now it's time to put his picture on the screen. First, we want to draw the outline of a horse. It's not important to show the fine details, such as the eyes or ears at this stage, — if we can just get the outline of a horse on the screen, everyone will know what it is.

If we want to show movement, we must show what creates this movement, but it's going to be hard to show the bones or muscles of the horse with our graphics. But wait! There's an easier way, — all we have to show is the shadows of those things.

The eyes can see an object not only by looking at the object itself, but also by looking at the absence of the object. Take, for example, a square drawn in the middle of your screen. We can draw the square and fill it completely in, or we can fill in everything around our square. The end result is still a square in the middle of the screen. Thinking negatively isn't always bad . . .

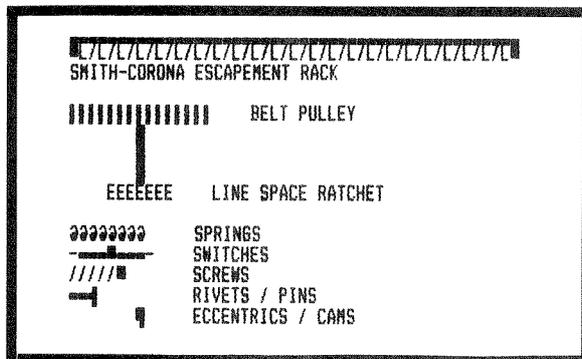


Figure 2.3 *Machine Parts Program*

```
10 CLS
15 CLEAR 1000
20 A$ = CHR$ (143) + "L/L/L/L/L/L/L/L/L/L/L/L/L/L/L/L/L
  /L/L/L/L/L/L" + CHR$ (143)
30 B$ = STRING$ (47,176)
40 PRINT B$
50 PRINT A$
60 PRINT "SMITH-CORONA ESCAPEMENT RACK"
65 PRINT
70 A$ = STRING$ (15,149)
80 B$ = "      EEEEEEE"
90 PRINT A$ + "      BELT PULLEY"
100 FOR X = 1 TO 3:
      PRINT TAB( 7) CHR$ (191):
      NEXT
110 PRINT B$ + "      LINE SPACE RATCHET"
120 PRINT
130 A$ = "@@@@@@@@ SPRINGS"
140 B$ = "-" + STRING$ (3,140) + CHR$ (143) + STRING$ (3
  ,140) + "-" SWITCHES"
150 PRINT A$
160 PRINT B$
170 A$ = "/////" + CHR$ (143)
180 PRINT A$ + "      SCREWS"
190 A$ = CHR$ (140) + CHR$ (140) + CHR$ (174)
200 PRINT A$ + "      RIVETS / PINS"
210 A$ = CHR$ (175)
220 PRINT "      " + A$ + "      ECCENTRICS / CAMS"
500 GOTO 500
```

# NOTES

# 3

---

## How to Get Ideas

The creation of a game has first to start with some idea of what type of game you're trying to write. Let's list some of the different types of games we could write.

WORD GAMES  
NUMBER GAMES  
GRAPHIC GAMES

If we were to place all games into one of these three categories, we could say that a game like POKER would fall under the heading of a NUMBER GAME. At the same time, if we make pictures of cards on the computer screen, then we might call the game POKER a GRAPHIC NUMBER GAME.

## Creating a Character

Since most of my games are about different types of characters, let's see how I go about creating a game program. The first thing I do is try to think of a graphic character. Most people would all look the same using the low resolution of the video monitor, so we will overlook using some of the characters we might already know.

How about animals, — any kind of animal will do! Maybe a horse, dog, cat, or a mouse (Oops! Sorry about that Walt). How about a *fish*? Okay, let's use a fish for our character.

## The Right Background

Now comes the big question. What does a fish do? Think about this for a moment, and also think about what other people would give you for an answer to this question.

The first thing that comes to my mind about a fish is water. Most fish that we know swim in the water, right? Now what else do most fish do? Who was that in the back of the room that said, "Have babies!"

## The Object of the Game

Well, since we don't want to end up with a screen full of little fish in our computer, we could say that fish eat food. But since dropping fish-food into a fish tank is not my idea of an exciting day, let's forget about what else fish do and think for a moment about what people do to fish . . .

### Where's the Action?

People catch fish, right? Or I should say most of us have *tried* to catch a fish. Now we already have the most important pieces for our new game.

First, the fish is swimming in the water, and second, the player is trying to catch the fish. Wow! I think we have a good idea for a complete game. Let's get out our video worksheets and draw a picture of a fish. Try not to make him too square-looking.

### Giving Your Characters Some Character

See, you don't have to be a creative genius to think up a character for a game, — although it wouldn't hurt if we *were*.

Now that we have our fish, we should try to give him some kind of personality. Let's make him clever by letting him swim around in the water and give him the ability to swim in any direction he wants.

We can make him smart by letting him swim faster when the hook is in the water with the worm on it, and we can make him look real by letting him open and close his mouth any time he wants to. We can even make him look funny by giving him a big grin if he beats us or wins the game. Let's see how our fish game turns out when it's all done.

### A Game Called JAWS

This program should really be called JAWS PART 2. In the original version of this game, I had the player dropping food into a fish tank, and wherever the food was dropped, JAWS would go after it and eat the food.

This version also became very messy because JAWS didn't eat his food all the time. We soon realized the need for a snail in the tank with JAWS. Then one day, my wife said she thought JAWS was lonely and that I should add another fish to keep JAWS company.

You guessed it! JAWS kept trying to eat the nice little angel fish I put in the tank! That's how this version of JAWS came about.

### The Object of the Game

The object of the game is to drop your hook into the water, and just before JAWS gets too close, pull your line out of the water by pressing the up-arrow key. If your timing is good, you can catch JAWS on your hook!

To continue playing, just press the down-arrow quickly, and JAWS will drop back into the water. Press the down-arrow a second time, and your hook will go into the water again.

**Programming Tricks**

One of the programming tricks used in JAWS to make the graphics faster is that when we print our graphics for JAWS on the screen, we also print a character 128 before and after JAWS.

This eliminates the need to have an extra print statement in our code just to clear JAWS each time we move him across the screen. This method clears the previous JAWS each time we print the new JAWS.

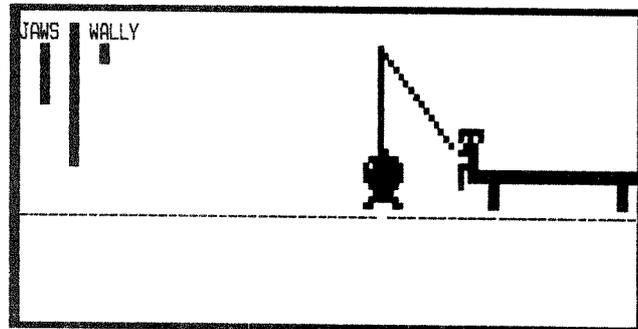
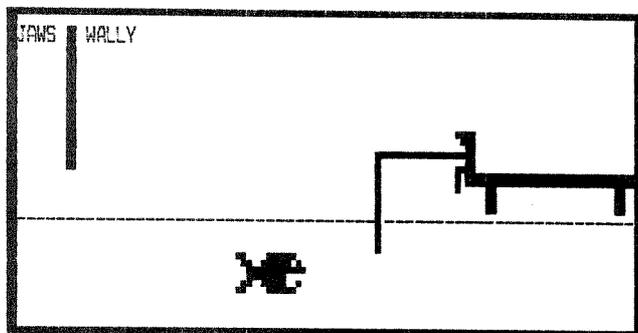
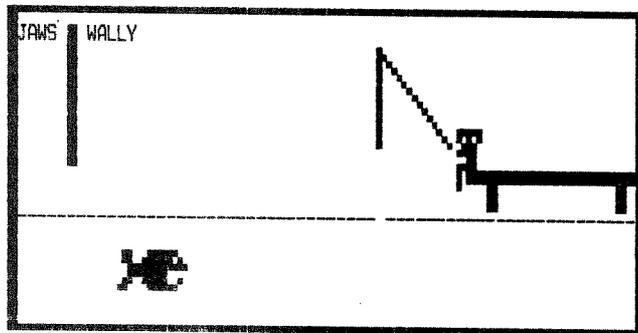


Figure 3.1 *Jaws Program*

```

10 REM  *** JAWS BY TOM DEMPSEY ***
15 REM  =====
      PRINT INSTRUCTIONS
      =====
20 CLEAR 1000
30 T = 131:
   WO = 137
40 CLS :
   PRINT CHR$(23):
   PRINT @462,"**** J A W S ****":
   FOR X = 1 TO 1000:
   NEXT :
   PRINT :
   INPUT "ENTER FIRST NAME";I$:
   CLS
50 CLS :
   PRINT "THE OBJECT IS TO CATCH JAWS BEFORE HE GETS YO
   UR BAIT":
   PRINT :
   PRINT "USE THE DOWN ARROW TO PLACE LINE IN WATER":
   PRINT "AND THE UP ARROW TO LIFT IT OUT":
   PRINT :
   PRINT "REMEMBER, IF YOU PULL YOUR LINE OUT OF THE WA
   TER WITHOUT JAWS ON ";
60 PRINT "THE END, THEN HE SCORES A POINT"
70 PRINT :
   INPUT "TO PLAY HIT ENTER";I1$:
   CLS
75 REM  =====
      JAWS MOVING RIGHT
      =====
80 F1$ = CHR$(128) + CHR$(139) + CHR$(180) + CHR$(1
   84) + STRING$(2,191) + CHR$(183) + CHR$(180)
90 F2$ = CHR$(128) + CHR$(184) + CHR$(135) + CHR$(1
   39) + STRING$(3,191) + CHR$(135)
100 F3$ = CHR$(128) + CHR$(138) + CHR$(181) + CHR$(1
   84) + STRING$(2,191) + CHR$(183) + CHR$(180)
110 F4$ = CHR$(128) + CHR$(168) + CHR$(151) + CHR$(1
   39) + STRING$(3,191) + CHR$(135)
115 REM  =====
      JAWS MOVING LEFT
      =====
120 F5$ = CHR$(128) + CHR$(184) + CHR$(187) + STRING$
   (2,191) + CHR$(180) + CHR$(184) + CHR$(135) + CHR$
   (128)
130 F6$ = CHR$(128) + CHR$(139) + STRING$(3,191) + CHR$
   (135) + CHR$(139) + CHR$(180) + CHR$(128)
140 F7$ = CHR$(128) + CHR$(184) + CHR$(187) + STRING$
   (2,191) + CHR$(180) + CHR$(186) + CHR$(133) + CHR$
   (128):

```

```

F8$ = CHR$ (128) + CHR$ (139) + STRING$ (3,191) + CHR$
(135) + CHR$ (171) + CHR$ (148) + CHR$ (128)
145 REM =====
      MOUTH OPEN MOVING RIGHT
      =====
150 O1$ = CHR$ (128) + CHR$ (139) + CHR$ (180) + CHR$ (1
84) + STRING$ (2,191) + CHR$ (183) + CHR$ (180) + CHR$
(128):
      O2$ = CHR$ (128) + CHR$ (184) + CHR$ (135) + CHR$ (1
39) + STRING$ (2,191) + CHR$ (176) + CHR$ (132) + CHR$
(128)
155 REM =====
      MOUTH OPEN MOVING LEFT
      =====
160 O3$ = CHR$ (128) + CHR$ (184) + CHR$ (187) + STRING$
(2,191) + CHR$ (180) + CHR$ (184) + CHR$ (135) + CHR$
(128):
      O4$ = CHR$ (128) + CHR$ (136) + CHR$ (176) + STRING$
(2,191) + CHR$ (135) + CHR$ (139) + CHR$ (180) + CHR$
(128)
165 REM =====
      JAWS HANGING BY HOOK
      =====
170 W1$ = CHR$ (160) + CHR$ (172) + CHR$ (191) + CHR$ (1
88) + CHR$ (144)
180 W2$ = CHR$ (138) + STRING$ (3,191) + CHR$ (133)
190 W3$ = CHR$ (160) + CHR$ (158) + CHR$ (143) + CHR$ (1
73) + CHR$ (144)
195 REM =====
      GRAPHICS FOR PIER
      =====
200 PRINT CHR$ (191) + STRING$ (62,131) + CHR$ (191);:
      FOR X = 1 TO 14:
          PRINT CHR$ (191) + STRING$ (62,128) + CHR$ (191
);:
      NEXT :
      PRINT CHR$ (191) + STRING$ (62,176);:
      POKE 16383,191
210 P1$ = STRING$ (2,143) + CHR$ (191) + STRING$ (12,143
) + CHR$ (191) + STRING$ (2,143):
      P2$ = CHR$ (191) + STRING$ (12,128) + CHR$ (191):
      M1$ = CHR$ (139) + CHR$ (191):
      M2$ = CHR$ (179) + CHR$ (191):
      M3$ = CHR$ (149)
220 PRINT @558,P1$;:
      PRINT @624,P2$;:
      PRINT @429,M1$;:
      PRINT @493,M2$;:
      PRINT @557,M3$;:
      PRINT @641, STRING$ (62,"-");:
      POKE 15935,191

```

Programming Tricks

```

230 A = 769:
    GOSUB 440
235 REM =====
        PRINT SCORE KEEPER
        =====
240 PRINT @65,"JAWS";:
    PRINT @72,I$;:
    FOR M = 3 TO 23:
        SET (12,M):
        SET (13,M):
    NEXT
250 A = 769:
    T1$ = F1$:
    T2$ = F2$:
    T3$ = F3$:
    T4$ = F4$:
    T5$ = F5$:
    T6$ = F6$:
    T7$ = F7$:
    T8$ = F8$
255 REM =====
        CONTROL FOR JAWS SWIMMING
        =====
260 PRINT @A,T1$;:
    PRINT @A + 64,T2$;:
    PRINT @A,T3$;:
    PRINT @A + 64,T4$;
270 A = A + 1:
    IF A = 823
    THEN 350 .....
280 R = RND (10):
    IF R = 1
    THEN T2$ = O2$:
        T4$ = O2$
290 IF R = 10
    THEN T2$ = F2$:
        T4$ = F4$
300 R = RND (50):
    IF R = 25
    THEN 360 .....
310 IF PEEK (14400) = 8 GOSUB 440
315 REM =====
        IF JAWS MOUTH OPEN MAKE SOUNDS
        =====
320 IF T2$ = O2$ FOR X = 1 TO 4:
        OUT 255,0:
        OUT 255,2:
        NEXT
330 IF PEEK (14400) = 16 GOSUB 500
340 GOTO 260 .....

```

```

345     REM =====
          RE-DEFINE STRING FOR DIRECTION
          =====
350     A = A - 1:
          IF A = 769
360     THEN 250 .....
          PRINT @A,T5$;:
          PRINT @A + 64,T6$;:
          PRINT @A,T7$;:
          PRINT @A + 64,T8$;
370     R = RND (10):
          IF R = 1
          THEN T6$ = O4$:
              T8$ = O4$
380     IF R = 10
          THEN T6$ = F6$:
              T8$ = F8$
390     R = RND (50):
          IF R = 25
          THEN 260 .....
395     REM =====
          CHECK IF PLAYER MOVES POLE
          =====
400     IF PEEK (14400) = 8 GOSUB 440
410     IF PEEK (14400) = 16 GOSUB 500
420     IF T6$ = O4$ FOR X = 1 TO 4:
          OUT 255,0:
          OUT 255,2:
          NEXT
430     GOTO 350 .....
-----
435     REM =====
          GRAPHICS FOR BOY FISHING
          =====
440     M = 165:
          PRINT @M + 320," " ";:
          PRINT @M + 384," " ";:
          PRINT @M + 448," " ";:
          PRINT @M + 512," " ";:
          PRINT @M + 576," " ";
450     M = 165:
          PRINT @M, CHR$ (157) + CHR$ (144);:
          PRINT @M + 64, CHR$ (149) + CHR$ (130) + CHR$
            (164);:
          PRINT @M + 128, CHR$ (149) + STRING$ (2,12
            8) + CHR$ (137) + CHR$ (144);:
          PRINT @M + 192, CHR$ (149) + STRING$ (3,12
            8) + CHR$ (130) + CHR$ (164);
460     PRINT @M + 256, CHR$ (149) + STRING$ (5,12
            8) + CHR$ (137) + CHR$ (144);
470     PRINT @429, CHR$ (167) + CHR$ (183) + CHR$

```

Programming Tricks

```

(133);
480 PRINT @A, " ";:
PRINT @A + 64, " ";:
490 RETURN .....
-----
495 REM =====
CHANGE JAWS DIRECTION
=====
500 M = 165:
PRINT @M, " ";:
PRINT @M + 64, " ";:
PRINT @M + 128, " ";:
PRINT @M + 192, " ";:
PRINT @M + 256, " ";:
PRINT @M + 320, CHR$ (151) + STRING$ (7,13
1);:
PRINT @M + 384, CHR$ (149);:
PRINT @M + 448, CHR$ (149);:
PRINT @M + 512, CHR$ (149);
510 PRINT @M + 576, CHR$ (149);:
PRINT @429, CHR$ (139) + CHR$ (191) + CHR$
(128);
520 H = 805
530 IF A < H
THEN 620 .....
540 IF A > H
THEN 560 .....
550 IF A = H
THEN 680 .....
560 PRINT @A, O3$;:
PRINT @A + 64, O4$;
570 A = A - 1
575 REM =====
CHECK IF JAWS GETS CAUGHT
=====
580 IF PEEK (14400) = 8 AND A > H GOSUB 720:
GOTO 440 .....
590 IF PEEK (14400) = 8 AND A = H
THEN 770 .....
600 IF A < H
THEN 680 .....
610 GOTO 560 .....
-----
620 H = 798:
PRINT @A, O1$;:
PRINT @A + 64, O2$;
630 A = A + 1
640 IF PEEK (14400) = 8 AND A < H GOSUB 720:
GOTO 440 .....

```

```

650      IF PEEK (14400) = 8 AND A = H
        THEN 770 .....
660      IF A > H
        THEN 680 .....
670      GOTO 620 .....
-----
675      REM =====
          PRINT JAWS GETTING HOOKED
          =====
680      PRINT @A, " ";
        PRINT @A + 64, " ";
690      PRINT @A - 64, W1$;:
        PRINT @A, W2$;:
        PRINT @A + 64, W3$;
700      PRINT @A - 64, " ";
        PRINT @A, " ";
710      PRINT @A + 64, " ";
720      PRINT @T, CHR$ (191);:
        T = T + 64
730      PRINT @A, " ";
        PRINT @A + 64, " ";
740      PRINT @A, CHR$ (184) + CHR$ (187) + CHR$ (
        143) + CHR$ (191) + CHR$ (180) + CHR$ (184
        ) + CHR$ (135);:
        PRINT @A + 64, CHR$ (136) + STRING$ (2,188
        ) + CHR$ (191) + CHR$ (135) + CHR$ (139) +
        CHR$ (180);
750      IF T = 515 FOR X = 1 TO 100:
          PRINT @65, "JAWS";:
          PRINT @65, " ";:
          OUT 255, 0:
          OUT 255, 2:
          NEXT :
          FOR T = 131 TO 515 STEP 64:
            PRINT @T, CHR$ (128);:
            PRINT @T + 6, CHR$ (128);:
          NEXT :
          T = 131:
          WO = 137:
          PRINT @65, "JAWS";:
760      RETURN .....
-----
770      GOSUB 440
780      M = 483
790      PRINT @A, " ";
        PRINT @A + 64, " ";
800      PRINT @M, W1$;:
        PRINT @M + 64, W2$;:
        PRINT @M + 128, W3$;
810      PRINT @WO, CHR$ (191);:
        WO = WO + 64

```

## Programming Tricks

```
820      IF PEEK (14400) = 16
      THEN PRINT @M, " ";:
          PRINT @M + 64, " ";:
          PRINT @M + 128, " ";:
          RETURN .....
825      REM =====
      ===
          END OF GAME
          =====
      =====
830      IF WO = 521 FOR X = 1 TO 100:
          PRINT @72, I$;:
          PRINT @72, " ";:
          OUT 255, 0:
          OUT 255, 2:
          NEXT :
          FOR WO = 137 TO 521 STEP
            64:
              PRINT @WO, CHR$ (
                128);:
              PRINT @WO - 6, CHR$
                (128);:
              NEXT :
              T = 131:
              WO = 137:
              PRINT @72, I$;
840      GOTO 820 .....
```

---

---

---

# 4

---

## Methods of Display Using PRINT @

The PRINT @ command allows us to print words and graphics on the screen at any of 1023 different positions. For most of our work with graphics and animation, we will not be using the last position (1023), but this position does come in handy when we want to scroll the screen up for different graphic effects.

The following program is called ROBOT BUILDER and is a good example of using the PRINT @ command.

### ROBOT BUILDER

This is one of my favorite programs because it lets the user be creative, and it gives the user most of the control over the outcome of the program.

This is great for children who need to feel creative but do not have the programming experience to draw a robot on the screen. By using multiple-choice questions we can give them this ability.

These same ideas can be used for a great many educational games or learning-aids. Another good use for computer instruction is in the hand-eye coordination field for children who have learning disabilities. We have used this for our son, and it has helped to improve his control and hand-eye reflexes to a great degree.

The program uses a very small amount of string\$ to create a very large number of pictures, — about 71 possible combinations in all! This gives the program the staying power to keep a child interested for quite some time.

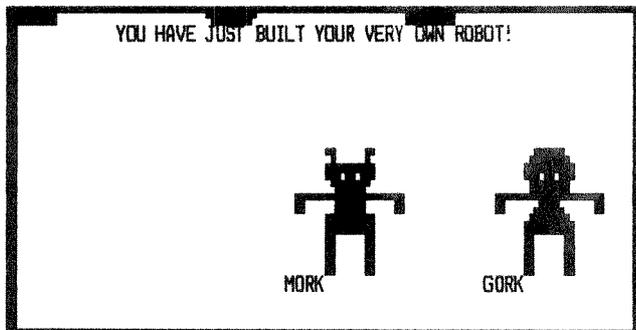
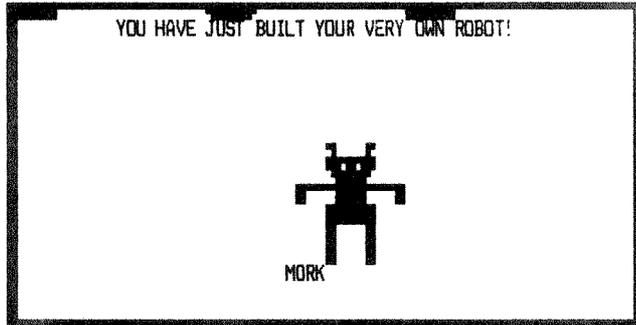


Figure 4.1 Robot Builder Program

```

10 CLS :
   CLEAR 1000
20 PRINT CHR$ (23):
   PRINT @448,"THE ROBOT BUILDER GAME"
30 PRINT :
   PRINT "NAME OF ROBOT #1";:
   INPUT P1$
40 PRINT "NAME OF ROBOT #2";:
   INPUT P2$
50 CLS
60 R = 0
70 A = 1:
   GOSUB 140:
   PRINT @0,TH$;
80 A = 2:
   GOSUB 140:
   PRINT @20,TH$;
90 A = 3:
   GOSUB 140:
   PRINT @40,TH$
100 PRINT "1                2                3";
110 PRINT :
   INPUT "CHOOSE THE TOP OF THE HEAD YOU WANT";A:
   GOSUB 140
120 IF R = 0 PRINT @480,TH$:
   GOTO 180 .....
130 IF R = 1 PRINT @500,TH$;:
   GOTO 180 .....
140 IF A = 1
   THEN TH$ = CHR$ (187) + STRING$ (3,176) + CHR$ (183)

150 IF A = 2
   THEN TH$ = STRING$ (5,191)
160 IF A = 3
   THEN TH$ = CHR$ (184) + STRING$ (3,191) + CHR$ (180)

170 RETURN
-----
180 A = 1:
   GOSUB 250:
   PRINT @0,BH$
190 A = 2:
   GOSUB 250:
   PRINT @20,BH$
200 A = 3:
   GOSUB 250:
   PRINT @40,BH$
210 PRINT "1                2                3";
220 PRINT :
   INPUT "NOW THE BOTTOM OF HEAD";A:

```

## Robot Builder

```

GOSUB 250
230 IF R = 0 PRINT @544,BH$:
      GOTO 290 .....
240 IF R = 1 PRINT @564,BH$;:
      GOTO 290 .....
250 IF A = 1
      THEN BH$ = CHR$ (139) + CHR$ (189) + CHR$ (191) + CHR$
              (190) + CHR$ (135)
260 IF A = 2
      THEN BH$ = CHR$ (191) + CHR$ (189) + CHR$ (191) + CHR$
              (190) + CHR$ (191)
270 IF A = 3
      THEN BH$ = CHR$ (171) + CHR$ (189) + CHR$ (191) + CHR$
              (190) + CHR$ (151)
280 RETURN
-----
290 A = 1:
      GOSUB 370:
      PRINT @0,TB$;
300 A = 2:
      GOSUB 370:
      PRINT @20,TB$;
310 A = 3:
      GOSUB 370:
      PRINT @40,TB$
320 PRINT "1                2                3";
330 PRINT :
      INPUT "TOP OF BODY";A:
      GOSUB 370
340 LA$ = CHR$ (191) + STRING$ (2,131):
      RA$ = STRING$ (2,131) + CHR$ (191)
350 IF R = 0 PRINT @605,LA$ + TB$ + RA$;:
      GOTO 410 .....
360 IF R = 1 PRINT @625,LA$ + TB$ + RA$;:
      GOTO 410 .....
370 IF A = 1
      THEN TB$ = CHR$ (131) + STRING$ (3,191) + CHR$ (131)
380 IF A = 2
      THEN TB$ = CHR$ (187) + STRING$ (3,191) + CHR$ (183)
390 IF A = 3
      THEN TB$ = CHR$ (131) + CHR$ (187) + CHR$ (191) + CHR$
              (183) + CHR$ (131)
400 RETURN
-----
410 A = 1:
      GOSUB 480:
      PRINT @0,BB$
420 A = 2:
      GOSUB 480:

```

```

PRINT @20, BB$
430 A = 3:
GOSUB 480:
PRINT @40, BB$
440 PRINT "1                2                3";
450 PRINT :
INPUT "BOTTOM OF BODY";A:
GOSUB 480
460 IF R = 0 PRINT @672, BB$:
GOTO 520 .....
470 IF R = 1 PRINT @692, BB$;:
GOTO 520 .....
480 IF A = 1
THEN BB$ = STRING$ (5,191)
490 IF A = 2
THEN BB$ = CHR$ (174) + STRING$ (3,191) + CHR$ (157)

500 IF A = 3
THEN BB$ = CHR$ (190) + STRING$ (3,191) + CHR$ (189)

510 RETURN

```

---

```

520 L$ = CHR$ (191) + STRING$ (3,128) + CHR$ (191)
530 IF R = 0 PRINT @736, L$:
PRINT @800, L$
540 IF R = 1 PRINT @756, L$;:
PRINT @820, L$;
550 PRINT @0, " YOU HAVE JUST BUILT YOUR VERY OWN ROBOT!"
560 PRINT :
PRINT " "
570 IF R = 0 PRINT @860, P1$;
580 IF R = 1 PRINT @880, P2$;
590 FOR X = 1 TO 5
600 IF R = 1 SET (107,24):
SET (110,24)
610 SET (67,24):
SET (70,24)
620 FOR Z = 1 TO 500:
NEXT
630 RESET (67,24):
RESET (70,24)
640 IF R = 1 RESET (107,24):
RESET (110,24)
650 FOR Z = 1 TO 500:
NEXT
660 NEXT X
670 PRINT @0, "
"
680 R = R + 1:
IF R = 2

```

## Operating Commands for Newsletter

```
THEN 690: .....  
ELSE 70 .....  
-----  
690 GOTO 690  
-----
```

---

### NEWS LETTER Program

The NEWS LETTER program lets you print your own little paper. This program does require a printer. The program enables you to print bold face headings about 3/4 inches high and provides a choice of four different sizes of letters. To jump from large print to regular size, press the (@) key. After running the program, the screen goes blank.

You begin in the large print mode. The spacing of the letters allows room for 13 letters on each line and then spaces down to the next line. Hitting the up arrow key sends the screen contents to the line printer.

One unique idea in this program is the use of variables for more than one purpose. I also try to make double use of control keys for a type of flip-flop control (or back and forth movement) in my programs whenever possible. This makes it easy for the operator to remember, with less chance of hitting the wrong key.

At line 2000 there is a subroutine which copies the screen and send it to your printer. The subroutine scans each screen location, and if it finds a null character, it prints a blank; otherwise, it prints what ever it finds. I like this routine because it dumps the screen in the same size on my printer as it was on the screen.

### Operating Commands for NEWSLETTER

UP ARROW	=	SEND SCREEN TO PRINTER
@	=	REG. & LARGE PRINT
UP ARROW1	=	SMALL PRINT
UP ARROW2	=	REG. PRINT
UP ARROW3	=	LARGE PRINT
L6	=	6 LINES PER INCH
L8	=	8 LINES PER INCH
C80	=	LINE LENGH
C64	=	LINE LENGH
T5	=	TAB(5) SPACES
;	=	PRINT CHR\$(23)
TWO UP ARROWS	=	LINE FEED

Figure 4.2 Newsletter Program

```

1 CLEAR 1000
2 P = 0
10 CLS
50 A3$ = CHR$ (131) + CHR$ (191)
52 A4$ = CHR$ (140) + CHR$ (143) + CHR$ (140)
54 B3$ = STRING$ (2,179) + CHR$ (191)
56 B4$ = CHR$ (143) + STRING$ (2,140)
58 C3$ = STRING$ (2,179) + CHR$ (191)
60 C4$ = STRING$ (2,140) + CHR$ (143)
62 D3$ = CHR$ (191) + CHR$ (176) + CHR$ (191)
64 D4$ = STRING$ (2,128) + CHR$ (143)
66 E3$ = CHR$ (191) + STRING$ (2,179)
68 E4$ = STRING$ (2,140) + CHR$ (143)
70 F3$ = CHR$ (191) + STRING$ (2,179)
72 F4$ = CHR$ (143) + CHR$ (140) + CHR$ (143)
74 G3$ = STRING$ (2,131) + CHR$ (191)
76 G4$ = STRING$ (2,128) + CHR$ (143)
78 H3$ = CHR$ (191) + CHR$ (179) + CHR$ (191)
80 H4$ = CHR$ (143) + CHR$ (140) + CHR$ (143)
82 I3$ = CHR$ (191) + CHR$ (179) + CHR$ (191)
84 I4$ = STRING$ (2,128) + CHR$ (143)
86 J3$ = CHR$ (191) + CHR$ (131) + CHR$ (191)
88 J4$ = CHR$ (143) + CHR$ (140) + CHR$ (143)
90 K3$ = CHR$ (128) + CHR$ (140)
92 K4$ = CHR$ (128) + CHR$ (131)
94 L3$ = STRING$ (3,176)
96 M3$ = CHR$ (140)
97 N3$ = CHR$ (172)
98 O3$ = CHR$ (128) + CHR$ (176) + CHR$ (140)
99 O4$ = CHR$ (131)
100 A1$ = CHR$ (191) + CHR$ (179) + CHR$ (191)
110 A2$ = CHR$ (143) + CHR$ (128) + CHR$ (143)
120 B1$ = CHR$ (191) + CHR$ (179) + CHR$ (159)
130 B2$ = CHR$ (143) + CHR$ (140) + CHR$ (143)
140 C1$ = CHR$ (191) + STRING$ (2,131)
150 C2$ = CHR$ (143) + STRING$ (2,140)
160 D1$ = CHR$ (191) + CHR$ (131) + CHR$ (189)
170 D2$ = CHR$ (143) + CHR$ (140) + CHR$ (131)
180 E1$ = CHR$ (191) + CHR$ (179) + CHR$ (131)
190 E2$ = CHR$ (143) + STRING$ (2,140)
200 F1$ = CHR$ (191) + CHR$ (179) + CHR$ (131)
210 F2$ = CHR$ (143)
220 G1$ = CHR$ (191) + CHR$ (163) + CHR$ (179)
230 G2$ = CHR$ (143) + CHR$ (140) + CHR$ (143)
240 H1$ = CHR$ (191) + CHR$ (176) + CHR$ (191)
250 H2$ = CHR$ (143) + CHR$ (128) + CHR$ (143)
260 I1$ = CHR$ (131) + CHR$ (191) + CHR$ (131)
270 I2$ = CHR$ (140) + CHR$ (143) + CHR$ (140)
280 J1$ = CHR$ (131) + CHR$ (191) + CHR$ (131)
290 J2$ = CHR$ (140) + CHR$ (143)

```

## Operating Commands for Newsletter

```
300 K1$ = CHR$ (191) + CHR$ (184) + CHR$ (143)
310 K2$ = CHR$ (143) + CHR$ (130) + CHR$ (143)
320 L1$ = CHR$ (191)
330 L2$ = CHR$ (143) + STRING$ (2,140)
340 M1$ = CHR$ (159) + CHR$ (188) + CHR$ (175)
350 M2$ = CHR$ (143) + CHR$ (128) + CHR$ (143)
360 N1$ = CHR$ (191) + CHR$ (181) + CHR$ (191)
370 N2$ = CHR$ (143) + CHR$ (138) + CHR$ (143)
380 O1$ = CHR$ (191) + CHR$ (131) + CHR$ (191)
390 O2$ = CHR$ (143) + CHR$ (140) + CHR$ (143)
400 P1$ = CHR$ (191) + CHR$ (179) + CHR$ (191)
410 P2$ = CHR$ (143)
420 Q1$ = CHR$ (191) + CHR$ (131) + CHR$ (149)
430 Q2$ = CHR$ (143) + CHR$ (142) + CHR$ (141)
440 R1$ = CHR$ (191) + CHR$ (179) + CHR$ (191)
450 R2$ = CHR$ (143) + CHR$ (131) + CHR$ (141)
460 S1$ = CHR$ (191) + STRING$ (2,179)
470 S2$ = STRING$ (2,140) + CHR$ (143)
480 T1$ = CHR$ (131) + CHR$ (191) + CHR$ (131)
490 T2$ = CHR$ (128) + CHR$ (143)
500 U1$ = CHR$ (191) + CHR$ (128) + CHR$ (191)
510 U2$ = CHR$ (143) + CHR$ (140) + CHR$ (143)
520 V1$ = CHR$ (191) + CHR$ (128) + CHR$ (191)
530 V2$ = CHR$ (139) + CHR$ (140) + CHR$ (135)
540 W1$ = CHR$ (159) + CHR$ (176) + CHR$ (175)
550 W2$ = CHR$ (143) + CHR$ (131) + CHR$ (143)
560 X1$ = CHR$ (143) + CHR$ (176) + CHR$ (143)
570 X2$ = CHR$ (143) + CHR$ (128) + CHR$ (143)
580 Y1$ = CHR$ (191) + CHR$ (128) + CHR$ (191)
590 Y2$ = CHR$ (128) + CHR$ (143)
600 Z1$ = CHR$ (131) + CHR$ (179) + CHR$ (143)
610 Z2$ = CHR$ (143) + STRING$ (2,140)
690 A$ = INKEY$
692 IF A$ = " " P = P + 5:
    X = X + 1
695 IF X = 13 P = P + 63:
    X = 0
700 IF A$ = "A" PRINT @P,A1$;:
    PRINT @P + 64,A2$;:
    P = P + 5:
    X = X + 1
710 IF A$ = "B" PRINT @P,B1$;:
    PRINT @P + 64,B2$;:
    P = P + 5:
    X = X + 1
720 IF A$ = "C" PRINT @P,C1$;:
    PRINT @P + 64,C2$;:
    P = P + 5:
    X = X + 1
730 IF A$ = "D" PRINT @P,D1$;:
    PRINT @P + 64,D2$;:
```

## Operating Commands for Newsletter

```
P = P + 5:
X = X + 1
740 IF A$ = "E" PRINT @P,E1$;;
PRINT @P + 64,E2$;;
P = P + 5:
X = X + 1
750 IF A$ = "F" PRINT @P,F1$;;
PRINT @P + 64,F2$;;
P = P + 5:
X = X + 1
760 IF A$ = "G" PRINT @P,G1$;;
PRINT @P + 64,G2$;;
P = P + 5:
X = X + 1
770 IF A$ = "H" PRINT @P,H1$;;
PRINT @P + 64,H2$;;
P = P + 5:
X = X + 1
780 IF A$ = "I" PRINT @P,I1$;;
PRINT @P + 64,I2$;;
P = P + 5:
X = X + 1
790 IF A$ = "J" PRINT @P,J1$;;
PRINT @P + 64,J2$;;
P = P + 5:
X = X + 1
800 IF A$ = "K" PRINT @P,K1$;;
PRINT @P + 64,K2$;;
P = P + 5:
X = X + 1
810 IF A$ = "L" PRINT @P,L1$;;
PRINT @P + 64,L2$;;
P = P + 5:
X = X + 1
820 IF A$ = "M" PRINT @P,M1$;;
PRINT @P + 64,M2$;;
P = P + 5:
X = X + 1
830 IF A$ = "N" PRINT @P,N1$;;
PRINT @P + 64,N2$;;
P = P + 5:
X = X + 1
840 IF A$ = "O" PRINT @P,O1$;;
PRINT @P + 64,O2$;;
P = P + 5:
X = X + 1
850 IF A$ = "P" PRINT @P,P1$;;
PRINT @P + 64,P2$;;
P = P + 5:
X = X + 1
860 IF A$ = "Q" PRINT @P,Q1$;;
```

Operating Commands for Newsletter

```

        PRINT @P + 64,Q2$;:
        P = P + 5:
        X = X + 1
870 IF  A$ = "R" PRINT @P,R1$;:
        PRINT @P + 64,R2$;:
        P = P + 5:
        X = X + 1
880 IF  A$ = "S" PRINT @P,S1$;:
        PRINT @P + 64,S2$;:
        P = P + 5:
        X = X + 1
890 IF  A$ = "T" PRINT @P,T1$;:
        PRINT @P + 64,T2$;:
        P = P + 5:
        X = X + 1
900 IF  A$ = "U" PRINT @P,U1$;:
        PRINT @P + 64,U2$;:
        P = P + 5:
        X = X + 1
910 IF  A$ = "V" PRINT @P,V1$;:
        PRINT @P + 64,V2$;:
        P = P + 5:
        X = X + 1
920 IF  A$ = "W" PRINT @P,W1$;:
        PRINT @P + 64,W2$;:
        P = P + 5:
        X = X + 1
930 IF  A$ = "X" PRINT @P,X1$;:
        PRINT @P + 64,X2$;:
        P = P + 5:
        X = X + 1
940 IF  A$ = "Y" PRINT @P,Y1$;:
        PRINT @P + 64,Y2$;:
        P = P + 5:
        X = X + 1
950 IF  A$ = "Z" PRINT @P,Z1$;:
        PRINT @P + 64,Z2$;:
        P = P + 5:
        X = X + 1
960 IF  A$ = "[" GOTO 2000 .....
970 IF  A$ = "@"
      THEN 3000 .....
980 IF  A$ = ";" PRINT CHR$ (23)
1000 IF  A$ = "1" PRINT @P,A3$;:
        PRINT @P + 64,A4$;:
        P = P + 5:
        X = X + 1
1010 IF  A$ = "2" PRINT @P,B3$;:
        PRINT @P + 64,B4$;:
        P = P + 5:
        X = X + 1

```

## Operating Commands for Newsletter

```
1020 IF  A$ = "3" PRINT @P,C3$;;  
        PRINT @P + 64,C4$;;  
        P = P + 5:  
        X = X + 1  
1030 IF  A$ = "4" PRINT @P,D3$;;  
        PRINT @P + 64,D4$;;  
        P = P + 5:  
        X = X + 1  
1040 IF  A$ = "5" PRINT @P,E3$;;  
        PRINT @P + 64,E4$;;  
        P = P + 5:  
        X = X + 1  
1050 IF  A$ = "6" PRINT @P,F3$;;  
        PRINT @P + 64,F4$;;  
        P = P + 5:  
        X = X + 1  
1060 IF  A$ = "7" PRINT @P,G3$;;  
        PRINT @P + 64,G4$;;  
        P = P + 5:  
        X = X + 1  
1070 IF  A$ = "8" PRINT @P,H3$;;  
        PRINT @P + 64,H4$;;  
        P = P + 5:  
        X = X + 1  
1080 IF  A$ = "9" PRINT @P,I3$;;  
        PRINT @P + 64,I4$;;  
        P = P + 5:  
        X = X + 1  
1090 IF  A$ = "0" PRINT @P,J3$;;  
        PRINT @P + 64,J4$;;  
        P = P + 5:  
        X = X + 1  
1100 IF  A$ = ":" PRINT @P,K3$;;  
        PRINT @P + 64,K4$;;  
        P = P + 5:  
        X = X + 1  
1110 IF  A$ = "-" PRINT @P,L3$;;  
        P = P + 5:  
        X = X + 1  
1120 IF  A$ = "." PRINT @P + 64,M3$;;  
        P = P + 5:  
        X = X + 1  
1130 IF  A$ = "," PRINT @P + 64,N3$;;  
        P = P + 5:  
        X = X + 1  
1140 IF  A$ = "/" PRINT @P,O3$;;  
        PRINT @P + 64,O4$;;  
        P = P + 5:  
        X = X + 1  
1150 GOTO 690
```

---

Operating Commands for Newsletter

```

2000 CLEAR 1000:
      LPRINT CHR$ (29):
      LPRINT CHR$ (27); CHR$ (56)
2010 FOR Y = 0 TO 47
2020     P$ = ""
2030     FOR X = 0 TO 127
2040         IF POINT (X,Y)
                THEN P$ = P$ + CHR$ (191):
                ELSE P$ = P$ + " "
2050     NEXT X
2060     LPRINT P$
2070 NEXT Y
2080 GOTO 690
-----
3000 INPUT A$
3010 IF A$ = "@"
      THEN 690 .....
3015 IF A$ = "-" PRINT CHR$ (23)
3030 IF A$ = "[1" LPRINT CHR$ (29):
      GOTO 3000 .....
3040 IF A$ = "[2" LPRINT CHR$ (30):
      GOTO 3000 .....
3050 IF A$ = "[3" LPRINT CHR$ (31):
      GOTO 3000 .....
3060 IF A$ = "[" LPRINT " ";
      GOTO 3000 .....
3070 IF A$ = "L6" LPRINT CHR$ (27); CHR$ (54):
      GOTO 3000 .....
3080 IF A$ = "L8" LPRINT CHR$ (27); CHR$ (56):
      GOTO 3000 .....
3090 IF A$ = "C80" LPRINT CHR$ (27); CHR$ (65):
      GOTO 3000 .....
3100 IF A$ = "C64" LPRINT CHR$ (27); CHR$ (66):
      GOTO 3000 .....
3110 IF A$ = "T5" INPUT A$:
      LPRINT TAB( 5)A$:
      GOTO 3000 .....
3120 LPRINT A$
3200 GOTO 3000
-----

```



### Methods of Display Using SET/RESET

The SET/RESET command can be used for slowing down certain graphics for a smoother-looking effect, as shown in the SLOT MACHINE and the SHOOTING COWBOY programs.

#### SLOT MACHINE

This game was the first I wrote after unpacking my computer and learning what RUN, LIST, END, etc. stood for.

I should take a moment to thank a young boy who helped me with a part of this program. At the time, I was lost in finding a way to randomly select the cherry, bar and apple. Then I met this young boy in our local computer store. He must have been about 12 to 15 years old. Well, I mentioned the problem I was having, and he just smiled at me, sat down with a piece of paper and pencil, and in about 30 seconds he wrote out the entire routine for selecting the random combinations. He said he was not old enough to have his own computer, so he just played with the store's machine. I never did get his name.

SLOT MACHINE starts by asking how much money you want to play with. Then it draws graphics on the screen and plays your first coin. All bets are 50 cents, and the jackpot is \$3.00, — not quite as much fun as Las Vegas, but it won't cost you as much either!

The important thing to note in this program is the animation for the arm movement. This is really an eye-trick which makes the arm appear to be turning around, while the routine is actually drawing only one line and then resetting the line and drawing another one.

One of the members of our computer club thought I used machine-language for the arm! BASIC can be fast, — it's just a matter of using it to best advantage to get the effects you want.

Figure 5.1 Slot Machine Program

```
10 CLS
20 PRINT CHR$(23)
30 PRINT @460, "SLOT MACHINE":
   FOR X = 1 TO 1000:
   NEXT
40 PRINT @452, "WRITTEN BY TOM DEMPSEY":
   FOR X = 1 TO 1000:
   NEXT :
   CLS
50 PRINT "THIS IS A 50 CENT SLOT MACHINE/WITH SOUND."
60 PRINT "JACKPOT IS $3.00 FOR 3 OF A KIND."
70 PRINT "HOW MANY 50 CENT PIECES DO YOU WANT TO START
   WITH";
80 INPUT M
90 LET M = M * .5
100 PRINT "YOU ARE STARTING WITH $";M
110 FOR X = 1 TO 1000
120 NEXT X:
   CLS
130 PRINT @273, " ";
140 PRINT @272, " ";
150 LET C = 0
160 LET L = 0
170 LET O = 0
180 FOR R = 90 TO 95:
   SET (R,20):
   NEXT
190 FOR R = 30 TO 90:
   SET (R,10):
   NEXT
200 FOR R = 30 TO 90:
   SET (R,17):
   NEXT :
   FOR R = 30 TO 90:
   SET (R,41):
   NEXT
210 FOR R = 10 TO 41:
   SET (30,R):
   NEXT :
   FOR R = 10 TO 41:
   SET (90,R):
   NEXT
220 FOR R = 30 TO 90:
   SET (R,35):
   NEXT :
   FOR R = 10 TO 20:
   SET (95,R):
   NEXT
230 FOR R = 10 TO 20:
   RESET (95,R):
```

```

NEXT :
FOR R = 20 TO 40:
  SET (95,R):
NEXT
240 FOR R = 40 TO 20 STEP - 1:
  RESET (95,R):
NEXT
250 FOR R = 20 TO 10 STEP - 1:
  SET (95,R):
NEXT
260 LET B = 0:
  GOTO 290
-----
270 LET B = 1:
  GOTO 290
-----
280 LET B = 2:
  GOTO 290
-----
290 N = RND (3)
300 IF N = 1
  THEN 330 .....
310 IF N = 2
  THEN 380 .....
320 IF N = 3
  THEN 430 .....
330 FOR X = 1 TO 500
340 NEXT X
350 PRINT " <LEMON> ";;
  FOR X = 1 TO 10:
    OUT 255,1:
    OUT 255,2:
  NEXT
360 LET C = C + 1
370 GOTO 480
-----
380 FOR X = 1 TO 500
390 NEXT X
400 PRINT " <APPLE> ";;
  FOR X = 1 TO 10:
    OUT 255,1:
    OUT 255,2:
  NEXT
410 LET L = L + 1
420 GOTO 480
-----
430 FOR X = 1 TO 500
440 NEXT X
450 PRINT " <*****> ";;
  FOR X = 1 TO 10:
    OUT 255,1:

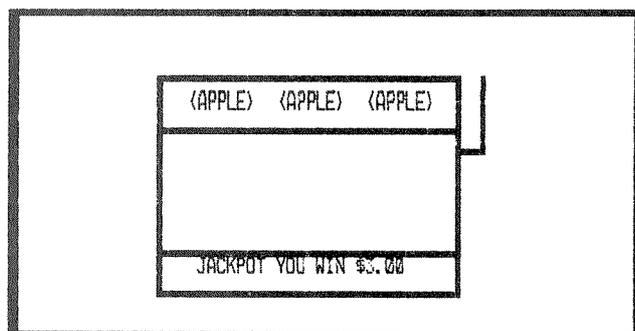
```

Slot Machine

```

                OUT 255,2:
            NEXT
460 LET O = O + 1
470 GOTO 480
-----
480 IF B = 0
    THEN 270 .....
490 IF B = 1
    THEN 280 .....
500 IF C = 3
    THEN 560 .....
510 IF L = 3
    THEN 560 .....
520 IF O = 3
    THEN 560 .....
530 LET M = M - .5
540 IF M = 0
    THEN 630 .....
550 GOTO 600
-----
560 FOR X = 1 TO 100:
    OUT 255,1:
    OUT 255,2:
    PRINT @787,"JACKPOT YOU WIN $3.00";:
    NEXT X
570 FOR X = 1 TO 1000:
    NEXT
580 PRINT @787,"                ";
590 LET M = M + 3
600 PRINT @0,"YOU HAVE $";M;" TO PLAY AGAIN HIT ENTER";
610 INPUT A$:
    PRINT @15,"                ";
620 GOTO 130
-----
630 PRINT @0,"YOU HAVE LOST ALL YOUR MONEY....PLAY AGAIN
    SOON"

```



### Program Explanation

- Lines 10 to 70 print the rules.
- Lines 80 to 100 change the coins to dollars.
- Lines 110 to 250 move the arm.
- Lines 260 to 280 keep the program from repeating.
- Lines 300 to 470 generate random selection and sound.
- Lines 480 to 520 provide program control.
- Lines 560 to 570 print jackpot.
- Lines 580 to 630 keep track of wins and end game.

### SHOOTING COWBOY

The following program is called SHOOTING COWBOY. It is another example of using SET and RESET.

This program uses the famous FOR NEXT commands for the graphics. I originally thought they would be too slow for the animation but was surprised at how fast they are in this program. This program was originally written for a Level I TRS-80 and later changed to Level II.

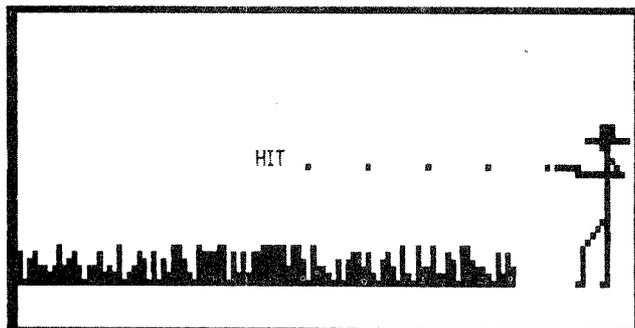
After entering RUN, the program starts out by drawing a town, — or is it a city? I guess the difference depends on the population. Then the cowboy is placed on the screen, and the targets start falling.

You shoot at the targets by hitting the up-arrow, — please don't hit it *too* hard. You get 50 shots. The player to hit the most targets with 50 shots wins the game.

In this program, we use the random number to draw our picture of the town. This saves a lot of programming, and it can also be used for POKEing the graphics on the screen for even greater speed.

One thing you should note in line 360 is that we check for the shot being fired right after SETting the target. This is done so that the target will be shown at the same time as the shot.

The animation is no more than we need to give the effects we want. Keep it simple but effective. The lower part of the arm from the elbow down is all we have to move to fire the gun and get the effects we need. The SET, RESET and FOR NEXT statements are plenty fast for this amount of animation.



## Shooting Cowboy

Figure 5.2 *Shooting Cowboy Program*

```
100 CLS :
    PRINT "TEST YOUR QUICK DRAW : SEE HOW MANY TARGETS Y
    OU"
110 PRINT "CAN HIT WITH 50 SHOTS. TO SHOOT, HIT THE ([)
    ARROW"
120 FOR X = 1 TO 3000:
    NEXT X:
    CLS
130 LET T = 0:
    LET S = 0
140 IF (M = 60) + (L = 23) LET T = T + 1:
    PRINT @0,"SCORE=";T:
    FOR X = 1 TO 500:
    NEXT X
150 A = RND (40):
    IF A < 35
    THEN 150 .....
160 FOR M = A TO 40:
    SET (B,M):
    NEXT
170 A = RND (40):
    IF A < 35
    THEN 170 .....
180 IF B > 100
    THEN 190: .....
    ELSE B = B + 1:
    GOTO 160 .....
-----
190 FOR M = 116 TO 124:
    SET (M,19):
    NEXT M
200 FOR M = 119 TO 121:
    SET (M,17):
    NEXT M
210 FOR M = 119 TO 121:
    SET (M,18):
    NEXT M
220 SET (119,20)
230 FOR M = 20 TO 40:
    SET (120,M):
    NEXT M
240 FOR M = 35 TO 40:
    SET (115,M):
    NEXT M
250 SET (119,31):
    SET (118,32):
    SET (117,33):
    SET (116,34)
260 SET (114,40):
    SET (119,40)
```

```

270     SET (121,22):
        SET (122,23):
        SET (123,24)
280     SET (123,25):
        SET (123,26):
        SET (123,27):
        SET (123,28)
290     SET (122,28):
        SET (122,29):
        SET (122,30)
300     FOR M = 108 TO 60 STEP - 12:
            RESET (M,23):
        NEXT
310     FOR M = 122 TO 114 STEP - 1:
            RESET (M,24):
        NEXT
320     FOR M = 115 TO 110 STEP - 1:
            RESET (M,23):
        NEXT
330     SET (120,24)
340     REM DROP THE TARGET
350     LET L = 2
360     SET (60,L):
        Y$ = INKEY$:
        IF Y$ = "["
            THEN 410 .....
        RESET (60,L)
370     LET L = L + 3
380     IF L = 47
390     THEN 350 .....
400     GOTO 360 .....
-----
410     REM SHOTS GUN
420     FOR X = 1 TO 10:
            OUT 255,1:
            OUT 255,2:
        NEXT
430     LET S = S + 1
440     IF S = 50 PRINT @0,"END OF GAME.....YOUR SCOR
            E IS";T:
            GOTO 600 .....
450     RESET (123,25):
        RESET (123,26):
        RESET (123,27):
        RESET (123,28)
460     RESET (122,28):
        RESET (122,29):
        RESET (122,30)
470     FOR M = 122 TO 114 STEP - 1:
            SET (M,24):
        NEXT M

```

Shooting Cowboy

```

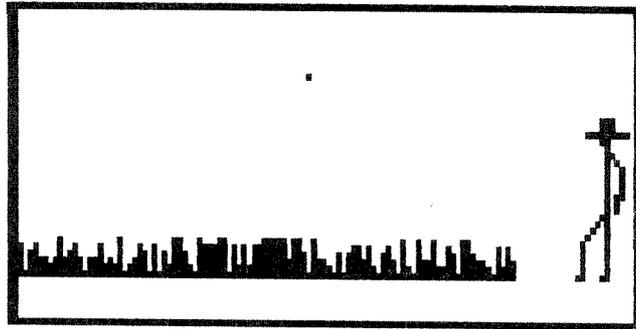
480     FOR M = 114 TO 110 STEP - 1:
        SET (M,23):
        NEXT M
490     FOR M = 108 TO 60 STEP - 12:
        SET (M,23):
        NEXT M
500     IF (L = 23) + (M = 60) PRINT @448, TAB( 25);"
        HIT";:
        FOR X = 1 TO 500:
            NEXT X:
            PRINT @448, TAB( 25);" ";:
            GOTO 140 .....
510     FOR M = 108 TO 60 STEP - 12:
        RESET (M,23):
        NEXT M
520     FOR M = 122 TO 114 STEP - 1:
        RESET (M,24):
        NEXT M
530     FOR M = 114 TO 110 STEP - 1:
        RESET (M,23):
        NEXT M
540     SET (120,24)
550     FOR M = 25 TO 28:
        SET (123,M):
        NEXT M
560     FOR M = 28 TO 30:
        SET (122,M):
        NEXT M
570     GOTO 370 .....
-----
600     C$ = CHR$ (191)
610     PRINT @384,C$;:
        FOR X = 1 TO 15:
            PRINT C$;:
        NEXT X
620     FOR X = 1 TO 500:
        NEXT X
630     PRINT @398, CHR$ (128) + CHR$ (128);
635     FOR X = 1 TO 500:
        NEXT X
640     PRINT @398,C$ + C$;
650     I$ = CHR$ (140)
660     A = 400
670     PRINT @A,I$;
675     PRINT @A, CHR$ (128);
680     A = A + 1:
        IF A > 447
        THEN 700 .....
690     GOTO 670 .....
-----
700     PRINT @379," ";

```

## Program Explanation

```
710          PRINT @1020, CHR$ (131) + CHR$ (191) + CHR$  
              (131);  
1000        GOTO @b```. . . . .
```

---



## Program Explanation

- Lines 10 to 20 print instructions.
  - Line 30 starts timer and clears screen.
  - Line 40 sets T and S at 0.
  - Line 50 checks for hit; changes score.
  - Lines 60 to 90 draw the town.
  - Lines 100 to 240 draw the cowboy.
  - Lines 250 to 310 drop the target.
  - Lines 320 to 480 sound and animation
  - Lines 490 to 630 surprise ending.
-

# NOTES

---

# 6

---

## Methods of Display Using FOR NEXT

The best method of speeding up our SET and RESET commands is the FOR NEXT loop. The following programs, called TANK CHASE, TWITS, and HORSE RACE are very fast, — especially for using only SET/RESET.

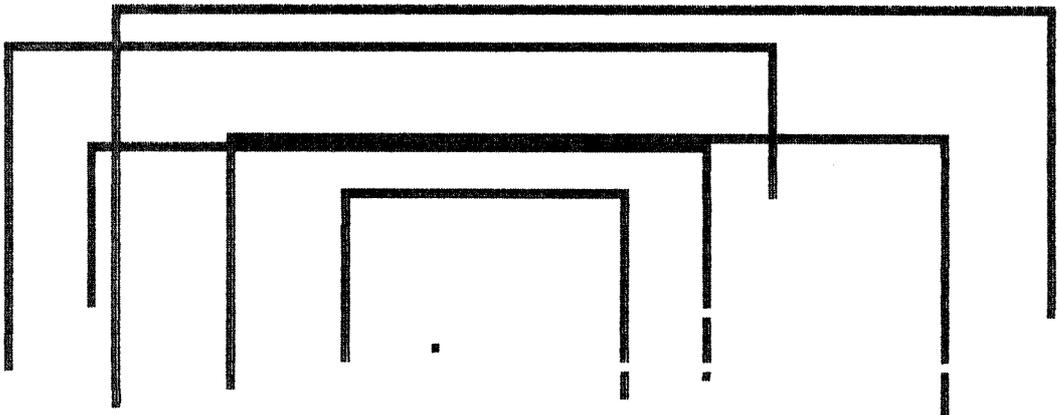
### TANK CHASE

TANK CHASE is one of the harder games to play. It's a game that starts out easy, but the longer you play, the harder it gets! This makes for a very challenging game. Once you start playing, you can't stop.

The object is to hit a target on the screen with your tank, which never stops moving. Each time you miss the target it becomes harder to hit. Play the game and you'll see what I mean. The up-arrow key controls all your turns, and even when you can no longer see the target, the computer knows where it is, — so keep trying!

One interesting change would be to let another player control a moving target.

SCORE= 0



## Tank Chase

Figure 6.1 *Tank Chase Program*

```
10 T = 0:
   P = 0
20 CLS
30 PRINT "THIS GAME IS CALLED TANK CHASE....TO TURN YOU
   R TANK"
40 PRINT "HIT THE ([]) ARROW. EACH TIME YOU HIT THE TARG
   ET YOU"
50 PRINT "GET 5 POINTS. YOU CAN ONLY HIT THE TARGET GOI
   NG IN"
60 PRINT "THIS ---> DIRECTION. EACH TIME YOU MISS THE T
   ARGET"
70 PRINT "THE GAME GETS HARDER TO PLAY AS YOU WILL SOON
   SEE!!"
80 PRINT :
   PRINT "WHEN READY HIT ENTER";:
   INPUT A$
90 CLS
100 PRINT @5,"SCORE=";T;:
   PRINT @0," ";
110 A$ = "["
120 C = RND (127):
   D = RND (47)
130 IF D < 10 GOTO 120 .....
140 SET (C,D)
150 A = 63:
   B = 23
160 RESET (A,B)
170 LET A = A + 1
180 IF INKEY$ = "["
   THEN 230 .....
190 IF POINT (C,D) = 0 LET T = T + 5:
   GOTO 90 .....
200 SET (A,B)
210 IF A = 127 GOTO 230 .....
220 GOTO 160

-----
230 SET (A,B)
240 IF B = 3 GOTO 280 .....
250 LET B = B - 1
260 IF INKEY$ = "["
   THEN 280 .....
270 GOTO 230

-----
280 SET (A,B)
290 IF A = 0 GOTO 330 .....
300 LET A = A - 1
310 IF INKEY$ = "["
   THEN 330 .....
320 GOTO 280

-----
```

```

330 SET (A,B)
340 IF B = 47 GOTO 380 .....
350 LET B = B + 1
360 IF INKEY$ = "["
    THEN 380 .....
370 GOTO 330
-----
380 LET P = P + 1
390 IF P = 20 GOTO 410 .....
400 GOTO 160
-----
410 PRINT @409,"END OF GAME":
    END
-----

```

**Program Explanation**

Lines 10 to 90 print the rules.  
 Lines 110 to 170 start the tank moving and set target.  
 Lines 180 to 410 check if player turns tank.  
 Lines 380 to 390 are a counter for number of turns made.

**TWITS IN WATER**

This program was written in order to show my son a graphic example of the random number generator. I try to put some kind of graphics into every program I write.

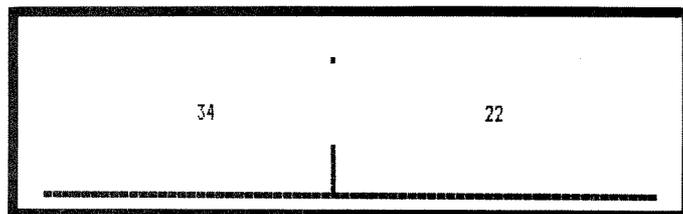
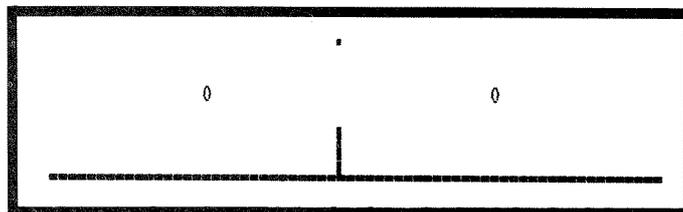


Figure 6.2 *Twits Program*

```

10 REM *** TWITS IN WATER
20 REM *** WRITTEN BY TOM DEMPSEY
30 CLS
40 PRINT "THIS IS A GRAPHIC DISPLAY OF THE COMPUTER'S R
  ANDOM"
50 PRINT "NUMBER GENERATOR."
60 PRINT "I NAMED IT TWITS IN WATER BECAUSE A TWIT IS"
70 PRINT "SOMETHING NO ONE NEEDS."
80 PRINT "PICTURE YOURSELF AS A BIG GIANT, STANDING OVE
  R A RIVER,"
90 PRINT "DROPPING TWITS INTO THE WATER. TO MAKE IT MOR
  E"
100 PRINT "INTERESTING, YOU DROP THEM ON TOP OF A WALL T
  O SEE"
110 PRINT "HOW MANY FALL ON EACH SIDE.":
  PRINT :
  PRINT "WHEN READY TO WATCH, HIT ENTER";:
  INPUT A$
120 CLS
130 FOR M = 30 TO 40:
  SET (60,M):
  NEXT
140 FOR M = 0 TO 127:
  SET (M,40):
  NEXT
150 PRINT @576, TAB( 15);B; TAB( 45);A
160 SET (60,T):
  FOR X = 1 TO 50:
  NEXT :
  RESET (60,T)
170 IF T > 30 LET T = 0:
  GOTO 190 .....
180 T = T + 4:
  GOTO 160

-----
190 N = RND (2)
200 IF N = 1
  THEN 220 .....
210 IF N = 2
  THEN 280 .....
220 SET (64,32):
  FOR X = 1 TO 50:
  NEXT :
  RESET (64,32)
230 SET (67,35):
  FOR X = 1 TO 50:
  NEXT :
  RESET (67,35)
240 SET (70,38):
  FOR X = 1 TO 50:

```

```

        NEXT :
        RESET (70,38)
250 RESET (73,40):
        SET (73,40)
260 A = A + 1
270 GOTO 150
-----
280 SET (56,32):
        FOR X = 1 TO 50:
        NEXT :
        RESET (56,32)
290 SET (53,35):
        FOR X = 1 TO 50:
        NEXT :
        RESET (53,35)
300 SET (50,38):
        FOR X = 1 TO 50:
        NEXT :
        RESET (50,38)
310 RESET (47,40):
        SET (47,40)
320 B = B + 1
330 GOTO 150
-----

```

### Program Explanation

Lines 10 to 120 are the instructions.  
 Lines 130 to 140 draw the wall and water.  
 Line 150 keeps the score.  
 Lines 160 to 330 drop the twits.

I sure hope there are no spelling errors in this program!

### HORSE RACE

This is a good game for a house full of company. We tried it one night on some friends, and all the laughter and yelling of ten people for their horse to come in first was something else! The game is played by placing your bets in a kitty and letting the winner take all. Of course, real gambling is not legal, so we use buttons, — Ha! Ha!

The game shows the horses running and also prints out the win, place and show, — so there's no mistaking who wins!

## Horse Race

Figure 6.3 *Horse Race Program*

```
10 CLS
20 FOR N = 1 TO 9
30 PRINT N
40 NEXT N
50 FOR M = 0 TO 26:
    SET (127,M):
    NEXT
60 A = 5:
    B = 5:
    C = 5:
    D = 5:
    E = 5:
    F = 5:
    G = 5:
    H = 5:
    I = 5
70 SET (A,1)
80 SET (B,4)
90 SET (C,7)
100 SET (D,10)
110 SET (E,13)
120 SET (F,16)
130 SET (G,19)
140 SET (H,22)
150 SET (I,25)
160 PRINT @640,"WELCOME TO THE RACES":
    FOR X = 1 TO 2000:
    NEXT
170 PRINT @640,"PLEASE PLACE YOUR BETS":
    FOR X = 1 TO 2000:
    NEXT
180 PRINT @640,"WHEN READY TO PLAY HIT (Y)"
190 A$ = INKEY$
200 IF A$ = "Y"
    THEN 220 .....
210 GOTO 190
-----
220 PRINT @640,"
    "
230 PRINT @640,"THE HORSES ARE AT THE POST":
    FOR X = 1 TO 2000:
    NEXT
240 PRINT @640,"*** THEY'RE OFF AND RUNNING ***"
250 R = RND (9):
    IF R > 9
    THEN 250 .....
260 IF R = 1 RESET (A,1):
    A = A + 1:
    SET (A,1)
270 IF R = 2 RESET (B,4):
```

```

B = B + 1:
SET (B,4)
280 IF R = 3 RESET (C,7):
C = C + 1:
SET (C,7)
290 IF R = 4 RESET (D,10):
D = D + 1:
SET (D,10)
300 IF R = 5 RESET (E,13):
E = E + 1:
SET (E,13)
310 IF R = 6 RESET (F,16):
F = F + 1:
SET (F,16)
320 IF R = 7 RESET (G,19):
G = G + 1:
SET (G,19)
330 IF R = 8 RESET (H,22):
H = H + 1:
SET (H,22)
340 IF R = 9 RESET (I,25):
I = I + 1:
SET (I,25)
350 IF A = 127 LET Z = Z + 1:
A = 1:
Y = A:
GOTO 450 .....
360 IF B = 127 LET Z = Z + 1:
B = 2:
Y = B:
GOTO 450 .....
370 IF C = 127 LET Z = Z + 1:
C = 3:
Y = C:
GOTO 450 .....
380 IF D = 127 LET Z = Z + 1:
D = 4:
Y = D:
GOTO 450 .....
390 IF E = 127 LET Z = Z + 1:
E = 5:
Y = E:
GOTO 450 .....
400 IF F = 127 LET Z = Z + 1:
F = 6:
Y = F:
GOTO 450 .....
410 IF G = 127 LET Z = Z + 1:
G = 7:
Y = F:
GOTO 450 .....

```



## RACETRAC

This program enables you to keep track of the jockeys' past records, the best post positions, and the average winning weight. You can compile these statistics for as long as you want. Once this information is saved, let the computer pick the winners.

You can play the computer's selections any way you wish. I tried this out last season and showed about an 80% prediction accuracy. The program picked a horse in the money about 7 out of 9 races for each day. I can only afford to place \$2.00 bets, but I still showed a profit of \$30 to \$40.00 per day! How well it works depends on how you play the computer selections.

Once the computer picked the winner of every race, but I had stayed at home that day! @\*%@%@\$%! If I had bet that day, I would have won over \$1000.00 using only \$2.00 bets!

I learned some interesting statistics last season. The winners of most races were in post positions 1, 2, 4 or 5, with number 3 horse getting squeezed out by the number 4 and 5 horses trying to get closer to the rail. Also, the best weight showed to be between 114 and 116.

There are many things that have not been taken into consideration, such as how the horse feels, how badly the jockey wants to win a race or whether he's just out to get ready for a bigger race later in the season. This program still helps make your choice a lot easier and may even help you retire sooner than you think!

## Operating Instructions

After loading the program, the first thing you must do is enter a list of jockeys' names for the computer data file. This enables you to customize the program for the area in which you live or for a special track at which you like to play.

Below is an example of how this should look on your screen.

```
A = PINCAY
B = OLIVARES
C = TORO
```

The limit is 15 jockeys. The reason you insert letters in front of each jockey is for ease of data entry later on. You must remember to use the *down arrow* after each name instead of the enter key. After the complete list is typed, press enter, and your complete list will be contained in A\$.

The program next goes to the menu and asks for your selection. At this time, I suggest that you prepare a blank tape to save data so you can be sure of having the list of jockeys on tape in case you're not going to enter any more race data at this time.

Be sure the record keys are pressed on the recorder before you enter a 5 to save data. There is a change you might like to add to the program. If you call for the computer to list all data and you have not entered any racing data, you will crash

## Program Explanation

the program and get a 0/ERROR in line 3060. To eliminate this, change line 3000 to read:

```
3000 CLS:ON ERROR GOTO 10
```

This was not done in my original program because I didn't think anyone would ask to see data if they hadn't put any in, but after making this mistake, I thought it best to add the error trap!

Another tip to help out is to make a list of the jockeys on paper. Because of space limitations, it wasn't practical to put all this on the screen for every mode. This program was written to allow changes to be made very easily. If you find a way to pick 9 out of 9 races, please feel free to tell me about it!

## Program Explanation

Lines 1 to 10 input names of jockeys for data file.  
Lines 10 to 85 set up jockey list and menu.  
Lines 90 to 160 input winner data from past races.  
Lines 200 to 340 keep track of jockeys' records.  
Lines 350 to 450 keep track of weights.  
Lines 460 to 560 post position records.  
Lines 570 to 800 store past records of horses.  
Lines 2000 to 2070 load past data.  
Lines 3000 to 3090 display data in file.  
Lines 4000 to 4145 input today's race info.  
Lines 5000 to 5140 compute jockey data.  
Lines 5150 to 5250 compute weight data  
Lines 5260 to 5360 post position data.  
Lines 5370 to 5590 past records of horses.  
Lines 6000 to 6200 print predictions.  
Lines 7000 to 7070 save all data.

```
TYPE NAMES OF JOCKEYS FOR DATA FILE - HIT DOWN ARROW AFTER EACH  
NAME - WHEN COMPLETE LIST IS DONE HIT ENTER! (MAX IS 15 NAMES)
```

```
EXAMPLE : A = PINCAY (DOWN ARROW) B = OLIVARES (DOWN ARROW)  
? ■
```

```
ENTER FILE NUMBER
```

```
1 = LOAD PAST DATA  
2 = ENTER NEW DATA  
3 = PRINT ALL DATA  
4 = MAKE SELECTION  
5 = SAVE RACE DATA
```

Figure 6.4 *Racetrac Program*

```

1 CLEAR 1000:
  R = 1
2 CLS :
  PRINT CHR$ (23)
3 PRINT @460,"RACE TRACK COMPUTER":
  FOR X = 1 TO 1000:
    NEXT :
  CLS
4 PRINT "TYPE NAMES OF JOCKEYS FOR DATA FILE - HIT DOWN
  N ARROW AFTER EACH NAME - WHEN COMPLETE LIST IS DONE
  HIT ENTER! (MAX IS 15 NAMES)"
5 PRINT :
  PRINT "EXAMPLE : A = PINCAY (DOWN ARROW) B = OLIVARES
  S (DOWN ARROW)"
6 INPUT A$:
  CLS
10 CLS :
  R = 1
20 PRINT "ENTER FILE NUMBER":
  PRINT
30 PRINT "1 = LOAD PAST DATA":
  PRINT "2 = ENTER NEW DATA":
  PRINT "3 = PRINT ALL DATA":
  PRINT "4 = MAKE SELECTION":
  PRINT "5 = SAVE RACE DATA"
40 I$ = INKEY$ :
  IF I$ = ""
  THEN 40 .....
50 IF I$ = "1"
  THEN 2000 .....
60 IF I$ = "2"
  THEN 90 .....
70 IF I$ = "3"
  THEN 3000 .....
80 IF I$ = "4"
  THEN 4000 .....
85 IF I$ = "5"
  THEN 7000 .....
90 INPUT "ENTER NUMBER OF RACES FOR TODAY'S DATA";RA:
  XX = XX + RA
100 PRINT A$
110 PRINT @30,"ENTER DATA FOR RACE # ";R;
120 PRINT @158,"WINNER";:
  WI$ = INKEY$ :
  IF WI$ = ""
  THEN 120: .....
  ELSE PRINT @166,WI$;
130 PRINT @222,"WEIGHT";:
  WE$ = INKEY$ :
  IF WE$ = ""

```

Program Explanation

```

        THEN 130: .....
        ELSE PRINT @230,WE$;
140 PRINT @286,"POST #";:
        PO$ = INKEY$ :
        IF PO$ = ""
        THEN 140: .....
        ELSE PRINT @294,PO$;
150 PRINT @350,"P/RECD";:
        PA$ = INKEY$ :
        IF PA$ = ""
        THEN 150: .....
        ELSE PRINT @358,PA$;
160 GOSUB 200:
        R = R + 1:
        IF R = RA + 1
        THEN 10: .....
        ELSE CLS :
        GOTO 100 .....

```

---

```

200 IF WI$ = "A"
    THEN A = A + 1
210 IF WI$ = "B"
    THEN B = B + 1
220 IF WI$ = "C"
    THEN C = C + 1
230 IF WI$ = "D"
    THEN D = D + 1
240 IF WI$ = "E"
    THEN E = E + 1
250 IF WI$ = "F"
    THEN F = F + 1
260 IF WI$ = "G"
    THEN G = G + 1
270 IF WI$ = "H"
    THEN H = H + 1
280 IF WI$ = "I"
    THEN I = I + 1
290 IF WI$ = "J"
    THEN J = J + 1
300 IF WI$ = "K"
    THEN K = K + 1
310 IF WI$ = "L"
    THEN L = L + 1
320 IF WI$ = "M"
    THEN M = M + 1
330 IF WI$ = "N"
    THEN N = N + 1
340 IF WI$ = "O"
    THEN O = O + 1
350 REM ***** WEIGHT
360 IF WE$ = "0"

```

```

THEN W = W + 110
370 IF WE$ = "1"
    THEN W = W + 111
380 IF WE$ = "2"
    THEN W = W + 112
390 IF WE$ = "3"
    THEN W = W + 113
400 IF WE$ = "4"
    THEN W = W + 114
410 IF WE$ = "5"
    THEN W = W + 115
420 IF WE$ = "6"
    THEN W = W + 116
430 IF WE$ = "7"
    THEN W = W + 117
440 IF WE$ = "8"
    THEN W = W + 118
450 IF WE$ = "9"
    THEN W = W + 119
460 REM ***** POST POSITIONS
470 IF PO$ = "1"
    THEN P = P + 1
480 IF PO$ = "2"
    THEN P = P + 2
490 IF PO$ = "3"
    THEN P = P + 3
500 IF PO$ = "4"
    THEN P = P + 4
510 IF PO$ = "5"
    THEN P = P + 5
520 IF PO$ = "6"
    THEN P = P + 6
530 IF PO$ = "7"
    THEN P = P + 7
540 IF PO$ = "8"
    THEN P = P + 8
550 IF PO$ = "9"
    THEN P = P + 9
560 IF PO$ = "0"
    THEN P = P + 10
570 REM ***** PAST RECORDS
580 IF PA$ = "0"
    THEN HO = 0
590 IF PA$ = "1"
    THEN HO = 1
600 IF PA$ = "2"
    THEN HO = 2
620 IF PA$ = "4"
    THEN HO = 4
622 IF PA$ = "5"
    THEN HO = 5

```

## Program Explanation

```
624 IF PA$ = "6"  
    THEN HO = 6  
626 IF PA$ = "7"  
    THEN HO = 7  
628 IF PA$ = "8"  
    THEN HO = 8  
630 IF WI$ = "A" AND HO > 0  
    THEN A1 = HO  
640 IF WI$ = "B" AND HO > 0  
    THEN B1 = HO  
650 IF WI$ = "C" AND HO > 0  
    THEN C1 = HO  
660 IF WI$ = "D" AND HO > 0  
    THEN D1 = HO  
670 IF WI$ = "E" AND HO > 0  
    THEN E1 = HO  
680 IF WI$ = "F" AND HO > 0  
    THEN F1 = HO  
690 IF WI$ = "G" AND HO > 0  
    THEN G1 = HO  
700 IF WI$ = "H" AND HO > 0  
    THEN H1 = HO  
710 IF WI$ = "I" AND HO > 0  
    THEN I1 = HO  
720 IF WI$ = "J" AND HO > 0  
    THEN J1 = HO  
730 IF WI$ = "K" AND HO > 0  
    THEN K1 = HO  
740 IF WI$ = "L" AND HO > 0  
    THEN L1 = HO  
750 IF WI$ = "M" AND HO > 0  
    THEN M1 = HO  
760 IF WI$ = "N" AND HO > 0  
    THEN N1 = HO  
770 IF WI$ = "O" AND HO > 0  
    THEN O1 = HO  
800 RETURN
```

---

```
1000 GOTO 1000
```

---

```
2000 CLS  
2010 PRINT @466, "LOADING DATA"  
2020 INPUT # - 1, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, W, XX, A$  
2030 CLS  
2040 PRINT @466, "DATA LOAD COMPLETE"  
2050 FOR X = 1 TO 1000:  
    NEXT  
2060 CLS  
2070 GOTO 20
```

---

```
3000 CLS
```

```

3010 PRINT A$;
3020 X = 20
3030 PRINT @X,A;:
      PRINT @84,B;:
      PRINT @148,C;:
      PRINT @212,D;:
      PRINT @276,E;:
      PRINT @340,F;:
      PRINT @404,G;:
      PRINT @468,H;:
      PRINT @532,I;:
      PRINT @596,J;:
      PRINT @660,K;:
      PRINT @724,L;:
      PRINT @788,M;:
      PRINT @852,N;:
      PRINT @916,O;
3040 PRINT @30,"RACES WON THIS SEASON";
3050 PRINT @94," ";
3060 PRINT @158,"AVERAGE WINNING WEIGHT =";:
      WE = INT (W / XX):
      PRINT WE;
3070 PRINT @222,"BEST POST POSITION      =";:
      PO = INT (P / XX):
      PRINT PO;
3090 PRINT @350,"HIT ENTER TO CONTINUE";:
      INPUT I$:
      GOTO 10

```

---

```

4000 PP = 1:
      T1 = 0:
      T2 = 0:
      T3 = 0:
      T4 = 0:
      T5 = 0:
      T6 = 0:
      T7 = 0:
      T8 = 0:
      T9 = 0
4010 CLS :
      INPUT "HOW MANY HORSES IN RACE (MAX IS 9)";I2:
      CLS
4030 PRINT "ENTER DATA FOR RACE # ";PP
4040 PRINT @146,"HORSE, JOCKEY CODE, WEIGHT, POST#, PAST
      RECORD";
4050 PRINT @210,"";:
      INPUT H1$,J1$,W1,P1,R1:
      GOSUB 5000
4055 T1 = T1 + T:
      I1 = I1 + 1:
      IF I1 = I2

```

Program Explanation

```
THEN 6000 .....
4060 PRINT @274,"";
      INPUT H2$,J1$,W2,P2,R2:
      GOSUB 5000
4065 T2 = T2 + T:
      I1 = I1 + 1:
      IF I1 = I2
      THEN 6000 .....
4070 PRINT @338,"";
      INPUT H3$,J1$,W3,P3,R3:
      GOSUB 5000
4075 T3 = T3 + T:
      I1 = I1 + 1:
      IF I1 = I2
      THEN 6000 .....
4080 PRINT @402,"";
      INPUT H4$,J1$,W4,P4,R4:
      GOSUB 5000
4085 T4 = T4 + T:
      I1 = I1 + 1:
      IF I1 = I2
      THEN 6000 .....
4090 PRINT @466,"";
      INPUT H5$,J1$,W5,P5,R5:
      GOSUB 5000
4095 T5 = T5 + T:
      I1 = I1 + 1:
      IF I1 = I2
      THEN 6000 .....
4100 PRINT @530,"";
      INPUT H6$,J1$,W6,P6,R6:
      GOSUB 5000
4105 T6 = T6 + T:
      I1 = I1 + 1:
      IF I1 = I2
      THEN 6000 .....
4110 PRINT @594,"";
      INPUT H7$,J1$,W7,P7,R7:
      GOSUB 5000
4115 T7 = T7 + T:
      I1 = I1 + 1:
      IF I1 = I2
      THEN 6000 .....
4120 PRINT @658,"";
      INPUT H8$,J1$,W8,P8,R8:
      GOSUB 5000
4125 T8 = T8 + T:
      I1 = I1 + 1:
      IF I1 = I2
      THEN 6000 .....
4130 PRINT @722,"";
```

Program Explanation

```
      INPUT H9$,J1$,W9,P9,R9:
      GOSUB 5000
4135  T9 = T9 + T:
      GOTO 6000
-----
5000  IF   J1$ = "A"
      THEN T = A
5010  IF   J1$ = "B"
      THEN T = B
5020  IF   J1$ = "C"
      THEN T = C
5030  IF   J1$ = "D"
      THEN T = D
5040  IF   J1$ = "E"
      THEN T = E
5050  IF   J1$ = "F"
      THEN T = F
5060  IF   J1$ = "G"
      THEN T = G
5070  IF   J1$ = "H"
      THEN T = H
5080  IF   J1$ = "I"
      THEN T = I
5090  IF   J1$ = "J"
      THEN T = J
5100  IF   J1$ = "K"
      THEN T = K
5110  IF   J1$ = "L"
      THEN T = L
5120  IF   J1$ = "M"
      THEN T = M
5130  IF   J1$ = "N"
      THEN T = N
5140  IF   J1$ = "O"
      THEN T = O
5150  REM ***** WEIGHT
5160  IF   W1 = WE
      THEN T1 = T1 + 1
5170  IF   W2 = WE
      THEN T2 = T2 + 1
5180  IF   W3 = WE
      THEN T3 = T3 + 1
5190  IF   W4 = WE
      THEN T4 = T4 + 1
5200  IF   W5 = WE
      THEN T5 = T5 + 1
5210  IF   W6 = WE
      THEN T6 = T6 + 1
5220  IF   W7 = WE
      THEN T7 = T7 + 1
5230  IF   W8 = WE
```

## Program Explanation

```
        THEN T8 = T8 + 1
5240 IF   W9 = WE
        THEN T9 = T9 + 1
5250 W1 = 0:
        W2 = 0:
        W3 = 0:
        W4 = 0:
        W5 = 0:
        W6 = 0:
        W7 = 0:
        W8 = 0:
        W9 = 0
5260 REM ***** POST
5270 IF   P1 = PO
        THEN T1 = T1 + 1
5280 IF   P2 = PO
        THEN T2 = T2 + 1
5290 IF   P3 = PO
        THEN T3 = T3 + 1
5300 IF   P4 = PO
        THEN T4 = T4 + 1
5310 IF   P5 = PO
        THEN T5 = T5 + 1
5320 IF   P6 = PO
        THEN T6 = T6 + 1
5330 IF   P7 = PO
        THEN T7 = T7 + 1
5340 IF   P8 = PO
        THEN T8 = T8 + 1
5350 IF   P9 = PO
        THEN T9 = T9 + 1
5360 P1 = 0:
        P2 = 0:
        P3 = 0:
        P4 = 0:
        P5 = 0:
        P6 = 0:
        P7 = 0:
        P8 = 0:
        P9 = 0
5370 REM ***** PAST RECORD
5380 IF   R1 > 0
        THEN T1 = T1 + 1
5390 IF   R2 > 0
        THEN T2 = T2 + 1
5400 IF   R3 > 0
        THEN T3 = T3 + 1
5410 IF   R4 > 0
        THEN T4 = T4 + 1
5420 IF   R5 > 0
        THEN T5 = T5 + 1
```

Program Explanation

```

5430 IF R6 > 0
      THEN T6 = T6 + 1
5440 IF R7 > 0
      THEN T7 = T7 + 1
5450 IF R8 > 0
      THEN T8 = T8 + 1
5460 IF R9 > 0
      THEN T9 = T9 + 1
5470 R1 = 0:
      R2 = 0:
      R3 = 0:
      R4 = 0:
      R5 = 0:
      R6 = 0:
      R7 = 0:
      R8 = 0:
      R9 = 0
5590 RETURN

```

---

```

6000 CLS :
      I1 = 0
6010 PRINT @466,"PLEASE WAIT FOR SELECTIONS"
6020 FOR X = 1 TO 2000:
      NEXT
6030 CLS
6040 X = 500
6050 IF T1 = X PRINT T1;H1$
6060 IF T2 = X PRINT T2;H2$
6070 IF T3 = X PRINT T3;H3$
6080 IF T4 = X PRINT T4;H4$
6090 IF T5 = X PRINT T5;H5$
6100 IF T6 = X PRINT T6;H6$
6110 IF T7 = X PRINT T7;H7$
6120 IF T8 = X PRINT T8;H8$
6130 IF T9 = X PRINT T9;H9$
6150 X = X - 1:
      IF X = 0 PRINT @462,"SELECTIONS FOR RACE # ";PP;:
          GOTO 6160 .....
6155 GOTO 6050

```

---

```

6160 PRINT @590,"MORE RACES (Y/N)";
6170 INPUT I$
6175 IF I$ = "N"
      THEN CLS :
          GOTO 20 .....
6180 PP = PP + 1:
      H1$ = " ":
      H2$ = " ":
      H3$ = " ":
      H4$ = " ":
      H5$ = " ":

```

## Program Explanation

```
H6$ = "" :  
H7$ = "" :  
H8$ = "" :  
H9$ = "" :  
6190 T = 0 :  
      T1 = 0 :  
      T2 = 0 :  
      T3 = 0 :  
      T4 = 0 :  
      T5 = 0 :  
      T6 = 0 :  
      T7 = 0 :  
      T8 = 0 :  
      T9 = 0 :  
6200 CLS :  
      GOTO 4010
```

---

```
7000 CLS  
7010 PRINT @468, "SAVING DATA"  
7020 PRINT # - 1, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, W, XX, A$  
7030 CLS  
7040 PRINT @468, "DATA SAVED"  
7050 FOR X = 1 TO 1000 :  
      NEXT  
7060 CLS  
7070 GOTO 20
```

---

---

# 7

---

## Methods of Display Using Block Moves

The FIND THE STATES program makes use of a block move of the complete screen.

### FIND THE STATES

In FIND THE STATES, a large map of the United States is displayed on the screen. You are given a state's capital and nickname, and you attempt to find the correct state on the map by placing the movable indicator on it. You move the indicator using the arrow keys. The game asks you 20 questions, and you are given a time limit to find the correct state.

The timer lasts long enough for you to go around the screen twice before your time is up. This should be enough to find the correct state if you know where it is, but if not, you haven't got much time for guessing.

The game keeps your score and gives the percentage of correct answers at the end of the game. This game saves the map in memory and makes a block move to the screen at the start of each question.

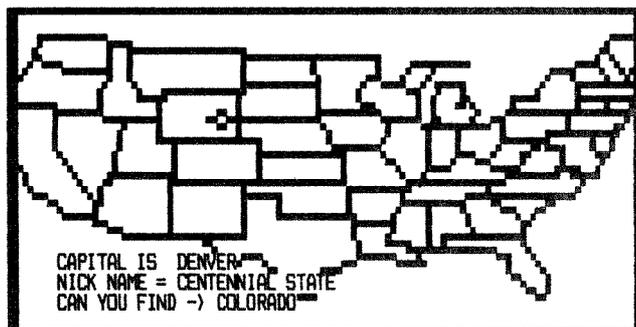


Figure 7.1 Find the States Program

```

5 CLS :
  PRINT CHR$( 23) :
  PRINT @460, "FIND THE STATES"
10 IF PEEK (16396) = 201
  THEN L$ = "T":
  ELSE L$ = "D"
11 IF L$ = "T"
  THEN A = 31700:
  ELSE A = - 23836
20 READ B:
  IF B = - 0
  THEN 450 .....
30 POKE A, B:
  A = A + 1:
  GOTO 20
-----
40 DATA 128,128,128,156,140,140,140,140,140,140,176,176
  ,144,128,128,128,128,128,128,128,128,128,128,128,128
  ,128,128,128,128,128,128,128,128,128,128,128,128,128
  ,128,128,128,128,128,128,128,128,128,128,128,128,128
  ,128,128,128,128,128
50 DATA 128,128,128,128,152,140,140,148
60 REM B$
70 DATA 128,128,154,164,176,144,128,128,128,128,149,128
  ,151,131,131,131,131,131,131,131,131,131,131,131,171,140
  ,140,140,140,140,140,172,140,140,140,140,140,140,148
  ,128,128,160,176,176,176,176,176,144,128,128,128,128
  ,128,128,128,128,128,128
80 DATA 176,176,150,137,144,128,149
90 REM C$
100 DATA 160,134,128,128,128,130,131,131,131,163,133,128
  ,165,128,128,128,128,128,128,128,128,128,128,170,176
  ,176,176,176,176,176,178,148,128,128,128,160,140,165
  ,176,184,129,160,134,176,176,176,128,128,128,128,128
  ,128,128,160,134
110 DATA 131,139,176,176,178,180,178,182,128
120 REM D$
130 DATA 170,176,176,176,176,128,128,128,128,149,128,128
  ,128,141,140,158,131,131,131,131,131,131,131,151,128
  ,128,128,128,128,128,128,149,128,128,128,165,176,128
  ,128,128,131,149,168,129,128,186,164,128,128,128,168
  ,140,140,133,128
140 DATA 128,128,157,140,140,156,140,172,133
150 REM E$
160 DATA 149,128,128,128,170,131,131,131,131,137,140,172
  ,140,140,140,149,128,128,128,128,128,128,128,157,140
  ,140,140,140,140,140,140,167,131,131,131,131,131,137
  ,156,140,140,165,186,180,176,176,184,177,152,172,131
  ,131,131,131,131
170 DATA 131,151,131,131,131,131,179,135,128

```

```

180 REM F$
190 DATA 165,128,128,128,170,128,128,128,128,128,128,149
,128,128,128,141,172,140,140,140,140,140,141,140
,148,128,128,128,128,128,128,165,176,176,176,176,152
,129,128,128,128,149,128,128,149,128,128,128,154,140
,140,172,156
200 DATA 183,179,131,131,171,131,171,129,128,128
210 REM G$
220 DATA 138,144,128,128,128,137,144,128,128,128,170,128
,128,128,128,128,170,128,128,128,128,128,128,128,128
,151,131,131,131,131,131,131,131,171,128,128,128,130
,140,144,128,128,181,176,152,131,131,140,166,176,176
,154,129
230 DATA 128,128,186,176,144,128,155,129,128,128,128
240 REM H$
250 DATA 128,138,144,128,128,128,130,164,128,152,135,131
,131,131,131,131,159,140,140,140,140,140,140,172,140
,141,140,140,140,140,140,140,140,142,176,176,176,176
,176,178,156,142,140,140,140,140,140,140,156,140,131
,131,131,131,131
260 DATA 129,130,155,131,129,128,128,128,128
270 REM I$
280 DATA 128,128,130,131,140,144,128,128,169,133,128,128
,128,128,128,128,149,128,128,128,128,128,128,170,131
,131,131,149,128,128,128,128,128,149,128,128,128
,128,154,140,140,156,140,140,172,131,131,139,167,179
,131,131,179
290 DATA 143,131,131,128,128,128,128,128,128,128
300 REM J$
310 DATA 128,128,128,128,128,130,131,131,139,140,176,144
,128,128,128,128,149,128,128,128,128,128,128,170,128
,128,128,131,131,131,131,131,131,131,171,140,140,140
,174,128,128,128,149,128,128,128,165,128,128,128,128
,169,134
320 DATA 128,128,128,128,128,128,128,128,128,128,128
330 REM K$
340 DATA 128,128,128,128,128,128,128,128,128,128,130
,131,131,131,131,131,131,131,139,167,131,131,131,128
,128,128,128,128,128,128,128,128,130,148,128,128
,141,140,180,176,181,186,179,179,179,179,179,131,131
,139,144
350 DATA 128,128,128,128,128,128,128,128,128,128,128
360 REM L$
370 DATA 128,128,128,128,128,128,128,128,128,128,128,128
,128,128,128,128,128,128,128,128,137,140,134,131
,137,176,128,128,128,128,160,176,152,140,131,131,131
,131,140,140,141,128,128,128,128,128,128,128,131,149
,128,130,164
380 DATA 128,128,128,128,128,128,128,128,128,128
390 REM M$
400 DATA 128,128,128,128,128,128,128,128,128,128,128

```

Find The States

```

,128,128,128,128,128,128,128,128,128,128,128,128,128
,128,128,137,176,128,168,129,128,128,128,128,128,128
,128,128,128,128,128,128,128,128,128,128,128,128,130
,137,176,128
410 DATA 149,128,128,128,128,128,128,128,128,128,128
420 REM N$
430 DATA 128,128,128,128,128,128,128,128,128,128,128,128
,128,128,128,128,128,128,128,128,128,128,128,128
,128,128,128,128,131,131,128,128,128,128,128,128,128
,128,128,128,128,128,128,128,128,128,128,128,128,128
,128,128,131
440 DATA 128,128,128,128,128,128,128,128,128,128,128,-0
450 IF L$ = "D"
THEN Z = - 21836:
ELSE Z = 31488
460 FOR X = Z TO Z + 11:
READ D:
POKE X,D:
NEXT
470 DATA 17,64,60,33,212,123,1,128,3,237,176,201
480 IF L$ = "T" POKE 16526,0:
POKE 16527,123
490 IF L$ = "D" POKE Z + 4,228:
POKE Z + 5,162:
DEF USR 0 = - 21836
600 CLS :
A = USR (0)
605 PRINT @0,"CORRECT ANSWERS ";SC;:
PRINT @40,"WRONG ANSWERS ";W;
620 R = RND (49):
IF R = 8
THEN 620 .....
630 IF R = 1S$ = "ALABAMA":
N$ = "COTTON STATE":
C$ = "MONTGOMERY":
L = 16044
640 IF R = 2S$ = "ARIZONA":
N$ = "APACHE STATE":
C$ = "PHOENIX":
L = 15948
650 IF R = 3S$ = "ARKANSAS":
N$ = "WONDER STATE":
C$ = "LITTLE ROCK":
L = 15972
660 IF R = 4S$ = "CALIFORNIA":
N$ = "GOLDEN STATE":
C$ = "SACRAMENTO":
L = 15746
670 IF R = 5S$ = "COLORADO":
N$ = "CENTENNIAL STATE":
C$ = "DENVER":

```

## Find The States

L = 15828  
680 IF R = 6S\$ = "CONNECTICUT":  
N\$ = "NUTMEG STATE":  
C\$ = "HARTFORD":  
L = 15674  
690 IF R = 7S\$ = "DELAWARE":  
N\$ = "DIAMOND STATE":  
C\$ = "DOVER":  
L = 15803  
710 IF R = 9S\$ = "FLORIDA":  
N\$ = "PENINSULA STATE":  
C\$ = "TALLAHASSEE":  
L = 16179  
720 IF R = 10S\$ = "GEORGIA":  
N\$ = "CRACKER STATE":  
C\$ = "ATLANTA":  
L = 16049  
730 IF R = 11S\$ = "IDAHO":  
N\$ = "GEM STATE":  
C\$ = "BOISE":  
L = 15627  
740 IF R = 12S\$ = "ILLINOIS":  
N\$ = "PRAIRIE STATE":  
C\$ = "SPRINGFIELD":  
L = 15784  
750 IF R = 13S\$ = "INDIANA":  
N\$ = "HOOSIER STATE":  
C\$ = "INDIANAPOLIS":  
L = 15787  
760 IF R = 14S\$ = "IOWA":  
N\$ = "HAWKEYE STATE":  
C\$ = "DES MOINES":  
L = 15714  
770 IF R = 15S\$ = "KANSAS":  
N\$ = "SUNFLOWER STATE":  
C\$ = "TOPEKA":  
L = 15836  
780 IF R = 16S\$ = "KENTUCKY":  
N\$ = "BLUE GRASS STATE":  
C\$ = "FRANKFORT":  
L = 15854  
790 IF R = 17S\$ = "LOUISIANA":  
N\$ = "PELICAN STATE":  
C\$ = "BATON ROUGE":  
L = 16100  
800 IF R = 18S\$ = "MAINE":  
N\$ = "PINE TREE STATE":  
C\$ = "AUGUSTA":  
L = 15549  
810 IF R = 19S\$ = "MARYLAND":  
N\$ = "OLD LINE STATE":

Find The States

C\$ = "ANNAPOLIS":  
L = 15803  
820 IF R = 20S\$ = "MASSACHUSETTS":  
N\$ = "BAY STATE":  
C\$ = "BOSTON":  
L = 15675  
830 IF R = 21S\$ = "MICHIGAN":  
N\$ = "WOLVERINE STATE":  
C\$ = "LANSING":  
L = 15660  
840 IF R = 22S\$ = "MINNESOTA":  
N\$ = "NORTH STAR STATE":  
C\$ = "ST. PAUL":  
L = 15585  
850 IF R = 23S\$ = "MISSISSIPPI":  
N\$ = "MAGNOLIA STATE":  
C\$ = "JACKSON":  
L = 16040  
860 IF R = 24S\$ = "MISSOURI":  
N\$ = "SHOW ME STATE":  
C\$ = "JEFFERSON CITY":  
L = 15843  
870 IF R = 25S\$ = "MONTANA":  
N\$ = "TREASURE STATE":  
C\$ = "HELENA":  
L = 15564  
880 IF R = 26S\$ = "NEBRASKA":  
N\$ = "CORNHUSKER STATE":  
C\$ = "LINCOLN":  
L = 15772  
890 IF R = 27S\$ = "NEVADA":  
N\$ = "SAGEBRUSH STATE":  
C\$ = "CARSON CITY":  
L = 15751  
900 IF R = 28S\$ = "NEW HAMPSHIRE":  
N\$ = "GRANITE STATE":  
C\$ = "CONCORD":  
L = 15548  
910 IF R = 29S\$ = "NEW JERSEY":  
N\$ = "GARDEN STATE":  
C\$ = "TRENTON":  
L = 15738  
920 IF R = 30S\$ = "NEW MEXICO":  
N\$ = "CACTUS STATE":  
C\$ = "SANTA FE":  
L = 15956  
930 IF R = 31S\$ = "NEW YORK":  
N\$ = "EMPIRE STATE":  
C\$ = "ALBANY":  
L = 15670  
940 IF R = 32S\$ = "NORTH CAROLINA":

N\$ = "TAR HEEL STATE":  
 C\$ = "RALEIGH":  
 L = 15924  
 950 IF R = 33S\$ = "NORTH DAKOTA":  
 N\$ = "FLICKERTAIL STATE":  
 C\$ = "BISMARCK":  
 L = 15579  
 960 IF R = 34S\$ = "OHIO":  
 N\$ = "BUCKEYE STATE":  
 C\$ = "COLUMBUS":  
 L = 15791  
 970 IF R = 35S\$ = "OKLAHOMA":  
 N\$ = "SOONER STATE":  
 C\$ = "OKLAHOMA CITY":  
 L = 15966  
 980 IF R = 36S\$ = "OREGON":  
 N\$ = "BEAVER STATE":  
 C\$ = "SALEM":  
 L = 15556  
 990 IF R = 37S\$ = "PENNSYLVANIA":  
 N\$ = "KEYSTONE STATE":  
 C\$ = "HARRISBURG":  
 L = 15732  
 1000 IF R = 38S\$ = "RHODE ISLAND":  
 N\$ = "LITTLE RHODY":  
 C\$ = "PROVIDENCE":  
 L = 15677  
 1010 IF R = 39S\$ = "SOUTH CAROLINA":  
 N\$ = "PALMETTO STATE":  
 C\$ = "COLUMBIA":  
 L = 15987  
 1020 IF R = 40S\$ = "SOUTH DAKOTA":  
 N\$ = "SUNSHINE STATE":  
 C\$ = "PIERRE":  
 L = 15643  
 1030 IF R = 41S\$ = "TENNESSEE":  
 N\$ = "VOLUNTEER STATE":  
 C\$ = "NASHVILLE":  
 L = 15915  
 1040 IF R = 42S\$ = "TEXAS":  
 N\$ = "LONE STAR STATE":  
 C\$ = "AUSTIN":  
 L = 16093  
 1050 IF R = 43S\$ = "UTAH":  
 N\$ = "DESERET":  
 C\$ = "SALT LAKE CITY":  
 L = 15821  
 1060 IF R = 44S\$ = "VERMONT":  
 N\$ = "GREEN MOUNTAIN STATE":  
 C\$ = "MONTPELIER":  
 L = 15610

## Find The States

```

1070 IF R = 45S$ = "VIRGINIA":
      N$ = "OLD DOMINION":
      C$ = "RICHMOND":
      L = 15862
1080 IF R = 46S$ = "WASHINGTON":
      N$ = "EVERGREEN STATE":
      C$ = "OLYMPIA":
      L = 15495
1090 IF R = 47S$ = "WEST VIRGINIA":
      N$ = "PANHANDLE STATE":
      C$ = "CHARLESTON":
      L = 15857
1100 IF R = 48S$ = "WISCONSIN":
      N$ = "BADGER STATE":
      C$ = "MADISON":
      L = 15654
1110 IF R = 49S$ = "WYOMING":
      N$ = "EQUALITY STATE":
      C$ = "CHEYENNE":
      L = 15699
2000 PRINT @832,"NICK NAME = ";N$;
2010 PRINT @768,"CAPITAL IS ";C$;
2020 PRINT @896,"CAN YOU FIND -> ";S$;
2030 A = 16000
2040 IF PEEK (14400) = 8
      THEN A = A - 64
2050 IF PEEK (14400) = 16
      THEN A = A + 64
2060 IF PEEK (14400) = 32
      THEN A = A - 1
2070 IF PEEK (14400) = 64
      THEN A = A + 1
2075 IF A < 15360 OR A > 16380
      THEN A = A1
2078 A1 = A
2080 B = PEEK (A):
      C = PEEK (A + 1):
      D = PEEK (A + 2)
2090 POKE A,140:
      POKE A + 1,179:
      POKE A + 2,140
2100 FOR X = 1 TO 10:
      NEXT
2110 POKE A,B:
      POKE A + 1,C:
      POKE A + 2,D
2120 IF A + 1 = L
      THEN FOR X = 1 TO 50:
            OUT 255,0:
            OUT 255,2:
            NEXT :

```

```

                SC = SC + 1:
                T = 0:
                GOTO 600 .....
2130      T = T + 1:
          IF T = 200W = W + 1:
                T = 0:
                GOTO 600 .....
2140      IF W = 10 CLS :
                PRINT CHR$ (23):
                PRINT "END OF GAME":
                PRINT "YOU GOT";SC;"OUT OF";SC + W;"QUESTI
                ONS ":
                PRINT :
                PRINT "TO PLAY AGAIN HIT ENTER":
                INPUT I$:
                W = 0:
                SC = 0:
                T = 0:
                GOTO 600 .....
2150      GOTO 2040 .....
-----

```

---

### Program Explanation

Lines 10 through 60 print the name on the screen, and line 70 checks to be sure that the system is in LEVEL II TAPE. Lines 100 through 290 poke the score card into high memory. This is later block-moved to the screen when needed.

Lines 300 to 360 get the three initials of each player, and lines 370 to 425 block-move the score card to the screen, print the players' initials on the score card and move it back to high memory again.

Lines 430 to 489 print the dice on the screen. If the up arrow is pressed, execution jumps to lines 3000 to 3120 and draws the picture of the man running across the screen. When you release the up arrow key, you GOTO lines 520 to 690 to see whether the player wants to save, roll or enter score.

Lines 1990 to 2070 are the cursor and the score card controls, and lines 5000 to 18100 are where all computations take place for the scores. Don't even *think* about asking me how long I spent writing this section, — all I know is I never want to do it again!

Last (but not least) are lines 19000 to 19080, which total the scores and stop the game.

### Las Wages POKER

The idea for this game came from spending a weekend in Las Vegas and losing about a hundred dollars playing one of those electronic slot machines. The game plays very much like the real ones in Vegas.

You start by pressing enter, and the computer will deal out your hand. Next, select the cards you wish to discard by pressing their number. Those cards are then turned face down, and the computer replaces them with new ones.

## Las Wages Poker

After checking your hand to see how well you did, press the <CLEAR> key, and the game starts over. This game is like playing five-card draw poker. The cards are shuffled again after each hand!

There is no score in the game, but it's good practice for learning how to play your hand, and you don't have to worry about losing \$100.00!

The game of poker consists of 52 cards divided into four suits: spades, hearts, diamonds and clubs. Each suit has 13 cards ranking in this order: ace (high), king, queen, jack, 10, 9, 8, 7, 6, 5, 4, 3 and 2 (low). From two to eight people play in an actual game, — unless they're playing a slot machine.

Below is a list of the poker hands, starting with the highest-ranked first.

**Straight Flush** — five cards of the same suit and in sequence. The ace-high straight flush is called a Royal Flush.

**Four of a Kind** — such as four 7's.

**Full House** — three of one kind and two of another, such as 10, 10, 10, 3 and 3.

**Flush** — five cards of the same suit.

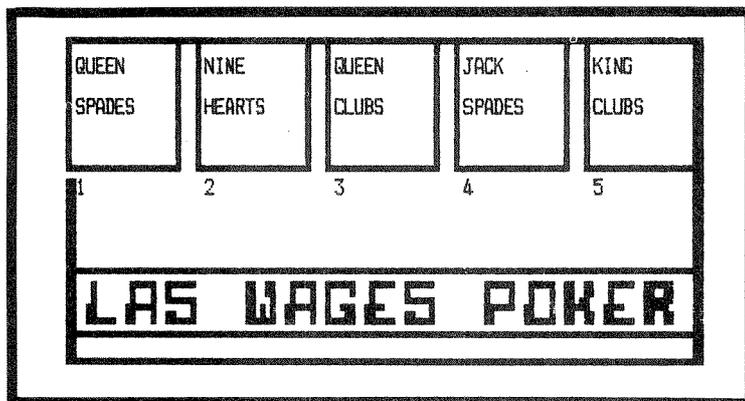
**Straight** — five cards in sequence, but having two or more suits.

**Three of a Kind** — such as 10, 10, 10, 3 and 8.

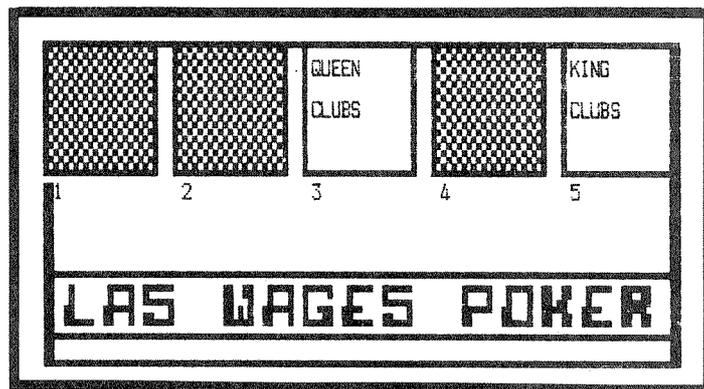
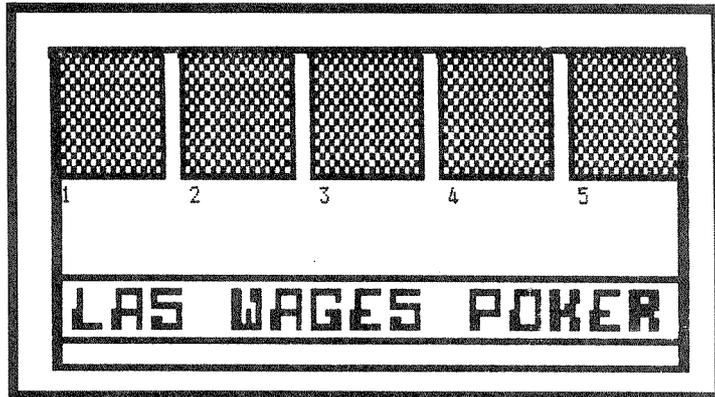
**Two Pairs** — such as 10, 10, Q, Q and 2.

**One Pair** — such as 10, 10, 2, 3 and 4.

If two players have the same hand, the one with the highest card wins. If both have exactly the same cards, they tie. There are 2,598,960 possible poker hands, but only 40 are straight flushes.



Las Wages Poker





```

END END END END END END END END END END
END END END END END END END END END END
END
END END END END END END END END END END
END
END END END END END END END END END END
END
END END END END END END END END END END
END
END END END END END END END END END END
"

```

```

-----
3020 A3$ = " GOTO LET LET
      CMD END GOTO LET LET
      CMD END GOTO LET LET
      CMD END GOTO LET LET
      CMD END END
      END END END END END END END END END END END END END END
      END END END END END END END END END END END END END END
      END END END END END END END END END END END END END END
      END END END END END END END END END END END END END END
      END END END END END END END END END END END END END END
      END END END END END END END END END END END END END END
      END END END END END END END END END END END END END END
      END END END END END END END END END END END END END "

```

```

-----
3030 A4$ = " END END
      END END END END END END END END END END END END END END
      END END END END END END END END END END END END END END
      END END END END END END END END END END END END END END
      END END END END END END END END END END END END END END
      END END END END END END END END END END END END END END
      END END END END END END END END END END END END END END
      END END END END END END END END END END END END END END
      END END END END END END END END END END END END END END
      LINE LET LET
      LET LET LET LET LET LET LET LET LET LET LET LET LET LET LET
      LET LET LET LET LET LET LET LET LET LET LET LET LET LET LET
      LET LET LET LET LET LET LET LET LET LET LET LET LET LET LET
      LET LET LET LET LET LET LET LET LET LET LET LET LET RSET "

```

```

3040 A5$ = "
ELSE END KILL END END END KILL SET LSET END AUTO CONT CONT
      END END END END END USING END LPRINT END KILL SET LSET
      END TROFF SET READ END KILL SET SET END AUTO CONT CONT
      END END END END END TROFF SET LSET END KILL SET LSET
      END USING OUT RESUME END KILL SET SET END USING CONT
      USING END END KILL

```

```

END USING DEF DEF END USING SET LSET END TAB( DEF CSAVE END
END END END END USING CSAVE CSAVE END USING SET LSET
END LLIST PRINT USING END USING CONT DEF END TAB( DEF
CSAVE END END END END END USING SET SET END USING DEF
CSAVE END USING SET TO END USING CONT DEF END USING
SET TO END END KILL GOTO LET LET LET LET LET LET LET
LET LET LET LET LET LET LET LET LET LET LET LET LET
LET LET LET LET LET LET LET LET LET LET LET LET LET
LET LET LET LET LET LET LET LET LET LET LET LET LET
LET LET LET LET LET LET LET LET LET LET LET LET LET
LET LET LET RUN "

```

---

```

3050 A6$ = " END END END END END END END END END END
END END END END END END END END END END END END END
END END END END END END END END END END END END END
END END END END END END END END END END END END END
END END END END END END END END END END END END END
"

```

---

```

3060 CLS :
PRINT A1$;A2$;A3$;A4$;A5$;A6$;
3065 ZZ = 448:
FOR X = 1 TO 5:
PRINT @ZZ,X;:
ZZ = ZZ + 13:
NEXT
3070 A$ = CHR$ (159) + STRING$ (10,155)
3072 B$ = CHR$ (183) + STRING$ (10,166)
3073 C$ = CHR$ (157) + STRING$ (10,153)
3074 D$ = STRING$ (11,141)
3080 IF C1 = 0 PRINT @0,A$;:
PRINT @64,B$;:
PRINT @128,C$;:
PRINT @192,B$;:
PRINT @256,C$;:
PRINT @320,B$;:
PRINT @384,D$;
3090 IF C2 = 0 PRINT @13,A$;:
PRINT @77,B$;:
PRINT @141,C$;:
PRINT @205,B$;:
PRINT @269,C$;:
PRINT @333,B$;:
PRINT @397,D$;
3100 IF C3 = 0 PRINT @26,A$;:
PRINT @90,B$;:
PRINT @154,C$;:
PRINT @218,B$;:
PRINT @282,C$;:
PRINT @346,B$;:
PRINT @410,D$;

```

```

3110 IF C4 = 0 PRINT @39,A$;:
      PRINT @103,B$;:
      PRINT @167,C$;:
      PRINT @231,B$;:
      PRINT @295,C$;:
      PRINT @359,B$;:
      PRINT @423,D$;
3120 IF C5 = 0 PRINT @52,A$;:
      PRINT @116,B$;:
      PRINT @180,C$;:
      PRINT @244,B$;:
      PRINT @308,C$;:
      PRINT @372,B$;:
      PRINT @436,D$;
3125 IF T = 1 RETURN .....
3130 IF PEEK (14400) < > 1
      THEN 3130 .....
3140 A$ = CHR$ (151) + STRING$ (10,131)
3150 B$ = CHR$ (149) + STRING$ (10,128)
3160 C$ = CHR$ (149) + STRING$ (10,128)
3170 D$ = CHR$ (141) + STRING$ (10,140)
3175 IF T = 2 RETURN .....
3180 T = 1:
      GOSUB 3080
4030 A$(1) = "ACE"
4040 A$(2) = "TWO"
4050 A$(3) = "THREE"
4060 A$(4) = "FOUR"
4070 A$(5) = "FIVE"
4080 A$(6) = "SIX"
4090 A$(7) = "SEVEN"
4100 A$(8) = "EIGHT"
4110 A$(9) = "NINE"
4120 A$(10) = "TEN"
4130 A$(11) = "JACK"
4140 A$(12) = "QUEEN"
4150 A$(13) = "KING"
4160 B$(1) = "HEARTS"
4170 B$(2) = "CLUBS"
4180 B$(3) = "DIAMONDS"
4190 B$(4) = "SPADES"
4200 FOR I = 1 TO 52:
      C(I) = I:
      NEXT I
4210 D = 53
4240 FOR I = 1 TO 5
4250   D = D - 1
4260   X = INT (D * RND (0) + 1)
4270   H(I) = C(X)
4280   C(X) = C(D)
4290   Q = INT ((H(I) - 1) / 13)

```

Las Wages Poker

```

4300      Y = H(I) - 1 - Q * 13
4310      H$(I) = A$(Y + 1)
4315      Z$(I) = B$(Q + 1)
4320 NEXT I
4325 IF   T = 0 RETURN .....
4330 PP = 65
4350 FOR  J = 1 TO 5
4360      PRINT @PP,H$(J);:
          PRINT @PP + 128,Z$(J);:
          PP = PP + 13
4370 NEXT J
4380 C1 = 1:
          C2 = 1:
          C3 = 1:
          C4 = 1:
          C5 = 1:
          E = 0
4390 I$ = INKEY$ :
          IF   I$ = ""
          THEN 4390 .....
4410 IF   I$ = "1"
          THEN C1 = 0:
          E = E + 1
4420 IF   I$ = "2"
          THEN C2 = 0:
          E = E + 1
4430 IF   I$ = "3"
          THEN C3 = 0:
          E = E + 1
4440 IF   I$ = "4"
          THEN C4 = 0:
          E = E + 1
4450 IF   I$ = "5"
          THEN C5 = 0:
          E = E + 1
4452 IF   PEEK (14400) = 1
          THEN 5000 .....
4455 IF   E = 4
          THEN 5000 .....
4460 GOSUB 3070:
          I$ = "":
          GOTO 4390
-----
5000 T = 2:
          GOSUB 3140
5010 T = 1:
          GOSUB 3080
5020 T = 0:
          GOSUB 4030
5030 FOR  J = 1 TO 5
5040      IF   C1 = 0 PRINT @65,H$(J);:

```

```

        PRINT @193,Z$(J);:
        C1 = 1:
        NEXT
5050    IF    C2 = 0 PRINT @78,H$(J);:
        PRINT @206,Z$(J);:
        C2 = 1:
        NEXT
5060    IF    C3 = 0 PRINT @91,H$(J);:
        PRINT @219,Z$(J);:
        C3 = 1:
        NEXT
5070    IF    C4 = 0 PRINT @104,H$(J);:
        PRINT @232,Z$(J);:
        C4 = 1:
        NEXT
5080    IF    C5 = 0 PRINT @117,H$(J);:
        PRINT @245,Z$(J);:
        C5 = 1:
        NEXT
5090 NEXT
6000 IF    PEEK (14400) < > 2
        THEN 6000: .....
        ELSE RUN .....
-----

```

**Program Explanation**

- Line 5 clears the screen; prints the title.
- Lines 10 to 30 save the map in memory.
- Lines 40 to 440 are data lines.
- Lines 450 to 600 get the block move routine in memory.
- Line 605 keeps the scores.
- Line 620 selects the questions.
- Lines 630 to 1110 are data for questions and screen locations.
- Lines 2000 to 2020 print data on states.
- Lines 2030 to 2110 move indicator.
- Lines 2120 to 2130 sound and points.
- Lines 2140 to 2150 end of game.

### Computer YAHTZEE

Computer YAHTZEE is played very much like the original game. The chief difference is that you are allowed only one yahtzee per player, and up to four players can play. The game is also cheat-proof. The computer keeps the scores, but it lets the players select where they wish to place their scores the board. If you try to put 3-of-a-kind on the space marked Full House, the computer will give a zero score in that position. After the last player has made his last roll at the end of the game, the computer totals all the players' scores and ends the game.

This game has a lot of graphics in it. As you roll the dice by pressing the up arrow, a little man runs across the screen to let you know you are still rolling. When you release the up arrow key, the man is cleared from the screen and you have the option of keeping the dice or rolling again.

Each player is allowed three rolls, and the screen shows the dice that you save from each roll. After your third roll, you go to the score card and move the cursor on the screen to the position where you want to place your score. Do this by using the up or down arrow. The score card on the screen is fully protected from a player making an improper move. After the cursor is in the correct position on the card, press the space bar to enter your score; then press the clear key to continue playing.

ONES = 3			ADD ONLY ACES	
TWOS = 6			ADD ONLY TWOS	
THREES = 9			ADD ONLY THREE	
FOURS = 12			ADD ONLY FOURS	
FIVES = 15			ADD ONLY FIVES	
SIXES = 18			ADD ONLY SIXES	
TOTAL = 63				
<hr/>				
3 OF A KIND			TOTAL ALL DICE	
4 OF A KIND			TOTAL ALL DICE	
FULL HOUSE			TOTAL SCORE 25	
SM. STRAIGHT			TOTAL SCORE 30	
LG. STRAIGHT			TOTAL SCORE 40	
Y A H T Z E			TOTAL SCORE 50	
C H A N C E			TOTAL ALL DICE	
GRAND TOTAL				

Figure 7.3 YAHTZEE Program

```

10 CLS :
   CLEAR 2000:
   ' COMPUTER YAHTZEE BY TOM DEMPSEY
20 PRINT CHR$(23):
   PRINT
30 PRINT STRING$(32,191);
40 FOR X = 1 TO 12:
   PRINT CHR$(191) + STRING$(30,128) + CHR$(191)
   );:
   NEXT
50 PRINT STRING$(32,191);
60 PRINT @466,"Y A H T Z E E";
70 FOR X = 1 TO 500:
   NEXT :
   CLS
100 DATA 149,79,78,69,83,128,128,128,61,128,128,51,170,1
   28,128,128,128,128,128,149,128,128,128,128,128,1
   49,128,128,128,128,128,170,128,128,128,128,128,1
   49,65,68,68,128,79,78,76,89,128,65,67,69,83,128,170,
   131
110 DATA 131,131,131,131,131,131,131,131,171,149,84,87,79,83
   ,128,128,128,61,128,128,54,170,128,128,128,128,1
   28,149,128,128,128,128,128,128,128,149,128,128,128,1
   28,170,128,128,128,128,128,128,149,65,68,68,128,79,7
   8,76,89
120 DATA 128,84,87,79,83,128,170,128,128,128,128,128,128
   ,128,128,170,149,84,72,82,69,69,83,128,61,128,128,57
   ,170,128,128,128,128,128,128,149,128,128,128,128,128
   ,128,149,128,128,128,128,128,170,128,128,128,128,128
   ,128,149
130 DATA 65,68,68,128,79,78,76,89,128,84,72,82,69,69,170
   ,128,128,128,128,128,128,128,170,149,70,79,85,82
   ,83,128,128,61,128,49,50,170,128,128,128,128,128,128
   ,149,128,128,128,128,128,128,149,128,128,128,128,128
140 DATA 170,128,128,128,128,128,128,149,65,68,68,128,79
   ,78,76,89,128,70,79,85,82,83,170,128,128,128,128,128
   ,128,128,128,170,149,70,73,86,69,83,128,128,61,128,4
   9,53,170,128,128,128,128,128,128,149,128,128,128,128
150 DATA 128,128,149,128,128,128,128,128,170,128,128,128
   ,128,128,128,149,65,68,68,128,79,78,76,89,128,70,73,
   86,69,83,170,128,128,128,128,128,128,128,128,170,149
   ,83,73,88,69,83,128,128,61,128,49,56,170,128,128,128
160 DATA 128,128,128,149,128,128,128,128,128,128,149,128
   ,128,128,128,128,170,128,128,128,128,128,128,149,65,
   68,68,128,79,78,76,89,128,83,73,88,69,83,170,128,128
   ,128,128,128,128,128,128,170,149,84,79,84,65,76,128,
   128

```

```

170 DATA 61,128,54,51,170,128,128,128,128,128,128,149,12
8,128,128,128,128,128,128,149,128,128,128,128,128,170,12
8,128,128,128,128,128,128,149,128,128,128,128,128,128,12
8,128,128,128,128,128,128,170,128,128,128,128,128,12
8,128,128,128,170
180 DATA 159,143,143,143,143,143,143,143,143,143,143,143
,175,128,128,128,128,128,128,128,149,128,128,128,128,128
,128,149,128,128,128,128,128,128,170,128,128,128,128,128
,128,159,143,143,143,143,143,143,143,143,143,143,143,143
,143,143,143,175,128
190 DATA 128,128,128,128,128,128,128,128,170,149,51,128,79,7
0,128,65,128,75,73,78,68,170,128,128,128,128,128,128
,149,128,128,128,128,128,128,128,149,128,128,128,128,128
,170,128,128,128,128,128,128,128,149,84,79,84,65,76,128,
65,76
200 DATA 76,128,68,73,67,69,170,128,128,128,128,128,128,
128,128,170,149,52,128,79,70,128,65,128,75,73,78,68,
170,128,128,128,128,128,128,128,149,128,128,128,128,128,
128,149,128,128,128,128,128,128,170,128,128,128,128,128,
128,149
210 DATA 84,79,84,65,76,128,65,76,76,128,68,73,67,69,170
,128,128,128,128,128,128,128,128,170,149,70,85,76,76
,128,128,72,79,85,83,69,170,128,128,128,128,128,128,
149,128,128,128,128,128,128,149,128,128,128,128,128
220 DATA 170,128,128,128,128,128,128,128,128,149,84,79,84,65,76,
128,83,67,79,82,69,128,50,53,170,128,128,128,128,128
,128,128,128,170,149,83,77,46,83,84,82,65,73,71,72,8
4,170,128,128,128,128,128,128,149,128,128,128,128
230 DATA 128,128,149,128,128,128,128,128,170,128,128,128
,128,128,128,149,84,79,84,65,76,128,83,67,79,82,69,1
28,51,48,170,128,128,128,128,128,128,128,128,128,170,149
,76,71,46,83,84,82,65,73,71,72,84,170,128,128,128
240 DATA 128,128,128,149,128,128,128,128,128,128,149,128
,128,128,128,128,170,128,128,128,128,128,128,149,84,
79,84,65,76,128,83,67,79,82,69,128,52,48,170,128,128
,128,128,128,128,128,128,170,149,89,128,65,128,72,12
8,84
250 DATA 128,90,128,69,170,128,128,128,128,128,128,149,1
28,128,128,128,128,128,149,128,128,128,128,128,170,1
28,128,128,128,128,128,149,84,79,84,65,76,128,83,67,
79,82,69,128,53,48,170,128,128,128,128,128,128,128,1
28,170
260 DATA 149,67,128,72,128,65,128,78,128,67,128,69,170,1
28,128,128,128,128,128,149,128,128,128,128,128,128,1
49,128,128,128,128,128,170,128,128,128,128,128,128,1
49,84,79,84,65,76,128,65,76,76,128,68,73,67,69,170,1
28
270 DATA 128,128,128,128,128,128,128,170,149,71,82,65,78
,68,128,84,79,84,65,76,170,128,128,128,128,128,128,1
49,128,128,128,128,128,128,149,128,128,128,128,128,1
70,128,128,128,128,128,128,149,128,128,128,128,128,1

```

```

28,128,128
280 DATA 128,128,128,128,128,128,170,176,176,176,176,176
,176,176,176,186
290 FOR M = 15360 TO 16383:
      READ C:
      POKE M,C:
      NEXT :
      GOSUB 50000:
      CLS
300 INPUT "HOW MANY PLAYERS (1-4)";T
305 IF T < 1 OR T > 4
      THEN CLS :
      GOTO 300 .....
306 CLS
310 FOR X = 1 TO T:
      PRINT "ENTER INITIALS OF PLAYER #";X;
320 A$ = INKEY$ :
      IF A$ = ""
      THEN 320 .....
325 IF A$ = CHR$ (13)
      THEN PRINT :
      NEXT X
330 PRINT A$;
340 N$(X) = N$(X) + A$:
      IF LEN (N$(X)) = 3
      THEN PRINT :
      NEXT X
350 IF X < T + 1
      THEN 320 .....
360 GOSUB 50110
410 A = 462:
      FOR X = 1 TO T:
          PRINT @A,N$(X);:
          A = A + 7:
      NEXT X:
      GOTO 2030 .....
-----
425 CLS
430 A1$ = STRING$ (9,191):
      A2$ = STRING$ (9,128)
435 C = 0
440 IF D1 = 0 PRINT @C,A1$;:
      PRINT @C + 64,A1$;:
      PRINT @C + 128,A1$;
441 PRINT @C + 192,"1";:
      PRINT @67,R1;
445 C = 11
450 IF D2 = 0 PRINT @C,A1$;:
      PRINT @C + 64,A1$;:
      PRINT @C + 128,A1$;
451 PRINT @C + 192,"2";:

```

Computer Yahtzee

```

455     PRINT @78,R2;
460     C = 22
        IF D3 = 0 PRINT @C,A1$;:
            PRINT @C + 64,A1$;:
            PRINT @C + 128,A1$;
461     PRINT @C + 192,"3";:
        PRINT @89,R3;
465     C = 33
470     IF D4 = 0 PRINT @C,A1$;:
            PRINT @C + 64,A1$;:
            PRINT @C + 128,A1$;
471     PRINT @C + 192,"4";:
        PRINT @100,R4;
475     C = 44
480     IF D5 = 0 PRINT @C,A1$;:
            PRINT @C + 64,A1$;:
            PRINT @C + 128,A1$;
481     PRINT @C + 192,"5";:
        PRINT @111,R5;
489     IF V1 = 1
        THEN V1 = 0:
            GOTO 520 .....
490     PRINT @448,"PRESS UP ARROW TO ROLL DICE "N$(PX
        )
500     I$ = INKEY$ :
        IF I$ = "["
        THEN GOSUB 3000:
        ELSE 500 .....
510     PRINT @A,A2$;:
        PRINT @A + 64,A2$;:
        PRINT @A + 128,A2$;:
        PRINT @A + 190,A2$;
520     PRINT @448,"(S)=SAVE DICE (E)=ENTER SCORE (R)
        =ROLL AGAIN (C)=SEE SCORE"
525     TT = TT + 1
530     I$ = INKEY$ :
        IF I$ = ""
        THEN 530 .....
535     IF TT = 3
        THEN TT = 0:
            PRINT "3 ROLL LIMIT":
            INPUT "PRESS ENTER TO RECORD SCORE";I$:
            GOSUB 50110:
            GOTO 1990 .....
540     IF I$ = "R"
        THEN 425 .....
550     IF I$ = "S"
        THEN 600 .....
560     IF I$ = "E" GOSUB 50110:
        GOSUB 1990:
        TT = 0:

```

```

565      IF      GOTO 425 .....
          I$ = "C" GOSUB 50110:
          FOR X = 1 TO 30000:
              IF PEEK (14400) < > 2
              THEN NEXT X:
              ELSE X = 0:
                  V1 = 1:
                  TT = TT - 1:
                  GOSUB 50000:
                  GOTO 425 .....
-----
570      GOTO 530 .....
-----
600      PRINT "WHICH DICE DO YOU WANT TO SAVE ? (E
          NTER (R) TO ROLL)"
610      I$ = " ":
          INPUT I$
630      IF I$ = "1"
          THEN D1 = 1
640      IF I$ = "2"
          THEN D2 = 1
650      IF I$ = "3"
          THEN D3 = 1
660      IF I$ = "4"
          THEN D4 = 1
670      IF I$ = "5"
          THEN D5 = 1
680      IF I$ = "R"
          THEN 425 .....
690      GOTO 610 .....
-----
1990     IF PX <= 1
          THEN P = 15373:
          ELSE IF PX = 2
              THEN P = 15380:
              ELSE IF PX = 3
                  THEN P = 15387:
                  ELSE IF PX = 4
                      THEN P = 15394
1991     W = P:
          LN = 1
2000     S = 120:
          PRINT @S,"PRESS";:
          S = S + 64:
          PRINT @S,"SPACE";:
          S = S + 64:
          PRINT @S,"TO";:
          S = S + 64:
          PRINT @S,"SCORE";
2001     Q = PEEK (P):
          POKE P,143:

```

```

FOR E = 1 TO 50:
NEXT :
POKE P,Q
2002 IF PEEK (14400) = 8
THEN P = P - 64:
LN = LN - 1:
IF LN < 1
THEN LN = 1
2003 IF PEEK (14400) = 16
THEN P = P + 64:
LN = LN + 1:
IF LN > 16
THEN LN = 16
2004 IF P < W
THEN P = W:
ELSE IF P > W + 960
THEN P = W + 896
2005 IF PEEK (14400) = 128 GOTO 4000 .....
2006 PRINT @P - 15360, " ";
2007 GOTO 2001 .....
-----
2030 GOSUB 50000
2041 IF PX = 1
THEN S1 = S1 + Z
2042 IF PX = 2
THEN S2 = S2 + Z
2043 IF PX = 3
THEN S3 = S3 + Z
2044 IF PX = 4
THEN S4 = S4 + Z
2045 Z = 0
2046 REM *** SCORE KEEPER [|||||]
2050 CLS :
PX = PX + 1:
:
IF PX > T
THEN PX = 1
2060 D1 = 0:
D2 = 0:
D3 = 0:
D4 = 0:
D5 = 0:
R1 = 0:
R2 = 0:
R3 = 0:
R4 = 0:
R5 = 0
2070 H$ = " ":
GOTO 425 .....
-----
3000 PRINT @448, " "

```

```

;
3010  A$ = CHR$ (191) + CHR$ (179) + CHR$ (191) +
      CHR$ (140) + STRING$ (60,32) + CHR$ (156) +
      CHR$ (191) + STRING$ (62,32) + CHR$ (131) +
      CHR$ (191) + STRING$ (2,179) + STRING$ (59
      ,32) + CHR$ (156) + CHR$ (140) + CHR$ (143
      ) + CHR$ (32) + CHR$ (143) + CHR$ (140)
3020  B$ = CHR$ (191) + CHR$ (179) + CHR$ (191) +
      CHR$ (140) + STRING$ (60,32) + CHR$ (168) +
      CHR$ (191) + STRING$ (2,176) + STRING$ (61
      ,32) + CHR$ (191) + CHR$ (176) + STRING$ (
      62,32) + STRING$ (2,143) + CHR$ (140)
3030  C$ = CHR$ (191) + CHR$ (179) + CHR$ (191) +
      CHR$ (140) + STRING$ (59,32) + CHR$ (168) +
      CHR$ (140) + CHR$ (191) + STRING$ (2,176) +
      STRING$ (61,32) + CHR$ (191) + STRING$ (2,
      176) + STRING$ (58,32) + CHR$ (143) + STRING$
      (3,131) + CHR$ (32) + CHR$ (143) + CHR$ (1
      40)
3040  D$ = CHR$ (191) + CHR$ (179) + CHR$ (191) +
      CHR$ (140) + STRING$ (60,32) + CHR$ (168) +
      CHR$ (191) + CHR$ (176) + CHR$ (154) + STRING$
      (61,32) + CHR$ (191) + CHR$ (176) + STRING$
      (60,32) + CHR$ (143) + STRING$ (2,131) + CHR$
      (191) + CHR$ (176)
3050  A = 704
3055  IF PEEK (14400) < > 8
      THEN RETURN .....
3060  PRINT @A,A$:
      PRINT @A,"      ":
      A = A + 1
3065  R = RND (6):
      IF D1 = 0 PRINT @67,R;:
      R1 = R
3070  PRINT @A,B$:
      PRINT @A,"      ":
      :
      A = A + 1
3075  R = RND (6):
      IF D2 = 0 PRINT @78,R;:
      R2 = R
3080  PRINT @A,C$:
      PRINT @A,"      ":
      A = A + 1
3085  R = RND (6):
      IF D3 = 0 PRINT @89,R;:
      R3 = R
3090  B = A
3095  R = RND (6):
      IF D4 = 0 PRINT @100,R;:
      R4 = R

```

```

3100      PRINT @A,D$:
          PRINT @A,"      ":
          A = A + 1
3105      R = RND (6):
          IF D5 = 0 PRINT @111,R;:
          R5 = R
3110      IF A > 755 RETURN .....
3120      GOTO 3055 .....
-----
4000      PRINT @184,"CLEAR";:
          PRINT @312,"PLAY ";
4005      PRINT @P - 15360,"";
4006      IF PEEK (P + 2) < > 128
          THEN 2000 .....
4010      ON LN GOTO 5000,6000,7000,8000,9000,10000,
          4020,4020,11000,12000,13000,14000,15000 .....
-----
4020      IF LN = 14
          THEN 16000: .....
          ELSE IF LN = 15
          THEN 17000 .....
4030      GOTO 2000 .....
-----
4040      ' :::::::::::::::::::: ONES
5000      IF R1 = 1Z = Z + 1
5010      IF R2 = 1Z = Z + 1
5020      IF R3 = 1Z = Z + 1
5030      IF R4 = 1Z = Z + 1
5040      IF R5 = 1Z = Z + 1
5060      GOTO 18000 .....
-----
5070      ' :::::::::::::::::::: TWOS
6000      IF R1 = 2Z = Z + 2
6010      IF R2 = 2Z = Z + 2
6020      IF R3 = 2Z = Z + 2
6030      IF R4 = 2Z = Z + 2
6040      IF R5 = 2Z = Z + 2
6060      GOTO 18000 .....
-----
6070      ' :::::::::::::::::::: THREES
7000      IF R1 = 3Z = Z + 3
7010      IF R2 = 3Z = Z + 3
7020      IF R3 = 3Z = Z + 3
7030      IF R4 = 3Z = Z + 3
7040      IF R5 = 3Z = Z + 3
7050      GOTO 18000 .....
-----
7060      ' :::::::::::::::::::: FOURS
8000      IF R1 = 4Z = Z + 4
8010      IF R2 = 4Z = Z + 4
8020      IF R3 = 4Z = Z + 4

```

Computer Yahtzee

```

8030      IF   R4 = 4Z = Z + 4
8040      IF   R5 = 4Z = Z + 4
8050      GOTO 18000 .....
-----
8060      '   ::::::::::::::::::::: FIVES
9000      IF   R1 = 5Z = Z + 5
9010      IF   R2 = 5Z = Z + 5
9020      IF   R3 = 5Z = Z + 5
9030      IF   R4 = 5Z = Z + 5
9040      IF   R5 = 5Z = Z + 5
9050      GOTO 18000 .....
-----
9060      '   ::::::::::::::::::::: SIXES
10000     IF   R1 = 6Z = Z + 6
10010     IF   R2 = 6Z = Z + 6
10020     IF   R3 = 6Z = Z + 6
10030     IF   R4 = 6Z = Z + 6
10040     IF   R5 = 6Z = Z + 6
10050     GOTO 18000 .....
-----
11060     '   ::::::::::::::::::::: 3 OF A KIND
11000     FOR  X = 1 TO 6
11010         IF   R1 = X
                THEN Z = Z + 1
11020         IF   R2 = X
                THEN Z = Z + 1
11030         IF   R3 = X
                THEN Z = Z + 1
11040         IF   R4 = X
                THEN Z = Z + 1
11050         IF   R5 = X
                THEN Z = Z + 1
11060         IF   Z >= 3
                THEN Z = R1 + R2 + R3 + R4 + R5:
                GOTO 18000 .....
11070         Z = 0:
                NEXT X:
                Z = 0:
                GOTO 18000 .....
-----
11080     '   ::::::::::::::::::::: 4 OF A KIND
12000     FOR  X = 1 TO 6
12010         IF   R1 = X
                THEN Z = Z + 1
12020         IF   R2 = X
                THEN Z = Z + 1
12030         IF   R3 = X
                THEN Z = Z + 1
12040         IF   R4 = X
                THEN Z = Z + 1
12050         IF   R5 = X

```

Computer Yahtzee

```

12060          THEN Z = Z + 1
              IF    Z >= 4
              THEN Z = R1 + R2 + R3 + R4 + R5:
              GOTO 18000 .....
12070          Z = 0:
          NEXT X:
          Z = 0:
          GOTO 18000 .....
-----
12080          ' ::::::::::::::::::::: FULL HOUSE
13000          FOR X = 1 TO 6
13010              IF R1 = X
              THEN Z = Z + 1:
              E1 = 1
13020              IF R2 = X
              THEN Z = Z + 1:
              E2 = 1
13030              IF R3 = X
              THEN Z = Z + 1:
              E3 = 1
13040              IF R4 = X
              THEN Z = Z + 1:
              E4 = 1
13050              IF R5 = X
              THEN Z = Z + 1:
              E5 = 1
13060              IF Z = 3
              THEN Z = 0:
              GOTO 13080 .....
13070          Z = 0:
          E1 = 0:
          E2 = 0:
          E3 = 0:
          E4 = 0:
          E5 = 0:
          NEXT X:
          Z = 0:
          GOTO 18000 .....
-----
13080          FOR X = 1 TO 6
13090              IF E1 = 0 AND R1 = X
              THEN Z = Z + 1
13100              IF E2 = 0 AND R2 = X
              THEN Z = Z + 1
13110              IF E3 = 0 AND R3 = X
              THEN Z = Z + 1
13120              IF E4 = 0 AND R4 = X
              THEN Z = Z + 1
13130              IF E5 = 0 AND R5 = X
              THEN Z = Z + 1
13140              IF Z = 2

```

```

                THEN Z = 25:
                GOTO 18000 .....
13150          Z = 0:
                NEXT X:
                Z = 0:
                GOTO 18000 .....
-----
13160          ' ::::::::::::::::::::: SMALL STRAIGHT
14000          X = 1
14010          IF R1 = X
                THEN X = X + 1:
                GOTO 14010 .....
14020          IF R2 = X
                THEN X = X + 1:
                GOTO 14010 .....
14030          IF R3 = X
                THEN X = X + 1:
                GOTO 14010 .....
14040          IF R4 = X
                THEN X = X + 1:
                GOTO 14010 .....
14050          IF R5 = X
                THEN X = X + 1:
                GOTO 14010 .....
14060          IF X >= 5
                THEN Z = 30:
                GOTO 18000 .....
14080          IF X = 1
                THEN X = 2:
                GOTO 14010 .....
14090          Z = 0:
                GOTO 18000 .....
-----
14150          ' ::::::::::::::::::::: LARGE STRAIGHT
15000          X = 1
15010          IF R1 = X
                THEN X = X + 1:
                GOTO 15010 .....
15020          IF R2 = X
                THEN X = X + 1:
                GOTO 15010 .....
15030          IF R3 = X
                THEN X = X + 1:
                GOTO 15010 .....
15040          IF R4 = X
                THEN X = X + 1:
                GOTO 15010 .....
15050          IF R5 = X
                THEN X = X + 1:
                GOTO 15010 .....
15060          IF X >= 6

```

Computer Yahtzee

```

IF EN = T * 13
THEN 19000 .....
18100 IF PEEK (14400) = 2
THEN 2030: .....
ELSE 18100 .....
-----
19000 IF Z1 > = 63
THEN Z1 = Z1 + 35
19010 IF Z2 > = 63
THEN Z2 = Z2 + 35
19020 IF Z3 > = 63
THEN Z3 = Z3 + 35
19030 IF Z4 > = 63
THEN Z4 = Z4 + 35
19040 IF Z1 > 0 PRINT @397,Z1;:
PRINT @973,Y1 + Z1;
19050 IF Z2 > 0 PRINT @404,Z2;:
PRINT @980,Y2 + Z2;
19060 IF Z3 > 0 PRINT @411,Z3;:
PRINT @987,Y3 + Z3;
19070 IF Z4 > 0 PRINT @418,Z4;:
PRINT @994,Y4 + Z4;
19080 GOTO 19080 .....
-----
50000 ' BLOCKMOVE TO MEMORY SUBROUTINE
50010 ZZ = PEEK (16634) * 256 + PEEK (16633) + 1
000
50020 ZY = ZZ + 12:
MS = INT (ZY / 256):
LS = ZY - MS * 256
50030 IF ZZ > 32767
THEN ZY = - 1 * (65536 - ZZ):
ZZ = ZY
50040 POKE ZZ,17:
POKE ZZ + 1,LS:
POKE ZZ + 2,MS:
POKE ZZ + 3,33:
POKE ZZ + 4,0
50050 POKE ZZ + 5,60:
POKE ZZ + 6,1:
POKE ZZ + 7,0:
POKE ZZ + 8,4
50060 POKE ZZ + 9,237:
POKE ZZ + 10,176:
POKE ZZ + 11,201
50070 MS = INT (ZZ / 256):
LS = ZZ - MS * 256
50080 IF PEEK (16396) = 201
THEN POKE 16526,LS:
POKE 16527,MS

```

Computer Yahtzee

```

THEN Z = 40:
      GOTO 18000 .....
15070 IF X = 1
      THEN X = X + 1:
      GOTO 15010 .....
15080 Z = 0:
      GOTO 18000 .....
-----
15090 ' ::::::::::::::::::::: YAHTZEE
16000 FOR X = 1 TO 6
16010     IF R1 = X
      THEN Z = Z + 1
16020     IF R2 = X
      THEN Z = Z + 1
16030     IF R3 = X
      THEN Z = Z + 1
16040     IF R4 = X
      THEN Z = Z + 1
16050     IF R5 = X
      THEN Z = Z + 1
16060     IF Z = 5
      THEN Z = 50:
      GOTO 18000 .....
16070     Z = 0:
      NEXT X:
      GOTO 18000 .....
-----
16130 ' ::::::::::::::::::::: CHANCE
17000 Z = R1 + R2 + R3 + R4 + R5
18000 IF PEEK (P + 2) < > 128
      THEN 2000 .....
18005 W$ = "###":
      PRINT USING W$;Z;
18010 IF PX = 1 AND P < 15821
      THEN Z1 = Z1 + Z
18020 IF PX = 2 AND P < 15828
      THEN Z2 = Z2 + Z
18030 IF PX = 3 AND P < 15835
      THEN Z3 = Z3 + Z
18040 IF PX = 4 AND P < 15842
      THEN Z4 = Z4 + Z
18050 IF PX = 1 AND P > 15821
      THEN Y1 = Y1 + Z
18060 IF PX = 2 AND P > 15828
      THEN Y2 = Y2 + Z
18070 IF PX = 3 AND P > 15835
      THEN Y3 = Y3 + Z
18080 IF PX = 4 AND P > 15842
      THEN Y4 = Y4 + Z
18090 TT = 0:
      EN = EN + 1:

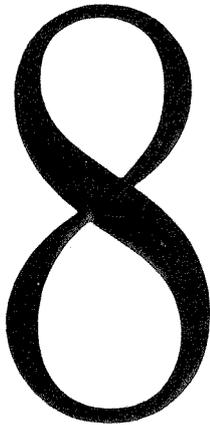
```

Computer Yahtzee

```
50090      DEF USR 0 = ZZ:
           A = USR (0)
50100      RETURN .....
-----
50110      '          BLOCKMOVE TO SCREEN SUBROUTINE

50120      POKE ZZ + 4, PEEK (ZZ + 1):
           POKE ZZ + 5, PEEK (ZZ + 2)
50130      POKE ZZ + 1,0:
           POKE ZZ + 2,60:
           A = USR (0):
           RETURN .....
```

---



### Working With Strings

If you learn to work with string capabilities, you'll gain a better understanding of how to use your computer to its best advantage. Without strings, your computer is just an overgrown calculator.

Let's try some string manipulation. Type in the following lines of code.

```
0 CLEAR 100
10 A$="NOW IS THE TIME FOR ALL GOOD MEN"
```

If we want to pull the word "NOW" out of this string, we can add the following line.

```
20 PRINTLEFT$(A$,3)
```

To pull the word "IS" out, we can use:

```
30 PRINTMID$(A$,5,2)
```

To pull the rest of the line out, we can use:

```
40 PRINTRIGHT$(A$,25)
```

Suppose we want to remove all the spaces from the string. We could use the following code:

```
50 FORX = 1 TO LEN(A$):IFMID$(A$,X,1) = " " THENNEXTELSEPRINT
MID$(A$,X,1);:NEXT:PRINT
```

We can even print the line *backwards* by using the following code:

```
60 FORX = LEN(A$) TO 1 STEP -1:PRINTMID$(A$,X,1);:NEXT:PRINT
```

We can change the composition of the string with this line:

```
70 PRINTRIGHT$(A$,25);:PRINTMID$(A$,4,4);:PRINTLEFT$(A$,3)
```

Now type in the following line:

```
80 B$=" " " :MID$(A$,12,4)=B$:PRINTA$
```

You can do a great deal using strings. The best way to learn is by playing with them, just as we did in the lines above. Keep trying different things, you can't hurt anything and you never know what you might learn!

The following one-liner prints a calendar for any year you wish. The variable (S) equals the print position for the first day of the year. You'll have to adjust M\$ yourself for leap years. When you type this line in, you may have to go into the editing mode to make it all fit!

```
50000
M$="JAN31FEB28MAR31APR30MAY31JUN30JUL31AUG31SEP30OCT31
NOV30DEC31":D$="SUN MON TUE WED THR FRI SAT":S=20:FORX
=1TOLEN(M$)STEP5:PRINTMID$(M$,X,3):PRINTD$:FORZ=1TOVAL
(MID$(M$,X+3,2)):IFS=24THENPRINTTAB(S)Z:S=0:NEXTZ,XELSE
PRINTTAB(S)Z::S=S+4:NEXTZ:PRINT:NEXTX
```

One nice feature about string packing is that you don't have to clear any string space in your program because the graphics are actually contained in the program lines and not in memory reserved for strings.

What we are actually doing is defining a string to be a certain length in our program. This string reserves that area of memory for us to use as we wish. By POKEing a memory space with graphic codes, we replace it with graphics characters which the computer converts into keywords it can understand.

This is why the listing looks so strange. If we wanted to make the effort, we could unpack these lines with the following code:

```
FOR X=1 TO LEN(A$):PRINT ASC(MID$(A$,X,1)):NEXT X
```

The variable A\$ can be any string you wish to unpack.

An explanation of the GRAPHICS PACKER utility program (Auto/Graphics Compiler) follows. This utility program will be used for other games in the book, and we recommend that you type it in before going much further.

### AUTO/GRAPHICS Compiler

Stop typing A\$=CHR\$(191)+CHR\$(178)+ETC+ETC+ETC! It's time we started writing programs that let our computers do all the work for us, — or at least *some* of the more time-consuming work in programming.

With AUTO/GRAPHICS, all you need to do is draw the picture you want (a background for your game program, for example). You don't have to worry about how big the picture is when it's time to save it; the computer will convert your picture into packed string\$ in about 30 seconds.

## How to Use AUTO/GRAPHICS

When you load the program and type RUN, the name will appear on the screen. Press the (enter) key to begin.

We start out in the move mode, with the following controls available to us:

(D)	= GOTO DRAW MODE
(E)	= ERASE MODE
(M)	= MOVE MODE
(S)	= SAVE PICTURE
(CLEAR KEY)	= CLEAR SCREEN
(ENTER)	= GOTO PRINT MODE

In the drawing mode, we have the following extra options:

(F)	= GOTO FINE LINE MODE
(SPACE BAR)	= ERASE OR MOVE MODE

In the print mode, the cursor will stop flashing and you can enter any text you wish to your graphics. To exit from this mode, press enter and you will return to the move mode.

If you press the (S) key while in the move mode, the computer will convert your picture into string\$. After this is done, the screen will clear and say READY. To see your new program, type LIST (enter). You may now save this picture on tape. If you want to see the picture again, add the lines below and run.

---

**Figure 8.1** *Additional Program Lines*

```
1060 CLS:PRINT A1$;A2$;A3$;A4$;A5$;A6$;
1070 GOTO 1070
```

---

Remember not to edit these lines. If you decide to add or make changes to the program, you'll have to adjust line 800 to give the new starting position in memory. To find that information, enter the lines below.

---

**Figure 8.2** *Additional Program Lines*

```
2000 A=17129
2010 B=PEEK(A)
2020 PRINT A;B;CHR$(B)
2030 IFPEEK(14400)=8THEN A=A+1:GOTO2010
2040 IFPEEK(14400)=16THEN A=A-1:GOTO2010
2050 GOTO 2030
```

---

## Program Explanation

### Program Explanation

Lines 10 and 20 clear the screen and convert over to the 32-character print mode.

Line 40 starts our inkey\$ strobe. As soon as any character is found, it lets our program continue. The name will remain on the screen until the operator starts pushing buttons. This method is fine if you have used the program before, but it can be a little confusing otherwise.

Line 50 clears the screen and initializes (A) for our screen starting position. When you begin using PEEKS and POKES, you have to keep the variables under complete control at all times or your program might self-destruct.

Line 60 is where we first PEEK location (A), enabling us to move around the screen, run over our drawings and then replace those locations with the values that were originally there.

Line 70 may look a little strange, but it is very important. This gives the last position of our cursor and puts us back on the screen if we are about to run off.

Lines 80 to 190 check the keyboard for a control key that would route to a different area of the program and also move the cursor in the direction we want. Lines 200 to 530 also repeat this function.

Line 710 is a fast way to convert POKE positions into PRINT positions. Take the POKE position minus 15360 and we obtain the same screen position in PRINT @ values.

When you see A1\$="XXXXXXXXXXXXXXXXXXXXX," use the address of the first X for the value of A in line 800; then you can delete lines 2000 to 2050 and you're ready to go.

Figure 8.3 Graphics Packer (AUTOGRAF) Program

```

10 CLS
20 PRINT CHR$(23)
30 PRINT @456,"AUTO/GRAPHICS COMPILER"
40 A$ = INKEY$ :
   IF A$ = ""
   THEN 40 .....
50 CLS :
   A = 15360
60 B = PEEK (A):
   POKE A,191:
   FOR X = 1 TO 5:
   NEXT :
   POKE A,B
70 C = A
80 IF PEEK (14400) = 1 GOSUB 700
100 IF PEEK (14400) = 8
   THEN A = A - 64
110 IF PEEK (14400) = 16
   THEN A = A + 64
120 IF PEEK (14400) = 32
   THEN A = A - 1
130 IF PEEK (14400) = 64
   THEN A = A + 1
140 IF PEEK (14337) = 16
   THEN GOSUB 200
150 IF PEEK (14340) = 8
   THEN GOTO 800 .....
160 IF A < 15360 OR A > 16382
   THEN A = C
170 IF PEEK (14400) = 2
   THEN CLS
180 B = PEEK (A):
   POKE A,128:
   POKE A,B
190 GOTO 60

-----
200 L = 191
210 POKE A,L
220 C = A
230 IF PEEK (14400) = 8
   THEN A = A - 64
240 IF PEEK (14400) = 16
   THEN A = A + 64
250 IF PEEK (14400) = 32
   THEN A = A - 1
260 IF PEEK (14400) = 64
   THEN A = A + 1
270 IF PEEK (14338) = 32
   THEN RETURN .....
280 IF PEEK (14337) = 32

```

Program Explanation

```

    THEN L = 128
290 IF PEEK (14337) = 16
    THEN L = 191
370 IF A < 15360 OR A > 16382
    THEN A = C
380 IF PEEK (14337) = 64
    THEN GOTO 400 .....
390 GOTO 210
-----
400 X = 0:
    Y = 0
410 IF PEEK (14400) = 8
    THEN Y = Y - 1
420 IF PEEK (14400) = 16
    THEN Y = Y + 1
430 IF PEEK (14400) = 32
    THEN X = X - 1
440 IF PEEK (14400) = 64
    THEN X = X + 1
450 IF PEEK (14400) = 136 RESET (X,Y):
    Y = Y - 1
460 IF PEEK (14400) = 144 RESET (X,Y):
    Y = Y + 1
470 IF PEEK (14400) = 160 RESET (X,Y):
    X = X - 1
480 IF PEEK (14400) = 192 RESET (X,Y):
    X = X + 1
490 IF X < 0 OR X > 127
    THEN X = X1
500 IF Y < 0 OR Y > 47
    THEN Y = Y1
510 SET (X,Y):
    X1 = X:
    Y1 = Y
520 IF PEEK (14338) = 32 GOTO 60 .....
530 GOTO 410
-----
700 REM ***** PRINT ROUTINE *****
710 I = A - 15360:
    PRINT @I + 1, " ";
720 A$ = INKEY$:
    IF A$ = ""
    THEN 720 .....
730 IF A$ = CHR$ (13) RETURN .....
740 PRINT A$;:
    A = A + 1
750 GOTO 720
-----
800 A = 18383
810 FOR X = 15360 TO 16382
815 IF X > 15375 PRINT @0, "SAVING DATA ";

```

```

820      B = PEEK (X)
830      C = PEEK (A):
        IF C < > 88
        THEN A = A + 1:
            GOTO 830 .....
840      IF C = 88 POKE A,B:
            A = A + 1:
            NEXT X:
            CLS
850      DELETE 10 - 850
1000     A1$ = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXX"
1010     A2$ = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXX"
1020     A3$ = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXX"
1030     A4$ = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXX"
1040     A5$ = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXX"
1050     A6$ = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXX"

```

---

## DATA/GRAPHICS Compiler

This program works the same way as AUTO/GRAPHICS, but it converts text and graphics on the screen into data lines. Best of all, you can edit them if needed!

I have to confess that I cheated a little on this listing. I wrote a short program to enter all the data lines. Don't feel bad though, — you only have to type them in once, provided you remember to save the program before running it.



```

200 DATA XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX
210 DATA XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX
220 DATA XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX
230 DATA XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX
240 DATA XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX
250 DATA XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX
260 DATA XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX
270 DATA XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX,XXX
280 DATA XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX,XXX
,XXX,XXX,XXX,XXX
1000 CLS
1010 PRINT CHR$(23)
1020 PRINT @456,"DATA/GRAPHICS COMPILER"
1030 A$ = INKEY$:
IF A$ = ""
THEN 1030 .....
1040 CLS :
A = 15360
1050 B = PEEK (A) :

```

Data/Graphics Compiler

```

        POKE A,191:
        FOR X = 1 TO 5:
        NEXT :
        POKE A,B
1060 C = A
1070 IF PEEK (14400) = 1 GOSUB 1450
1080 IF PEEK (14400) = 8
        THEN A = A - 64
1090 IF PEEK (14400) = 16
        THEN A = A + 64
1100 IF PEEK (14400) = 32
        THEN A = A - 1
1110 IF PEEK (14400) = 64
        THEN A = A + 1
1120 IF PEEK (14337) = 16
        THEN GOSUB 1180
1130 IF PEEK (14340) = 8
        THEN GOTO 1510 .....
1140 IF A < 15360 OR A > 16383
        THEN A = C
1150 IF PEEK (14400) = 2
        THEN CLS
1160 B = PEEK (A):
        POKE A,128:
        POKE A,B
1170 GOTO 1050
-----
1180 L = 191
1190 POKE A,L
1200 C = A
1210 IF PEEK (14400) = 8
        THEN A = A - 64
1220 IF PEEK (14400) = 16
        THEN A = A + 64
1230 IF PEEK (14400) = 32
        THEN A = A - 1
1240 IF PEEK (14400) = 64
        THEN A = A + 1
1250 IF PEEK (14338) = 32
        THEN RETURN .....
1260 IF PEEK (14337) = 32
        THEN L = 128
1270 IF PEEK (14337) = 16
        THEN L = 191
1280 IF A < 15360 OR A > 16383
        THEN A = C
1290 IF PEEK (14337) = 64
        THEN GOTO 1310 .....
1300 GOTO 1190
-----
1310 X = 0:

```

```

Y = 0
1320 IF PEEK (14400) = 8
    THEN Y = Y - 1
1330 IF PEEK (14400) = 16
    THEN Y = Y + 1
1340 IF PEEK (14400) = 32
    THEN X = X - 1
1350 IF PEEK (14400) = 64
    THEN X = X + 1
1360 IF PEEK (14400) = 136 RESET (X,Y):
    Y = Y - 1
1370 IF PEEK (14400) = 144 RESET (X,Y):
    Y = Y + 1
1380 IF PEEK (14400) = 160 RESET (X,Y):
    X = X - 1
1390 IF PEEK (14400) = 192 RESET (X,Y):
    X = X + 1
1400 IF X < 0 OR X > 127
    THEN X = X1
1410 IF Y < 0 OR Y > 47
    THEN Y = Y1
1420 SET (X,Y):
    X1 = X:
    Y1 = Y
1430 IF PEEK (14338) = 32 GOTO 1050 .....
1440 GOTO 1320
-----
1450 REM ***** PRINT ROUTINE *****
1460 I = A - 15360:
    PRINT @I + 1, " ";
1470 A$ = INKEY$:
    IF A$ = " "
    THEN 1470 .....
1480 IF A$ = CHR$ (13) RETURN .....
1490 PRINT A$;:
    A = A + 1
1500 GOTO 1470
-----
1510 S = 15360:
    M = 17129:
    X = 0
1520 A = PEEK (S):
    B = PEEK (S + 1):
    C = PEEK (S + 2):
    D = PEEK (S + 3):
    E = PEEK (S + 4)
1530 IF A < 128 PEEK PEEK PEEK PEEK PEEK PEEK PEEK PEEK
    PEEK PEEK PEEK PEEK PEEK PEEK PEEK PEEK PEEK PEEK
    PEEK PEEK PEEK PEEK PEEK PEEK PEEK PEEK PEEK PEEK
    PEEK PEEK PEEK PEEK PEEK PEEK PEEK PEEK PEEK PEEK

```





# NOTES

---

# 9

---

## **MAKE A MAZE**

This program is called **MAKE A MAZE**. This time, instead of just showing you the listing of the program to type into your computer, I'm going to show you all the steps that I go through to write a game program. Let's work on this one together from start to finish.

**MAKE A MAZE** is a game for two players. The idea is for the first player to create a path around the maze for his piece to travel. This is done by pressing the arrow keys for the direction you want to go.

The program stores all these moves in an array for later use. The second player then presses **ENTER** to start the game. The object for the second player is to move his piece around the maze, hitting the stars for points and at the same time trying to avoid the first player's pieces.

When the game ends, you can either press **(S)** to repeat the same maze game or press **(N)** to draw a new maze.

## **Load GRAPHICS PACKER**

The first thing I want you to do is load in the **GRAPHICS PACKER** program. If you haven't already typed this program in, this would be a good time to do so.

After we have loaded this drawing program, we're going to make a couple of changes to make it easier for you to follow me. First, list the program to make sure it has loaded correctly; then add the following line to the program:

```
525 PRINT@70,"X=";X;"Y=";Y;:PRINT@70," 12 spaces ";
```

This new line will help us to draw exactly the same picture on the screen. After the new line has been entered, type "**RUN 1000**" <**ENTER**> in the command mode. The screen should now show the **READY** sign.

### Finding the Starting Address

By running the program starting from line 1000, we let the computer find the new starting address of the lines that will contain our packed strings. Remember, by adding a new line to the program, we have moved the rest of the lines higher up in memory!

Now that the computer knows the new starting address of A1\$ in line 1000, we have to find a way of getting this information from the computer. Here comes VARPTR to the rescue.

In the command mode, I want you to type this line:

```
A = VARPTR(A1$):PRINTPEEK(A + 2)*256 + PEEK(A + 1) <press enter>.
```

We're doing this not only to find the new address for A1\$ but also to make the GRAPHICS PACKER program work for you if you're using a tape or disk system. The reason for this is that the user memory starts at a different address in disk BASIC than it does in a system without disk.

Write down the number you see on your screen and change line 800 to read:

```
800 A=(number on your screen)
```

Before we go any further, this would be a good time for you to save this program again, so that you won't have to make these changes the next time you need to use it.

### Running GRAPHICS PACKER

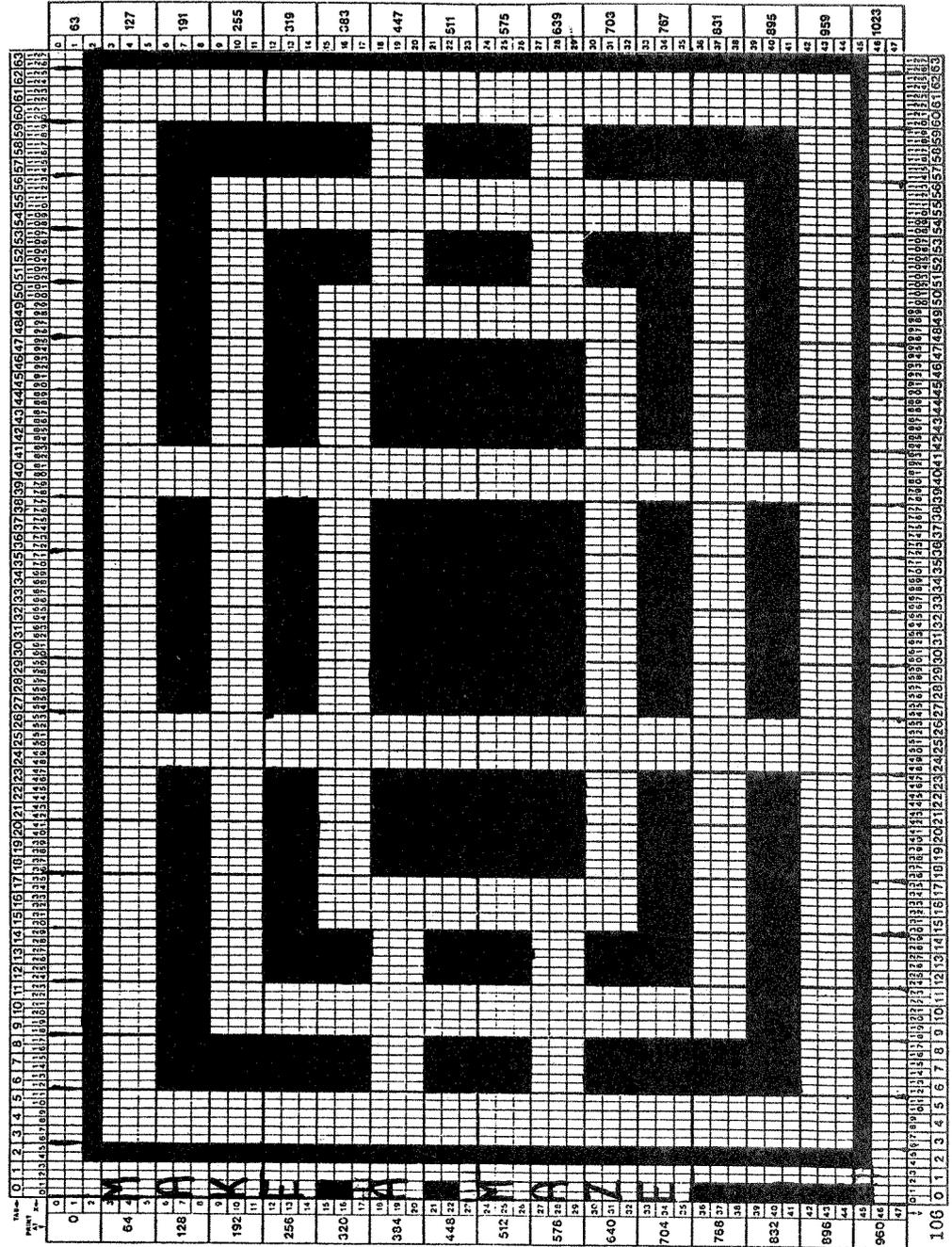
Now we should be ready to run the GRAPHICS PACKER program. When the program starts, you will see the name in the center of the screen. Press the space bar to begin.

You should see a large flashing cursor in the top left corner of the screen, — this is called the move mode of the program. Next, press the (D) key; the cursor should stop flashing and remain on the screen.

Now press the (F) key. This will put us in the fine line drawing mode. To test this out, press the down arrow and draw a line from the top to the bottom of the screen. To erase this line, press the up arrow and space bar at the same time.

With the help of the X,Y coordinates displayed at the top-left corner of your screen and the video display worksheet, I want you to duplicate exactly the MAKE A MAZE picture you see on the worksheet. It's very important that you draw your picture exactly like mine, so take your time and do it right the first time.

Figure 9.1 MAKE A MAZE Worksheet



---

### Moving Around the Screen

Remember, to move around on the screen, you press the arrow key for the direction you want to go and the space bar at the same time.

After you have drawn the picture, press the (M) key to go into the main control mode. You should see the large flashing cursor faintly on the screen.

Now press the (S) key to save your graphics and convert them into packed strings. After a few moments, the screen will clear and you should see the READY sign.

### Testing the Packed String\$

We have now converted our picture into packed strings. Let's add the following lines and run the program so we can test to see if our picture looks the way it should:

```
3060 CLS:PRINTA1$;A2$;A3$;A4$;A5$;A6$;:POKE16383,131
3070 GOTO 3070
```

Now run the program. Wow, it works! If you don't have a nice picture of a MAZE on your screen, start over again. If everything looks okay, delete line 3070 and save this much of the program so that if we mess things up farther down the line, we won't have to draw our picture all over again.

Let's add our sound routine using the string packing method. Add the following lines for our sound:

---

Figure 9.2 *Sound Routine*

```
100 CLS:DIMM(500)
110 M$="XXXXXXXXXXXXXXXXXXXXXXXXXXXX"
120 I=VARPTR(M$):M=PEEK(I+1)+256*PEEK(I+2):Z=M
130 READD:POKEM,D:M=M+1:IFD=201THEN140ELSE130
140 DEFUSR0=Z
150 DATA 205,127,10,77,68,62,1,105,211,255,45,32,253,60
      105,211,255,45,32,253,13,16,238,175,211,255,201
```

---

Note: For tape systems, change line 140 to read:

```
140 POKE16526,PEEK(I+1):POKE16527,PEEK(I+2)
```

Also, each time you see the USR call A=USR1(NUMBER), leave out the number 1 so it looks like this: A=USR(NUMBER).

### Testing the Sound

Now run the program again, and enter the following test in the command mode.

```
FORX=1000TO2000STEP10:A=USR1(X):NEXT
FORX=1000TO2000STEP10:A=USR(X):NEXT (TAPE USERS)
```

Before you do this, hook up your sound box to the computer. You should hear some sounds.

Now if all systems are go, it's time to save our program again, — just in case!

### Program Control

Moving right along, we can now get into the program control section. We have about 99% of our graphics out of the way, and we didn't even have to write one line of code to do it! That's the easy way to program graphics.

Now enter lines 3070 TO 3100 from the listing at the end of this chapter (you don't have to enter the remark lines).

The following is an explanation of what each line does:

- 3070 Define first player's piece.
- 3080 Define second player's piece.
- 3090 Define string for clearing pieces.
- 3100 Set up starting locations for players.

Now we add lines 3110 to 3170 for the first player to create a maze.

- 3110 Print first player's piece.
- 3120 Check if he wants to move up.
- 3130 Check for move down.
- 3140 Check for move left.
- 3150 Check for move right.
- 3160 If space bar pressed, all done.
- 3170 Make sure they don't go out of maze.

The following lines (3210 and 3220) save all moves in array M.

- 3210 If player is pressing a key, save move.
- 3220 Go back for more.

Now we are ready to start the game, and the second player takes over from here! Add lines 4000 to 4035.

- 4000 Print both players on screen.
- 4010 Wait for second player to press enter before starting.
- 4020 Start array loop for first player's pieces.
- 4025 Repeat loop if needed.
- 4030 Print first player's piece number 1 and make noise.
- 4035 Print first player's piece number 2.

Now add lines 4040 to 4050 to check for players' pieces hitting each other.

- 4040 If array position 1 is in same location as player 2,
- 4045 or if reverse array position 2 is same, then end.
- 4050 Print player number 1.

Lines 4060 to 4090 check whether player 2 makes any moves. Let's add these lines to our program.

## Program Control

4060 Check for move up. If yes, then remove old position.  
4070 Check for move down.  
4080 Check for move left.  
4090 Check for move right.

The following lines place and remove the stars from the playing field. Add lines 4095 to 4120.

4095 If player hits star, then increment point counter.  
4100 Keep player in maze.  
4110 Random road block and point stars.  
4115 Random set and reset road blocks.  
4120 End of array loop.

Now we'll insert the last lines for our game. Lines 5000 to 5040 check whether the player wants to play the same game again or start a new game.

5000 Sound end of game noise.  
5010 Start INKEY loop.  
5020 If player hits (S), then play same game.  
5030 If player hits (N), then play new game.  
5040 End of INKEY loop.

This game was a lot of fun to write. I hope it gives you a better idea of how I go about writing a graphics game.

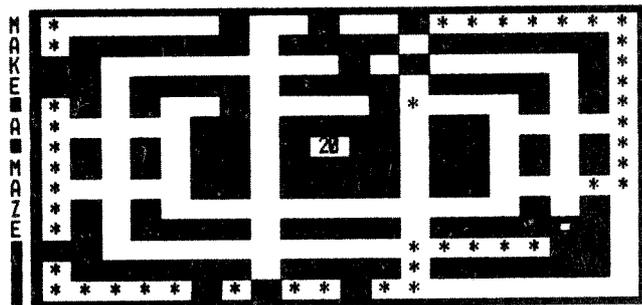


Figure 9.3 MAKE A MAZE Program

```

100 CLS :
    DIM M(500)
110 M$ = "XXXXXXXXXXXXXXXXXXXXXXXXXXXX"
120 I = VARPTR (M$):
    M = PEEK (I + 1) + 256 * PEEK (I + 2):
    Z = M
130 READ D:
    POKE M,D:
    M = M + 1:
    IF D = 201
    THEN 140: .....
    ELSE 130 .....
-----
140 DEF USR 0 = Z
150 DATA 205,127,10,77,68,62,1,105,211,255,45,32,253,60,
    105,211,255,45,32,253,13,16,238,175,211,255,201
995 REM =====
        PACKED STRINGS FOR MAZE GAME
        =====
1000 A1$ = " END END DEF DEF
    DEF DEF DEF DEF DEF DEF DEF DEF DEF DEF DEF DEF DEF DEF DEF
    DEF DEF DEF DEF DEF DEF DEF DEF DEF DEF DEF DEF DEF DEF DEF
    DEF DEF DEF DEF DEF DEF DEF DEF DEF DEF DEF DEF DEF DEF DEF
    DEF M END USING END END END END END END END END END END
    END END END END END END END END END END END END END END
    END END END END END END END END END END END END END END
    END END END END END END END END END END END END END END
    END END END END END END END END END END END END END USING
    A END USING END END END USING USING USING USING USING
    USING USING USING USING USING USING USING USING USING USING
    USING USING USING USING END END END USING USING USING
    USING USING USING USING USING USING USING USING USING USING
    END END END USING USING USING USING USING USING USING USING
    USING USING USING USING USING USING USING USING USING USING
    USING USING END END END USING "
1010 A2$ = "K END USING END END END USING USING USING END
    END END END END END END END END END END END END END END
    END END END END END END END END END END END END END END
    END END END END END END END END END END END USING USING END
    END END USING E END USING END END END USING USING USING
    END END END USING USING USING USING USING USING USING
    USING USING USING USING USING END END END USING USING
    USING USING USING USING USING USING USING USING USING
    USING END END END USING USING USING USING USING USING
    USING USING USING USING USING USING USING END END END USING
    USING USING END END END USING IF END USING END END
    END USING USING USING END END END USING USING USING
    END END

```



```

        USING USING USING USING END END END USING USING USING
        USING USING USING USING USING USING USING USING USING
        USING USING USING USING USING USING END END END USING
        USING END USING END END END END END END END END END
        END END END END END END END END END END END END END
        END END END END END END END END END END END END END
        END END END END END END END END END END END END END
        END END END END END END END END END END END END USING
        "
1050  A6$ = " SET END SET SET SET SET SET SET SET SET SET SET
        SET SET SET SET SET SET SET SET SET SET SET SET SET
        SET SET SET SET SET SET SET SET SET SET SET SET SET
        SET SET SET SET SET SET SET SET SET SET SET SET SET
        SET SET SET SET SET SET SET SET SET SET SET SET SET
        "
3060  CLS :
        PRINT A1$;A2$;A3$;A4$;A5$;A6$;:
        POKE 16383,131
3065  REM =====
        DEFINE STRING$ FOR GAME PIECES
        =====
3070  P1$ = CHR$ (191) + CHR$ (191) + CHR$ (191)
3080  P2$ = CHR$ (191) + CHR$ (179) + CHR$ (191)
3090  C$ = CHR$ (128) + CHR$ (128) + CHR$ (128)
3100  P1 = 67:
        P2 = 956:
        E = 67
3105  REM =====
        FIRST PLAYER MAKES A MAZE
        =====
3110  PRINT @P1,P1$;:
        FOR Q = 1 TO 10:
        NEXT Q:
        PRINT @P1,C$;:
        Z = P1
3120  IF PEEK (14400) = 8
        THEN P1 = P1 - 64
3130  IF PEEK (14400) = 16
        THEN P1 = P1 + 64
3140  IF PEEK (14400) = 32
        THEN P1 = P1 - 3
3150  IF PEEK (14400) = 64
        THEN P1 = P1 + 3
3160  IF PEEK (14400) = 1
        THEN 4000: .....
        ELSE IF I = 1
        THEN 4000 .....
3170  IF PEEK (15360 + P1) < > 128
        THEN P1 = Z
3185  REM =====
        SAVE MOVES IN ARRAY (M)

```

```

=====
3210 IF PEEK (14591) > 0
    THEN M(T) = P1:
        T = T + 1:
        IF T = 500
            THEN 4000 .....
-----
3220 GOTO 3110
-----
3225 REM =====
                START OF GAME / SECOND PLAYER
                =====
4000 PRINT @P2,P2$;:
    PRINT @P1,P1$;
4010 IF PEEK (14400) < > 1
    THEN 4010 .....
4020 FOR X = 1 TO T:
    IF M(X) < 67
        THEN M(X) = 67
4025 O = T - X:
    IF O = 0
        THEN O = T
4030 PRINT @M(X),P1$;:
    A = USR 0(1111)
4035 PRINT @M(O),P1$;
4038 REM =====
                CHECK IF PIECES HIT EACH OTHER
                =====
4040 IF M(X) = P2
    THEN 5000 .....
4045 IF M(O) = P2
    THEN 5000 .....
4050 PRINT @P2,P2$;:
    Z = P2
4055 REM =====
                CHECK IF PLAYER MOVES AROUND
                =====
4060 IF PEEK (14400) = 8
    THEN PRINT @P2,C$;:
        P2 = P2 - 64
4070 IF PEEK (14400) = 16
    THEN PRINT @P2,C$;:
        P2 = P2 + 64
4080 IF PEEK (14400) = 32
    THEN PRINT @P2,C$;:
        P2 = P2 - 3
4090 IF PEEK (14400) = 64
    THEN PRINT @P2,C$;:
        P2 = P2 + 3
4092 REM =====
                PRINT & REMOVE POINT STARS

```

```

=====
4095     IF   PEEK (15361 + P2) = 42A = USR 0(1919):
           H = H + 10:
           PRINT @478,H;:
           GOTO 4110 .....
4100     IF   PEEK (15360 + P2) < > 128
           THEN P2 = Z
4110     R = RND (10):
           IF   R = 5
           THEN PRINT @M(X),P1$;:
                A = USR 0(2222):
           ELSE PRINT @M(X)," * ";
4115     R = RND (10):
           IF   R = 5
           THEN PRINT @M(O),P1$;:
           ELSE PRINT @M(O), STRING$ (3,128);
4120 NEXT X:
           GOTO 4020
-----
4125 REM  =====
           PLAY AGAIN ? SAME OR NEW GAMES
           =====
5000 A = USR 0(0)
5010 I$ = INKEY$
5020 IF   I$ = "S"
           THEN I = 1:
                H = 0:
                GOTO 3060 .....
5030 IF   I$ = "N"
           THEN RUN .....
5040 GOTO 5010
-----

```

---

### Program Explanation

Lines 10 to 80 are the instructions.  
 Lines 90 to 110 get input from the players.  
 Lines 140 to 180 LISA also does math problems for you.  
 Lines 200 to 250 are the string reader.  
 Lines 260 to 590 are questions and answers.

### LISA

This is my own version of the ELIZA program that is so popular. Instead of being a head doctor, this program acts more like a friend. The program breaks the player's input down into single words. It then looks for a match for these words in its own data. I kept this listing short for the simple reason that the game is more fun when you add your own data lines. You could put in some information about some of your own friends.

Figure 9.4 LISA Program

```

10 REM *** LISA
20 REM *** WRITTEN BY TOM DEMPSEY
30 CLS :
  PRINT "LISA WILL TALK TO YOU : WHEN YOU ASK A QUESTI
  ON"
40 PRINT "ALWAYS PUT A SPACE AND A PERIOD AT THE END OF
  YOUR QUESTION.."
50 PRINT "IF YOU WANT LISA TO ASK YOU A QUESTION TYPE,
  ASK ME A QUESTION ."
60 INPUT "HIT ENTER TO PLAY";A$
70 CLS
80 CLEAR 100
90 INPUT "WHAT IS YOUR NAME";N$
100 X = 1:
  PRINT " "
110 INPUT "QUESTION";A$
120 IF A$ = "WHAT IS MY NAME ." PRINT "YOUR NAME IS ";
  N$:
  GOTO 100 .....
130 IF A$ = "DO YOU KNOW MY NAME ." PRINT "YES":
  GOTO 100 .....
140 IF A$ = "ASK ME A QUESTION ." GOTO 410 .....
150 IF A$ = "+":
  INPUT A,B:
  PRINT A + B:
  GOTO 100 .....
160 IF A$ = "/":
  INPUT A,B:
  PRINT A / B:
  GOTO 100 .....
170 IF A$ = "*":
  INPUT A,B:
  PRINT A * B:
  GOTO 100 .....
180 IF A$ = "-":
  INPUT A,B:
  PRINT A - B:
  GOTO 100 .....
190 IF A$ = "GOOD BYE" PRINT "GOOD BYE ";N$:
  END .....
200 D = LEN (A$)
210 B$ = MID$ (A$,X,1)
220 X = X + 1:
  IF X > D
  THEN 100 .....
230 IF ASC (B$) = 32 GOTO 260 .....
240 LET D$ = D$ + B$
250 GOTO 210
-----
260 IF D$ = "WHAT" PRINT "IT'S ";

```

```

270 IF D$ = "MEMORY" PRINT MEM :
      GOTO 100 .....
280 IF D$ = "NAME" PRINT "LISA ";
290 IF D$ = "OLD" PRINT "3 MONTHS ";
300 IF D$ = "AGE" PRINT "3 MONTHS ";
310 IF D$ = "DO" AND X < 5
      THEN R = RND (2):
           IF R = 1 PRINT "YES ";;
           ELSE PRINT "NO ";
320 IF D$ = "TIME" PRINT "TIME ";
330 IF D$ = "ARE" AND X < 6 PRINT "YES ";
340 IF D$ = "COST" PRINT "TOO MUCH ";N$
350 IF D$ = "FEEL" PRINT "O.K.";
360 IF D$ = "HELL" PRINT "PLEASE DON'T USE WORDS LIKE
      THAT ";N$
370 IF D$ = "LEVEL" PRINT "LEVEL 2 ";
380 IF D$ = "EAT" PRINT "FOOD "
390 IF D$ = "FOOD" PRINT "CALLED ELECTRICITY ";N$
400 D$ = "":
      B$ = MID$ (W$,1):
      GOTO 210

-----
410 R = RND (10)
420 IF R = 1 PRINT "HOW OLD ARE YOU ";N$
430 IF R = 2 PRINT "DO YOU HAVE ANY HOBBIES"
440 IF R = 3 PRINT "DO YOU LIKE GIRLS"
450 IF R = 4 PRINT "DO YOU LIKE BOYS"
460 IF R = 5 PRINT "DO YOU LIKE COMPUTERS ";N$
470 IF R = 6 PRINT "WHAT IS YOUR FAVORITE SPORT ";N$
480 IF R = 7 PRINT "WHAT TOWN DO YOU LIVE IN ";N$
490 IF R = 8 PRINT "DO YOU LIKE TO FOOL AROUND ";N$
500 IF R = 9 PRINT "DO YOU PLAY GOLF ";N$
510 IF R = 10 PRINT "DO YOU THINK MY RESPONSES ARE GOOD
      D ";N$
520 INPUT A$
530 IF A$ = "YES" PRINT "THATS GOOD":
      GOTO 560 .....
540 IF A$ = "NO" PRINT "THATS TOO BAD":
      GOTO 560 .....
550 PRINT "THATS NICE"
560 PRINT "WANT ME TO ASK MORE QUESTIONS";
570 INPUT A$
580 IF A$ = "YES" GOTO 410 .....
590 GOTO 100
-----

```

# NOTES

---

# 10

---

## Graphic Art and Computers

This chapter contains programs that make use of the video screen as a canvas. It gives examples of what can be done using low resolution.

### 1932 FORD ROADSTER

The 1932 FORD ROADSTER is not a game, but it is a good example of what can be done with graphics on the TRS-80 computer. The program starts by printing a 1932 FORD ROADSTER on the screen and then transfers the screen contents to your line printer. Afterwards, the program lists itself.

The original version of this program allowed you to press the up arrow key after printing the car on the screen, causing the wheels to turn around, the horn to beep and the radiator to boil over, — shooting steam up to the top of the screen.

There's an unlimited number of things you can do with what first appears to be a stationary object. All you need is a little imagination and a lot of time. This car took three 14-hour days to complete.

I think the effort was worth it. It was published in Interface Age Magazine in February, 1981. I included it in this book because I think it's a good example of what can be done with a low resolution screen.

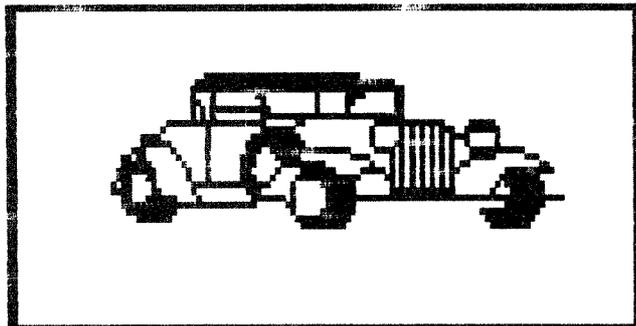


Figure 10.1 Ford Roadster Program

```

1 REM *** 1932 FORD ROADSTER ***
2 REM *** WRITTEN BY TOM DEMPSEY ; APRIL 10,1980
3 LPRINT CHR$ (31)"1932 FORD ROADSTER"
5 CLEAR 1000
6 REM *** SET UP CHR$ ***
10 A$ = CHR$ (176) + STRING$ (15,188) + CHR$ (180) + STRING$
    (3,176)
20 B$ = CHR$ (170) + CHR$ (147) + CHR$ (171) + STRING$
    (4,131) + CHR$ (175) + STRING$ (5,131) + CHR$ (151) +
    STRING$ (2,131) + CHR$ (191) + CHR$ (143) + STRING$
    (3,131) + CHR$ (191)
30 C$ = STRING$ (2,176) + STRING$ (3,186) + STRING$ (4,
    179) + CHR$ (187) + CHR$ (141) + STRING$ (2,140) + CHR$
    (188) + CHR$ (140) + CHR$ (141) + STRING$ (2,140) +
    CHR$ (141) + STRING$ (2,140) + STRING$ (2,188) + CHR$
    (191) + STRING$ (4,176) + STRING$ (2,128)
40 D$ = CHR$ (160) + STRING$ (2,176)
50 E$ = CHR$ (168) + CHR$ (188) + CHR$ (182) + CHR$ (14
    4) + STRING$ (3,128) + CHR$ (149) + STRING$ (3,128) +
    CHR$ (184) + CHR$ (190) + STRING$ (3,191) + CHR$ (18
    0) + STRING$ (6,128) + CHR$ (170) + STRING$ (2,128) +
    CHR$ (186) + STRING$ (4,170) + CHR$ (171) + CHR$ (14
    0) + CHR$ (151)
60 F$ = STRING$ (2,131) + CHR$ (149)
70 G$ = CHR$ (168) + CHR$ (182) + CHR$ (191) + CHR$ (14
    4) + CHR$ (128) + CHR$ (130) + CHR$ (164) + STRING$
    (2,128) + CHR$ (149) + STRING$ (2,128) + CHR$ (168) +
    CHR$ (191) + CHR$ (129) + CHR$ (130) + CHR$ (175) +
    CHR$ (191) + CHR$ (183) + STRING$ (2,179) + STRING$
    (2,131)
80 H$ = CHR$ (137) + CHR$ (140) + CHR$ (164) + CHR$ (13
    1) + CHR$ (171) + STRING$ (6,170) + CHR$ (160) + CHR$
    (154) + STRING$ (2,143) + STRING$ (3,131) + CHR$ (14
    0) + CHR$ (164) + CHR$ (144)
90 I$ = CHR$ (160) + CHR$ (175) + CHR$ (151) + CHR$ (13
    1) + CHR$ (175) + STRING$ (3,128) + CHR$ (165) + CHR$
    (176) + CHR$ (181) + STRING$ (2,176) + CHR$ (186) +
    CHR$ (189) + CHR$ (176) + CHR$ (152) + CHR$ (129) +
    CHR$ (160) + CHR$ (142) + CHR$ (143) + CHR$ (175) +
    CHR$ (191) + STRING$ (2,188)
100 J$ = CHR$ (134) + STRING$ (2,131) + CHR$ (174) + STRING$
    (6,170) + CHR$ (129) + STRING$ (3,128) + CHR$ (152) +
    STRING$ (3,191) + CHR$ (183) + CHR$ (131)
110 K$ = CHR$ (130) + CHR$ (171) + CHR$ (181) + CHR$ (17
    6) + CHR$ (190) + CHR$ (189) + CHR$ (188) + CHR$ (17
    6) + CHR$ (178) + CHR$ (180) + STRING$ (4,176) + CHR$
    (187) + CHR$ (141) + STRING$ (2,140) + CHR$ (186) +
    STRING$ (3,128) + STRING$ (3,191) + STRING$ (3,140) +
    CHR$ (142)
120 L$ = STRING$ (6,143) + STRING$ (3,140) + CHR$ (142) +

```

```

CHR$ (140) + CHR$ (190) + STRING$ (3,191) + STRING$
(2,140)
130 M$ = CHR$ (131) + STRING$ (3,143) + CHR$ (135) + STRING$
(11,128) + CHR$ (130) + CHR$ (172) + CHR$ (188) + CHR$
(190) + STRING$ (2,191) + CHR$ (135) + STRING$ (12,1
28) + CHR$ (130) + CHR$ (175) + STRING$ (3,191) + CHR$
(159) + CHR$ (129)
140 CLS
145 REM *** PRINT PICTURE ***
150 PRINT @211,A$;
160 PRINT @274,B$;
170 PRINT @336,C$;
180 PRINT @366,D$;
190 PRINT @397,E$;
200 PRINT @431,F$;
210 PRINT @459,G$;
220 PRINT @482,H$;
230 PRINT @522,I$;
240 PRINT @547,J$;
250 PRINT @586,K$;
260 PRINT @615,L$;
270 PRINT @652,M$;
275 REM *** SUBROUTINE SCAN SCREEN AND LPRINT ***
280 CLEAR 1000:
      LPRINT CHR$ (29):
      LPRINT CHR$ (27); CHR$ (56)
290 FOR Y = 0 TO 47
300   P$ = ""
310   FOR X = 0 TO 127
320     IF POINT (X,Y)
          THEN P$ = P$ + CHR$ (191):
          ELSE P$ = P$ + " "
330   NEXT X
340   LPRINT P$
350 NEXT Y
355 REM *** ADJ. PRINT SIZE & PRINT LISTING ***
360 LPRINT CHR$ (30):
      LPRINT CHR$ (27); CHR$ (54):
      LPRINT CHR$ (27); CHR$ (66):
      LLIST

```

---

## VIDEOHEX

This program is not actually a game, but it does give an example of the different types of things that can be done with graphics and your computer. What this program lacks in speed, it makes up for in grace.

What does it do? It converts graphics on the screen into data lines, — not your *usual* data lines, but graphic codes in hex.

To use the program, type RUN, press ENTER and sit back to watch. After the program converts the screen into hex data lines, it clears the screen and deletes line 1000. You may now run the program again to watch it read the hex codes and draw the graphics back on the screen.

This program is strictly for people who like hex numbers. As for any practical uses, I haven't found them yet, — it's just too slow.

Figure 10.2 VIDEOHEX Program

```

1000 CLS :
      CLEAR 1000:
      ' BY TOM DEMPSEY
1010 PRINT @0,"BASIC HEX FILE ZAP V1.0 <> PRESS (M) TO MO
      DIFY AND ENTER TO SAVE"; STRING$ (64,143);
1020 PRINT @896,"FILE, RELATIVE SECTOR, BYTE ";;
      INPUT F$,V$,K$
1025 B$ = V$:
      GOSUB 1300:
      V = B + 1:
      B$ = K$:
      GOSUB 1300:
      K = B + 1
1030 OPEN "R",1,F$
1040 FIELD 1,255ASS$
1050 CH$ = S$
1060 GET 1,V
1070 CH$ = S$:
      CLOSE
1075 REM =====
           CONVERT DECIMAL NUMBERS TO HEX
           =====
1080 H$ = "0123456789ABCDEF":
      PRINT @960,"ONE MOMENT PLEASE ";
1090 S(1) = 4096:
      S(2) = 256:
      S(3) = 16:
      S(4) = 1:
      Z = K + 20:
      IF Z > 255
      THEN Z = 255
1100 FOR X = K TO Z:
      N = ASC ( MID$ ( S$,X,1)):
      X$ = "":
      FOR J = 3 TO 4:
          P = S(J)
1110          FOR I = 1 TO 16:
              IF N - I * P < 0
              THEN X$ = X$ + MID$ (H$,I,1):
                  N = N - (I - 1) * P:
              NEXT J:
              ELSE NEXT I
1120          M$ = M$ + X$ + " ":
              PRINT ". ";;
          NEXT X
1130          PRINT @960,M$;
1135          REM =====
                   MOVE CURSOR MODE TO BEGIN EDIT
                   =====
1140          M = 16320

```

```

1150      P = PEEK (M) :
          Q = PEEK (14400) :
          IF   PEEK (14338) = 320$ = INKEY$ :
              GOTO 1200 .....
1160      POKE M,143:
          FOR T = 1 TO 10:
          NEXT :
          POKE M,P:
          FOR T = 1 TO 10:
          NEXT
1170      IF   Q = 32
          THEN M = M - 3:
          ELSE IF   Q = 64
              THEN M = M + 3
1180      IF   M < 16320
          THEN M = 16320:
          ELSE IF   M > 16381
              THEN M = 16380
1190      GOTO 1150 .....
-----
1195      REM =====
          EDITING MODE INSERT NEW HEX #S
          =====
1200      I$ = INKEY$ :
          P = PEEK (M)
1210      POKE M,128:
          FOR T = 1 TO 10:
          NEXT :
          POKE M,P:
          FOR T = 1 TO 10:
          NEXT
1220      IF   I$ = ""
          THEN 1200: .....
          ELSE IF   I$ = CHR$ (13)
              THEN OPEN "R",1,F$:
                  GOTO 1270 .....
1230      IF   ASC (I$) < 48
          THEN 1200: .....
          ELSE IF   ASC (I$) > 70
              THEN 1200 .....
1240      IF   ASC (I$) > 57 AND ASC (I$) < 65
          THEN 1200 .....
1250      PRINT @M - 15360,I$;:
          M = M + 1:
          IF   PEEK (M) = 32M = M + 1
1260      GOTO 1200 .....
-----
1265      REM =====
          SCANS SCREEN & WRITES NEW FILE
          =====
1270      FOR X = 16320 TO 16380 STEP 3:

```

## Program Explanation

```

                                P1 = PEEK (X) :
                                P2 = PEEK (X + 1) :
                                B$ = CHR$ (P1) + CHR$ (P2) :
                                GOSUB 1300:
                                C$ = C$ + CHR$ (B) :
                                NEXT X
1280   MID$ (CH$,K,Z) = C$:
                                LSET S$ = CH$:
                                PUT 1,V:
                                CLOSE :
                                PRINT :
                                PRINT
1290   M$ = "":
                                S$ = "":
                                C$ = "":
                                GOTO 1010 .....
-----
1295   REM  =====
                                CONVERT HEX TO DECIMAL NUMBERS
                                =====
1300   J = 1
1310   FOR I = 1 TO 2:
                                D(I) = 0:
                                D(I) = ASC ( MID$ (B$,J,1)) - 48
1320   IF D(I) > 9
                                THEN D(I) = D(I) - 7
1330   J = J + 1:
                                NEXT I:
                                B = 4096 * D(1) + 256 * D(2) + 16 * D(3) +
                                D(4):
                                B = B / 256
1340   RETURN .....
```

---

### Program Explanation

Lines 1000 to 1600 are dummy data lines. I think it's time programmers started calling dummy data and dummy string\$ by a better name. How about memory reserve lines? That is what they really do.

Lines 1710 to 1760 are the drawing subroutine. They are put here only for demonstration; you can place any drawing program here that you wish. In lines 1770 to 1940 we PEEK the screen locations, convert the decimal values into X\$, then convert the string back into decimal to POKE into our memory reserve lines.

Lines 1610 to 1700 read the hex data lines and convert them back into decimal before POKEing them back into the screen positions.

---

---

# 11

---

## Educational Games

MATH TEACHER is a fun game to try to outsmart. If you can make it to the last section, good luck. As you play, the questions get harder.

The overall program is several small programs which are all strung together. In operation, it adds two random numbers and then asks you for the correct answer, which it already knows.

This shows yet another way to make use of random numbers, — letting the random number generator create math problems for us instead of programming each problem into the listing. This technique can give you long-running programs from short listings.

```
HOW MUCH IS 23 + 24 =? 47
```

```
HOW MUCH IS 23 + 24 =? 47  
*** YOU ARE RIGHT 47
```

```
HOW MUCH IS 35 + 5 =? 39  
WRONG ANSWER 35 + 5 = 40
```

Figure 11.1 *Math Teacher Program*

```

1 CLS
2 PRINT "MATH PROBLEMS"
3 FOR X = 1 TO 1000
4 NEXT X
9 LET C = 0
10 CLS
20 N = RND (50)
30 LET N = N
40 A = RND (50)
50 LET A = A
60 PRINT "HOW MUCH IS ";N;"+";A;"=";
70 INPUT B
80 IF B = N + A
    THEN 100 .....
90 PRINT "WRONG ANSWER";N;"+";A;"=";N + A
92 FOR X = 1 TO 5000
93 NEXT X
94 GOTO 10

-----
100 PRINT "*** YOU ARE RIGHT";N + A
110 LET C = C + 1
120 FOR X = 1 TO 1000
130 NEXT X
140 IF C = 10
    THEN 200 .....
150 GOTO 10

-----
200 PRINT "I THINK YOU'RE READY FOR SOME HARDER QUESTION
    S!"
210 FOR X = 1 TO 3000
220 NEXT X
230 LET C = 0
240 CLS
250 N = RND (10)
260 LET N = N
270 A = RND (10)
280 LET A = A
290 PRINT "HOW MUCH DOES";N;"X";A;"=";
300 INPUT B
310 IF B = N * A
    THEN 350 .....
320 PRINT "NO YOU'RE WRONG";N;"X";A;"=";N * A
330 FOR X = 1 TO 5000
340 NEXT X
345 GOTO 240

-----
350 PRINT "YOU ARE RIGHT";N * A
355 FOR X = 1 TO 1000
356 NEXT X
360 LET C = C + 1

```

```

370 IF C = 10
    THEN 400 .....
380 GOTO 240
-----
400 PRINT "THINK YOU'RE SMART, TRY THESE"
410 FOR X = 1 TO 3000
415 NEXT X
420 LET C = 0
430 CLS
440 N = RND (100)
450 LET N = N
460 A = RND (100)
470 LET A = A
480 PRINT "HOW MUCH IS";N;"X";A;"=";
490 INPUT B
500 IF B = N * A
    THEN 550 .....
510 PRINT "WRONG";N;"X";A;"=";N * A
520 FOR X = 1 TO 5000
530 NEXT X
540 GOTO 420
-----
550 PRINT "YOU'RE RIGHT";N * A
560 LET C = C + 1
570 FOR X = 1 TO 1000
580 NEXT X
590 IF C = 10
    THEN 700 .....
600 GOTO 430
-----
700 PRINT "YOU'RE DOING BETTER THAN I EXPECTED, TRY THESE!"
701 PRINT "PLEASE GET A PEN AND PAPER! WHEN READY, ENTER #1"
702 INPUT O
703 IF O = 1
    THEN 740 .....
710 FOR X = 1 TO 1000
720 NEXT X
740 CLS
750 N = RND (100)
760 LET N = N
770 A = RND (100)
780 LET A = A
790 D = RND (10)
800 LET D = D
810 S = RND (10)
820 LET S = S
830 P = RND (100)
840 LET P = P
850 PRINT "ARE YOU READY FOR THIS! YOU HAVE 60 SECONDS T

```

### Suggested Changes

```
      O ANSWER"  
880 PRINT N;"X";A;"/";D;"-";S;"+";P;"=";  
910 FOR X = 1 TO 30000  
920 NEXT X  
945 PRINT "THE ANSWER IS";N * A / D - S + P  
950 FOR X = 1 TO 1000  
951 NEXT X  
952 GOTO 740
```

---

---

### MORSE CODE

MORSE CODE is great for learning the morse code. After typing RUN <ENTER>, the screen goes blank, and each time you type a letter, it is displayed on the screen in morse code. Send a secret message to your friends.

### SUGGESTED CHANGES

Why not add sound so you can hear the code as you type it? A dot can be OUT255,0:OUT255,2 and a dash can be FORX = 1TO2:OUT255,0:OUT255,2:NEXTX.

In operation, this program makes use of the complete keyboard for control keys, and the graphics are created with the PRINT command.

If you press the letter (E), the computer prints a dot, which represents the morse code symbol for the letter E. We could just as easily have the computer print any letter or symbol we want when the typist hits different letters.

Figure 11.2 Morse Code Program

```

10 CLS
20 PRINT "THIS PROGRAM TURNS YOUR KEYBOARD INTO A"
30 PRINT "MORSE CODE PRINTER...WHEN THE SCREEN IS CLEAR"
40 PRINT "IT'S READY TO USE. JUST START TYPING."
50 PRINT :
PRINT "WHEN READY HIT ENTER....."
60 INPUT B$
70 CLS
80 PRINT CHR$(23)
90 AS$ = INKEY$ :
IF AS$ = "A" PRINT ".-";
100 IF AS$ = "B" PRINT "-.-.";
110 IF AS$ = "C" PRINT "-.-.";
120 IF AS$ = "D" PRINT "-.-.";
130 IF AS$ = "E" PRINT ".";
140 IF AS$ = "F" PRINT ".-";
150 IF AS$ = "G" PRINT ".-";
160 IF AS$ = "H" PRINT ".-";
170 IF AS$ = "I" PRINT ".-";
180 IF AS$ = "J" PRINT ".-";
190 IF AS$ = "K" PRINT ".-";
200 IF AS$ = "L" PRINT ".-";
210 IF AS$ = "M" PRINT "-.-";
220 IF AS$ = "N" PRINT "-.-";
230 IF AS$ = "O" PRINT "-.-";
240 IF AS$ = "P" PRINT ".-";
250 IF AS$ = "Q" PRINT ".-";
260 IF AS$ = "R" PRINT ".-";
270 IF AS$ = "S" PRINT ".-";
280 IF AS$ = "T" PRINT "-";
290 IF AS$ = "U" PRINT ".-";
300 IF AS$ = "V" PRINT ".-";
310 IF AS$ = "W" PRINT "-.-";
320 IF AS$ = "X" PRINT "-.-";
330 IF AS$ = "Y" PRINT "-.-";
340 IF AS$ = "Z" PRINT "-.-";
350 IF AS$ = " " PRINT " ";
360 GOTO 90

```

## SILVER WEIGHT CONTENT

This one is for all the coin collectors out there! The program does all the work for you. It tells you how much silver is in your coins and what their dollar value is at the current market price!

The following explains the program operation.

Line 20 gets the present silver value.

Lines 30 to 180 multiply the number of coins by silver content.

Lines 190 to 310 print out the totals.

Figure 11.3 Silver Program

```

10 CLS
20 INPUT "ENTER PRESENT VALUE OF SILVER FOR 1 OZ.";I:
CLS
30 PRINT "HOW MANY SILVER DOLLARS"
40 INPUT S
50 LET S = S * .77

```

## Memory Magic

```
60 PRINT "HOW MANY HALF DOLLARS"
70 INPUT H
80 LET H = H * .385
90 PRINT "HOW MANY QUARTERS"
100 INPUT Q
110 LET Q = Q * .1925
120 PRINT "HOW MANY DIMES"
130 INPUT D
140 LET D = D * .077
150 PRINT "HOW MANY 1942-1945 NICKELS"
160 INPUT N
170 LET N = N * .0385
180 CLS
190 PRINT "SILVER WEIGHT CONTENT"
200 PRINT
210 PRINT "DOLLARS=";S;" OZ."
220 PRINT "HALF DOLLARS=";H;" OZ."
230 PRINT "QUARTERS=";Q;" OZ."
240 PRINT "DIMES=";D;" OZ."
250 PRINT "NICKELS=";N;" OZ."
260 PRINT
    PRINT "TOTAL WEIGHT IN SILVER=";S + H + Q + D + N;"
    OZ."
270 PRINT "TO CONTINUE HIT ENTER"
280 INPUT AS$
290 PRINT "TOTAL DOLLAR VALUE IN SILVER IS...$";(S + H +
    Q + D + N) * I
300 PRINT
310 END
```

---

## MEMORY MAGIC

MEMORY MAGIC is a memory game for 1 or more players. The game starts by showing you a 6 by 8 grid and filling up random squares on the grid. This game has 3 levels of play, so it's good for all ages. The game is a very good method of increasing your memory retention and becomes very competitive between two or more players.

After a few seconds delay, the filled in squares are cleared, and the object is to replace them in the correct order. After you have entered the total number that you think were originally on the screen, the computer shows you where the original squares were and prints out the percentage of correct answers you had in the game.

The first time my wife sat down and tried this game, she selected the expert mode. Then she played and got 100% correct on the first try. I didn't have the courage to tell her that I was still trying to get past the beginner mode. It's very embarrassing for a programmer to lose at a game he wrote himself!

The following is an explanation of MEMORY MAGIC.

- Lines 10 to 15 clear screen, print title.
- Lines 20 to 250 set up the playing board.
- Lines 260 to 460 get players' input.
- Lines 465 to 490 check for match.
- Lines 500 to 530 keep scores.
- Lines 600 to 680 show original board; ask to play again.

Figure 11.4 Memory Magic Program

```

10 CLS :
   DIM B(48)
15 PRINT CHR$( 23) :
   PRINT @448,"MEMORY MAGIC BY TOM DEMPSEY":
   FOR X = 1 TO 1000:
     NEXT :
   CLS
16 PRINT "ENTER LEVEL OF PLAY":
   PRINT :
   PRINT "1 = BEGINNER":
   PRINT "2 = ADVANCED":
   PRINT "3 = EXPERT"
17 INPUT S:
   IF S = 1S = 5:
   ELSE IF S = 2S = 4:
     ELSE IF S = 3S = 2
18 CLS
20 FOR A = 2 TO 38 STEP 6
30   FOR M = 0 TO 113:
     SET (M,A):
     NEXT
40 NEXT A
50 FOR A = 0 TO 113 STEP 14
60   FOR M = 2 TO 38:
     SET (A,M):
     SET (A + 1,M):
     NEXT
70 NEXT A
75 B = 1
80 FOR A = 121 TO 824 STEP 128
90   PRINT @A,B;:
     B = B + 1:
     NEXT A
100 X = 65
110 FOR A = 835 TO 885 STEP 7
120   POKE 15360 + A,X:
     X = X + 1:
     NEXT A
130 DATA 65,72,79,86,93,100,107,114
140 DATA 193,200,207,214,221,228,235,242
150 DATA 321,328,335,342,349,356,363,370
160 DATA 449,456,463,470,477,484,491,498
170 DATA 577,584,591,598,605,612,619,626
180 DATA 705,712,719,726,733,740,747,754
190 A$ = STRING$( 6,191):
   B$ = STRING$( 6,128):
   C$ = STRING$( 6,176)
200 FOR X = 1 TO 48:
   R = RND (S)
210   READ D:

```

## Memory Magic

```

                IF R = 1 PRINT @D,A$;:
                PRINT @D + 64,A$;
220             IF R = 1B(N) = D:
                N = N + 1
230 NEXT X
240 FOR X = 1 TO 10000:
    NEXT
250 RESTORE
260 FOR X = 1 TO 48:
    READ D:
    PRINT @D,B$;:
    PRINT @D + 64,C$;:
    NEXT
270 PRINT @896,"                ";:
    PRINT @896,"NUMBER ";
280 I$ = INKEY$ :
    IF I$ = ""
    THEN 280 .....
281 IF PEEK (14400) = 2 AND K = 0 PRINT @T,B$;:
    PRINT @T + 64,C$;:
    TT = TT - 1:
    K = 1:
    QQ = QQ - 1:
    GOTO 270 .....
285 IF I$ < CHR$ (49) OR I$ > CHR$ (54)
    THEN 270 .....
290 IF I$ = "1" T = 65
300 IF I$ = "2" T = 193
310 IF I$ = "3" T = 321
320 IF I$ = "4" T = 449
330 IF I$ = "5" T = 577
340 IF I$ = "6" T = 705
350 PRINT I$;
360 PRINT @910,"                ";:
    PRINT @910,"LETTER ";
370 I$ = INKEY$ :
    IF I$ = ""
    THEN 370 .....
375 IF I$ < CHR$ (65) OR I$ > CHR$ (72)
    THEN 360 .....
380 IF I$ = "A" T = T
390 IF I$ = "B" T = T + 7
400 IF I$ = "C" T = T + 14
410 IF I$ = "D" T = T + 21
420 IF I$ = "E" T = T + 28
430 IF I$ = "F" T = T + 35
440 IF I$ = "G" T = T + 42
450 IF I$ = "H" T = T + 49
460 PRINT I$;
470 TT = TT + 1
480 FOR Q = 0 TO N - 1

```

```

490     IF    T = B(Q)
        THEN QQ = QQ + 1
495 NEXT Q
500 PRINT @T,A$;:
      PRINT @T + 64,A$;:
      Q = 0:
      K = 0:
      IF    TT = N
        THEN 600: .....
        ELSE 270 .....
-----
600 REM  *** SEE BOARD ***
610 PRINT @896,"HIT ENTER TO COMPARE BOARD";
620 L$ = INKEY$ :
      IF    L$ < > CHR$ (13)
        THEN 620 .....
630 FOR  P = 0 TO N - 1:
      PRINT @B(P),"*";:
      NEXT
640 PRINT @896,QQ;"OUT OF ";N;"CORRECT OR ";:
      PRINT USING "##.##";QQ / N;:
      PRINT "% PLAY AGAIN (Y/N)";
650 I$ = INKEY$ :
      IF    I$ = ""
        THEN 650 .....
660 IF    I$ = "N"
        THEN CLS :
          END .....
670 IF    I$ = "Y" RUN .....
680 GOTO 650
-----

```

# NOTES

---

# 12

---

## Puzzle Games

TREASURE HUNT is played on a 9 by 5 grid. The idea is to find the gold by digging holes. Each time you play the game the treasure is hidden in a different location. It's harder than it looks. The best score I know of so far is 7 tries! Can you think of the fastest way to find the treasure?

In TREASURE HUNT we use our random number to hide the treasure on the grid. This type of game is good for display on the screen. You might want to let two players play at the same time, and the first to find the gold wins.

The following lines contain an explanation of TREASURE HUNT.

Lines 10 to 70 print the rules.

Lines 90 to 540 convert RND to grid locations.

Lines 550 to 730 set up grid lines.

Lines 740 to 1190 convert player's input to grid locations.

Lines 1200 to 1270 dig the holes and check for wins.

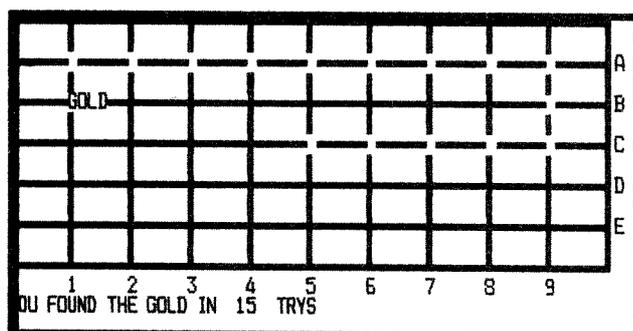


Figure 12.1 *Treasure Hunt Program*

```

10 CLS :
20 PRINT "THIS GAME IS CALLED TREASURE HUNT"
30 PRINT "THE OBJECT IS TO FIND THE GOLD..ENTER THE POS
   ITIONS"
40 PRINT "YOU HAVE TO DIG YOUR HOLE LIKE THIS (C6). SEE
   HOW"
50 PRINT "MANY HOLES YOU HAVE TO DIG TO FIND THE GOLD."
60 PRINT "YOU CANNOT MOVE MORE THAN 1 POSITION IN ANY "
70 PRINT "DIRECTION IN A TURN...."
80 PRINT "TO PLAY HIT ENTER";:
90 INPUT A$
100 CLS
110 A = 1:
120 S = 0
130 T = RND (44)
140 IF T = 1 LET T = 134
150 IF T = 2 LET T = 140
160 IF T = 3 LET T = 146
170 IF T = 4 LET T = 152
180 IF T = 5 LET T = 158
190 IF T = 6 LET T = 164
200 IF T = 7 LET T = 170
210 IF T = 8 LET T = 176
220 IF T = 9 LET T = 182
230 IF T = 10 LET T = 262
240 IF T = 11 LET T = 268
250 IF T = 12 LET T = 274
260 IF T = 13 LET T = 280
270 IF T = 14 LET T = 286
280 IF T = 15 LET T = 292
290 IF T = 16 LET T = 298
300 IF T = 17 LET T = 304
310 IF T = 18 LET T = 310
320 IF T = 19 LET T = 390
330 IF T = 20 LET T = 396
340 IF T = 21 LET T = 402
350 IF T = 22 LET T = 408
360 IF T = 23 LET T = 420
370 IF T = 24 LET T = 426
380 IF T = 25 LET T = 432
390 IF T = 26 LET T = 438
400 IF T = 27 LET T = 518
410 IF T = 28 LET T = 524
420 IF T = 29 LET T = 530
430 IF T = 30 LET T = 536
440 IF T = 31 LET T = 542
450 IF T = 32 LET T = 548
460 IF T = 33 LET T = 554
470 IF T = 34 LET T = 560
480 IF T = 35 LET T = 566
490 IF T = 36 LET T = 646
500 IF T = 37 LET T = 652
510 IF T = 38 LET T = 658
520 IF T = 39 LET T = 664
530 IF T = 40 LET T = 670
540 IF T = 41 LET T = 676
550 IF T = 42 LET T = 682
560 IF T = 43 LET T = 688
570 IF T = 44 LET T = 694
580 FOR M = 0 TO 120:
590 SET (M,A):
600 NEXT
610 IF A = 37
620 THEN 580 .....
630 A = A + 6:
640 GOTO 550

```

```

580 A = 0
590 FOR M = 1 TO 37:
      SET (A,M):
      NEXT M
600 A = A + 12
610 IF A > 120
      THEN GOTO 630 .....
620 GOTO 590 .....

630 A = 1:
      B = 837
640 PRINT @B, A;
650 A = A + 1:
      B = B + 6
660 IF B > 886
      THEN GOTO 680 .....
670 GOTO 640 .....

680 PRINT @189, "A";
690 PRINT @317, "B";
700 PRINT @445, "C";
710 PRINT @573, "D";
720 PRINT @701, "E";
730 B = 414:
      PRINT @B, " ";
740 PRINT @896, "MOVE";
      INPUT AS$ .....

750 IF AS$ = "A1" LET A = 134
760 IF AS$ = "A2" LET A = 140
770 IF AS$ = "A3" LET A = 146
780 IF AS$ = "A4" LET A = 152
790 IF AS$ = "A5" LET A = 158
800 IF AS$ = "A6" LET A = 164
810 IF AS$ = "A7" LET A = 170
820 IF AS$ = "A8" LET A = 176
830 IF AS$ = "A9" LET A = 182
840 IF AS$ = "B1" LET A = 262
850 IF AS$ = "B2" LET A = 268
860 IF AS$ = "B3" LET A = 274
870 IF AS$ = "B4" LET A = 280
880 IF AS$ = "B5" LET A = 286
890 IF AS$ = "B6" LET A = 292
900 IF AS$ = "B7" LET A = 298
910 IF AS$ = "B8" LET A = 304
920 IF AS$ = "B9" LET A = 310
930 IF AS$ = "C1" LET A = 390
940 IF AS$ = "C2" LET A = 396
950 IF AS$ = "C3" LET A = 402
960 IF AS$ = "C4" LET A = 408
970 IF AS$ = "C5" LET A = 414
980 IF AS$ = "C6" LET A = 420
990 IF AS$ = "C7" LET A = 426
1000 IF AS$ = "C8" LET A = 432
1010 IF AS$ = "C9" LET A = 438
1020 IF AS$ = "D1" LET A = 518
1030 IF AS$ = "D2" LET A = 524
1040 IF AS$ = "D3" LET A = 530
1050 IF AS$ = "D4" LET A = 536
1060 IF AS$ = "D5" LET A = 542
1070 IF AS$ = "D6" LET A = 548
1080 IF AS$ = "D7" LET A = 554
1090 IF AS$ = "D8" LET A = 560
1100 IF AS$ = "D9" LET A = 566
1110 IF AS$ = "E1" LET A = 646
1120 IF AS$ = "E2" LET A = 652
1130 IF AS$ = "E3" LET A = 658
1140 IF AS$ = "E4" LET A = 664
1150 IF AS$ = "E5" LET A = 670
1160 IF AS$ = "E6" LET A = 676
1170 IF AS$ = "E7" LET A = 682
1180 IF AS$ = "E8" LET A = 688
1190 IF AS$ = "E9" LET A = 694
1200 IF B - A = 6 OR A - B = 6

```

## Last Straw

```
1210 THEN 1230
1210 IF B - A = 128 OR A - B = 128
1220 THEN 1230
1220 PRINT @896, "INVALID MOVE":
FOR X = 1 TO 500:
NEXT :
GOTO 740
-----
1230 PRINT @A, " ";
1240 LET B = A
1250 S = S + 1
1260 IF T = B PRINT @T, "GOLD";:
PRINT @896, "YOU FOUND THE GOLD IN ";S;" TRYS":
GOTO 1260
1270 GOTO 740
-----
```

---

### LAST STRAW

LAST STRAW is a popular numbers game, but this one uses graphics. The game starts by asking how many straws you would like to start with and the maximum number a player can remove in a turn. It then draws the straws and begins the game.

There is a number combination you can play to beat the computer every time, but I'm not going to tell!

The following is an explanation of LAST STRAW.

Lines 10 to 180 are for player input and rules.

Lines 190 to 230 draw the straws.

Lines 240 to 560 keep the score.

Lines 570 to 680 remove the straws.

Figure 12.2 Last Straw Program

```

10 REM *** LAST STRAW
20 REM *** WRITTEN BY TOM DEMPSEY
30 CLS
40 LET A = 0:
LET B = 0
50 PRINT "THIS GAME IS CALLED LAST STRAW LOSES"
60 PRINT "HOW MANY STRAWS DO YOU WANT TO START WITH (2-
26)."
70 INPUT N
80 IF N > 26
THEN 60 .....
90 IF N < 2 .....
THEN 60 .....
100 PRINT "MAXIMUM NUMBER OF STRAWS WE CAN REMOVE"; .....
110 INPUT K
120 PRINT
130 PRINT "INSTRUCTIONS FOR PLAYING LAST STRAW"
140 PRINT "THE NUMBER OF STRAWS IN THE PILE IS".N
150 PRINT "WE CAN REMOVE FROM 1 TO ";K;" STRAWS IN A TURN"
160 PRINT "THE PLAYER TO REMOVE THE LAST STRAW LOSES."
170 INPUT "WHEN YOU'RE READY TO PLAY, HIT ENTER";A$
180 CLS
190 FOR M = 30 TO 47:
SET (A,M):
NEXT M
200 LET A = A + 5
210 LET B = B + 1
220 IF B = N LET A = 0:
GOTO 240 .....
230 GOTO 190
-----
240 PRINT @0, "YOUR MOVE";
250 INPUT X
260 IF X > K
THEN 290 .....
270 IF X < 1 .....
THEN 290 .....
280 GOTO 570
-----
290 PRINT @0, "ILLEGAL MOVE"
300 PRINT
310 GOTO 240
-----
320 LET N = N - X
330 IF N > 0
THEN 370 .....
340 IF N = 0 .....
THEN 350 .....
350 PRINT "***** COMPUTER WINS *****" .....
360 IF N = 0
THEN 560 .....
370 PRINT
380 LET Q = INT ((N - 1) / (K + 1))
390 LET Y = N - 1 - Q * (K + 1)
400 IF Y = 0
THEN 490 .....
410 LET N = N - Y
420 PRINT "COMPUTER TAKES";Y
430 GOTO 630
-----
440 IF N = 0
THEN 550 .....
450 PRINT "STRAWS LEFT IN PILE-";N .....
460 PRINT
470 PRINT
480 GOTO 240
-----

```

Figure 13.1 Invasion Program

```

10 B = 120:
   C = 192:
   D = 320:
   E = 500
30 CLS :
   INPUT "ENTER NAME,HIGH SCORE OF LAST GAME";W$,O
40 GOTO 450
-----
50 TIME = 250
60 INPUT "ENTER YOUR NAME";N$
70 CLS
80 S$ = CHR$(191)
90 A$ = CHR$(140) + STRING$(3,143) + CHR$(140)
100 B$ = CHR$(176) + STRING$(3,143) + CHR$(176)
110 C$ = "<***>"
120 D$ = CHR$(140) + STRING$(3,191) + CHR$(140)
130 P$ = " USING "
140 P = 960
150 PRINT @P,P$;
160 L = P
170 IF PEEK(14400) = 32
   THEN P = P - 4
180 IF PEEK(14400) = 64
   THEN P = P + 4
190 IF PEEK(14400) = 8 GOSUB 330
200 IF P < 960 P = 960
210 IF P > 1022 P = 1022
220 TIME = TIME - 1:
   PRINT @50,"TIME LEFT";TIME:
   IF TIME = 0 PRINT @470,"E N D O F G A M E":
   FOR X = 1 TO 1000:
     NEXT X
     GOTO 450 .....
230 PRINT @L," ";:
   PRINT @P,P$;
240 PRINT @B,A$;" "
250 B = B - 2:
   IF B = 64 PRINT @B," "
     B = 120
260 PRINT @C," ";B$;
270 C = C + 1:
   IF C = 250 PRINT @C," "
     C = 192
280 PRINT @D," ";C$;
290 D = D + 4:
   IF D > 378 PRINT @D," "
     D = 320
300 PRINT @E,D$;" "
310 E = E - 2:
   IF E = 448 PRINT @E," "
     E = 500
320 GOTO 160 .....
-----
330 A = P - 128
340 PRINT @A,S$;:
   OUT 255,0:
   OUT 255,2:
   PRINT @A," ";
350 IF A = B + 2 OR A = C + 2 OR A = D OR A = E +
   2 GOSUB 380
360 A = A - 128:
   IF A < 64 RETURN .....
370 GOTO 340 .....

```

---

# 13

---

## Arcade Games

### SAUCER INVASION

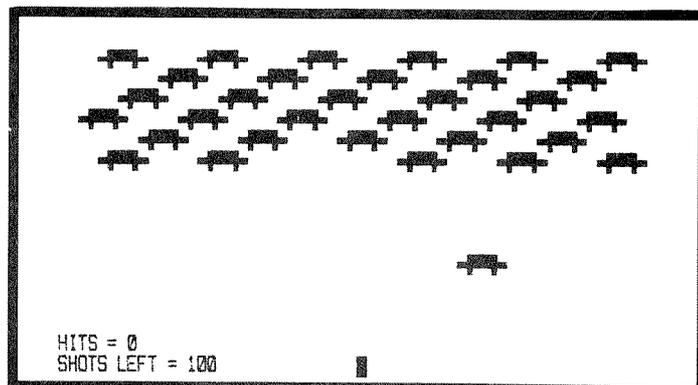
I started writing this game at about five o'clock one morning. I was wondering what a graphics game book would be like without a real-time, arcade-type game in BASIC. A real-time game in BASIC? Did you know that the PRINT statement is faster than the POKE statement for certain conditions?

The game starts by asking for the high score and the player's name. If this is your first time playing, just enter (A,0). The program keeps track of the name and score of the best player after each game.

When you're ready to play, just enter your name. If you beat the last high score, you get your name in lights.

To move your gun, use the left and right arrows, and use the up arrow to fire. I always let my kids test my games because their reflexes are faster than an adult's. They think that most games which seem fast to me are too slow.

They gave me their approval on this game after playing it only once. The only problem was that they wouldn't let me get near the computer for about four hours.



Last Straw

```
490 IF N > 1
    THEN 520 .....
500 LET Y = 1
510 GOTO 410
-----
520 LET Y = 1 + INT (K * RND (1))
530 IF Y > K
    THEN 500 .....
540 GOTO 410
-----
550 PRINT "***** YOU WIN *****"
560 END
-----
570 LET B = 0:
    LET A = A
580 FOR M = 30 TO 47:
    RESET (A,M):
    NEXT M
590 LET A = A + 5
600 LET B = B + 1
610 IF B = X GOTO 320 .....
620 GOTO 580
-----
630 LET B = 0:
    LET A = A
640 FOR M = 30 TO 47:
    RESET (A,M):
    NEXT M
650 LET A = A + 5
660 LET B = B + 1
670 IF B = Y GOTO 440 .....
680 GOTO 640
-----
```

```

380      IF  A = B + 2 FOR  X = 1 TO 10:
          PRINT @B, "P O W":
          PRINT @B, "":
          NEXT
390      IF  A = C + 2 FOR  X = 1 TO 10:
          PRINT @C, "P O W":
          PRINT @C, "":
          NEXT
400      IF  A = D FOR  X = 1 TO 10:
          PRINT @D, "P O W":
          PRINT @D, "":
          NEXT:
410      T = T + 90
          IF  A = E + 2 FOR  X = 1 TO 10:
              PRINT @E, "P O W":
              PRINT @E, "":
              NEXT
420      T = T + 10:
430      PRINT @0, "POINTS = "; T
          FOR  X = 1 TO 10:
              OUT 255, 0:
              OUT 255, 2:
          NEXT
440      RETURN .....
-----
450      CLS :
460      PRINT CHR$ (23)
          IF  T > O
          THEN O = T:
              W$ = N$
470      PRINT @448, "HIGH SCORE"
480      PRINT :
          PRINT W$; " WITH "; O; " POINT
          S"
490      FOR  X = 1 TO 2000:
          NEXT
500      T = 0
510      GOTO 50 .....
-----

```

**Explanation of SAUCER INVASIONS**

- Line 10 sets up the starting values for the saucers.
- Line 20 disables the break key in case a player wants to change his score.
- Line 30 gets high score for score to beat.
- Line 40 GOTO score keeper and print score.
- Line 50 is the timer.
- Line 60 inputs next player's name.
- Lines 80 to 130 set up graphics characters.
- Lines 140 to 210 print controls.
- Line 190 checks if shot is fired.
- Line 220 checks for end of game.
- Lines 230 to 320 advance and print saucers.
- Lines 330 to 340 print shot and sound effects.
- Line 350 checks for hit.
- Line 360 checks if shot is done.
- Lines 380 to 410 show hits on screen.
- Line 420 adds score.
- Line 430 is sound for hits.
- Lines 450 to 510 are score keeper.

### SAUCER WARS

The SAUCER WARS game started out as an experiment for moving shapes in different directions around the screen without getting lost in a maze of code. It ended up becoming a game for my kids. A lot of my games start this way. My kids see them and say, "Gee, that would make a neat game, Dad." Well, you know the rest of the story.

In this game you control your own flying saucer. You move your ship with the arrow keys and fire at the enemy by pressing the space bar. The enemy cannot shoot back at you, but if they hit your ship, the game ends. It's not as easy as it sounds because they're coming at you from all sides!

The listing for this game contains the complete program explanation. The variations of this type game are endless. Just keep in mind that when using BASIC it's best to keep your code as short as possible. This does not mean short programs, but short loops. Don't try to make too many things happen at a time when working with BASIC.

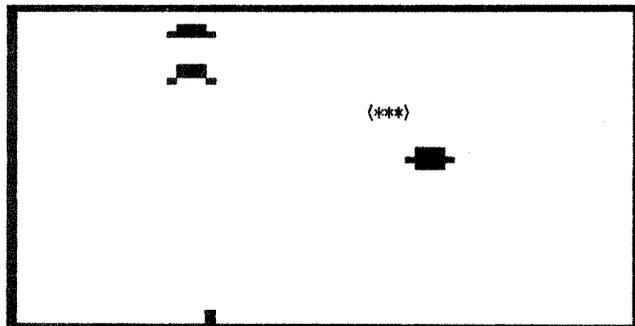
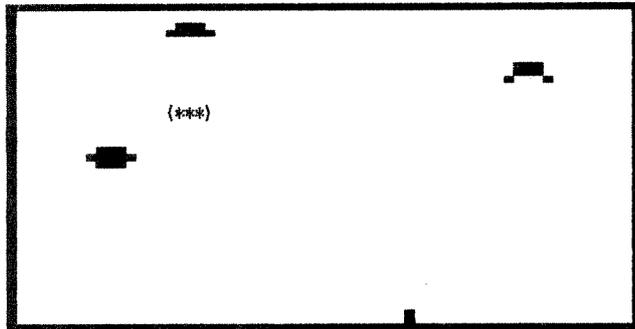


Figure 13.2 Saucer Wars Program

```

10 CLS :
   IF PEEK (16396) < > 201
   THEN PRINT "NOT FOR DISK SYSTEM":
      END
20 DEFINT A - Z:
   CLEAR 100
30 AD = 32739:
   HI = INT (AD / 256):
   POKE 16527,HI:
   POKE 16526,AD - HI * 256
40 FOR I = AD TO AD + 28:
   READ DT:
   POKE I,DT:
   NEXT
50 DATA 205,127,10,62,1,14,0,237,91,61,64,69,47,230,3,1
   79,211,255,13,40,4,16,246,24,242,37,32,241,201
60 CLS
70 PRINT CHR$(23)
80 PRINT @456,"S A U C E R   W A R S"
90 FOR X = 1 TO 1500:
   NEXT
100 PRINT @448,"FUN GAMES WRITTEN BY TOM DEMPSEY"
110 FOR X = 1 TO 2000:
   NEXT :
   CLS
120 PRINT CHR$(23):
   PRINT @448,"DO YOU NEED INSTRUCTIONS (Y/N)";
130 INPUT IS:
   IF IS = "N"
   THEN 190 .....
140 CLS :
   PRINT CHR$(23):
   PRINT "TO MOVE YOUR SHIP USE THE ARROW KEYS"
150 PRINT "USE THE SPACE BAR TO FIRE LASER"
160 PRINT "IF THE ENEMY CRASHES INTO YOUR SHIP THE GAME
   ENDS!"
170 PRINT :
   PRINT "DONT FORGET TO USE SOUND IF YOU HAVE IT"
180 PRINT :
   INPUT "TO CONTINUE HIT ENTER";IS
190 CLS
200 H = RND (1000)
210 T = RND (1000)
220 S = 476
230 REM ***** CREATE PLAYERS SHIP
240 AS = CHR$(176) + CHR$(188) + CHR$(188) + CHR$(18
   8) + CHR$(176)";
250 PRINT @L,"";
260 IF S < 1 OR S > 954 LET S = L
270 L = S
280 PRINT @S,AS;
290 REM ***** CHECK IF PLAYER MOVES SHIP
300 BS = INKEY$
310 IF PEEK (14400) = 8
   THEN S = S - 64:
      GOTO 250 .....
320 IF PEEK (14400) = 16
   THEN S = S + 64:
      GOTO 250 .....
330 IF PEEK (14400) = 32
   THEN S = S - 2:
      GOTO 250 .....
340 IF PEEK (14400) = 64
   THEN S = S + 2:
      GOTO 250 .....
350 REM ***** RELOCATE ENEMY
360 IF H < 10H = RND (1000)
370 IF T < 10T = RND (1000)

```

```

380 REM ***** FIRE LASER
390 IF B$ = " " PRINT @S + 5, ".....
    . . . . . ;
400 PRINT @H, VS;
410 PRINT @T, VS;
420 REM ***** CREATE LASER SOUND
430 IF B$ = " " FOR N = 1 TO 4:
        SS = USR (5115):
        NEXT N
440 REM ***** CREATE SOUND OF ENEMY SHIP
450 OUT 255, 0:
    OUT 255, 2
460 REM ***** REMOVE LASER FIRE
470 IF B$ = " " PRINT @S + 5, "
    ;
480 K = S:
    W = SS + 30:
    U = S + 30:
490 REM ***** CHECK FOR LASER HIT
500 IF B$ = " " AND H > K AND H < W GOTO 910
510 IF B$ = " " AND T > K AND T < U GOTO 1080 .....
520 REM ***** CREATE ENEMY SHIP
530 VS = CHR$ (176) + CHR$ (188) + CHR$ (188) + CHR$
    (188) + CHR$ (176)
540 PRINT @H, VS;
550 PRINT @H, " ";
560 PRINT @T, " ";
570 REM ***** CHECK FOR PLAYERS SHIP DE
    STROYED
580 IF S = H OR S = T GOTO 800 .....
590 IF S = H GOTO 800 .....
600 IF SS = H + 1 GOTO 800 .....
610 IF S = H - 1 GOTO 800 .....
620 IF S = H + 2 GOTO 800 .....
630 IF S = H - 2 GOTO 800 .....
640 IF S = H + 3 GOTO 800 .....
650 IF S = H - 3 GOTO 800 .....
660 IF S = T GOTO 1020 .....
670 IF S = T - 1 GOTO 1020 .....
680 IF S = T + 1 GOTO 1020 .....
690 IF S = T - 2 GOTO 1020 .....
700 IF S = T + 2 GOTO 1020 .....
710 IF S = T - 3 GOTO 1020 .....
720 IF S = T + 3 GOTO 1020 .....
730 IF H < 10 H = RND (1000)
740 IF T < 10 T = RND (1000)
750 REM ***** ADVANCE ENEMY SHIPS
760 H = H - 6
770 T = T - 65
780 GOTO 300 .....
-----
790 REM ***** DESTROY PLAYERS SHIP
800 PRINT @S, CHR$ (176) + " " + CHR$ (188) + " " +
    CHR$ (176)
810 FOR J = 0 TO 60 STEP 5:
        FOR I = 277 TO 267 STEP - 1:
            SS = USR (I + J):
            NEXT I:
        NEXT J
820 FOR X = 1 TO 100:
        NEXT
830 CLS
840 PRINT @S, CHR$ (176) + " " + CHR$ (188) + " "
    + CHR$ (176)
850 FOR N = 1 TO 10:
        SS = USR (5454):
        NEXT N
860 FOR X = 1 TO 100:
        NEXT
870 CLS
880 PRINT @0, "ENEMY SHIPS DESTROYED = "; P
890 INPUT "TO PLAY AGAIN HIT ENTER"; IS:
    GOTO 190 .....

```

```

-----
900    REM ***** DESTROY ENEMY SHIP
910    PRINT @H,"      "
920    P = P + 1:
        IF P = 20 PRINT @448,"MAY THE FORCE BE WITH Y
          OU : YOU KILLED 20 ALIENS":
          END
930    PRINT @H, CHR$(183) + " " + CHR$(155) + " " + .....
        CHR$(166) + " " + CHR$(191)
940    FOR J = 0 TO 60 STEP 5:
          FOR I = 377 TO 367 STEP - 1:
            SS = USR (I + J):
          NEXT I:
        NEXT J
950    FOR X = 1 TO 100:
        NEXT
960    PRINT @H,"      "
970    PRINT @H, CHR$(183) + "      " + CHR$(191)
980    FOR X = 1 TO 100:
        NEXT
990    PRINT @H,"      "
1000   H = RND (1000)
1010   CLS :
        GOTO 250 .....
-----
1020   REM ***** DESTROY PLAYERS SHIP
1030   PRINT @S,"      "
1040   FOR J = 0 TO 60 STEP 5:
          FOR I = 277 TO 267 STEP - 1:
            SS = USR (I + J):
          NEXT I:
        NEXT J
1050   FOR N = 1 TO 10:
          SS = USR (5454):
        NEXT N
1060   PRINT @0,"ENEMY SHIPS DESTROYED = ";P
1070   INPUT "TO PLAY AGAIN HIT ENTER";I$:
        GOTO 190 .....
-----
1080   REM ***** DESTROY 2ND ENEMY SHIP
1090   PRINT @T,"      "
1100   FOR J = 0 TO 60 STEP 5:
          FOR I = 377 TO 367 STEP - 1:
            SS = USR (I + J):
          NEXT I:
        NEXT J
1110   P = P + 1:
        IF P = 20 PRINT @448,"YOU SAVED THE UNIVERSE
          : YOU DESTROYED 20 ALIENS":
          END
1120   PRINT @T, CHR$(183) + " " + CHR$(155) + " " + .....
        CHR$(166) + " " + CHR$(191)
1130   FOR X = 1 TO 100:
        NEXT
1140   PRINT @T,"      "
1150   PRINT @T, CHR$(183) + "      " + CHR$(191)
1160   FOR X = 1 TO 100:
        NEXT
1170   PRINT @T,"      "
1180   T = RND (1000)
1190   GOTO 250 .....
-----

```

### SPEED SHIFTER

This program, like the program TWO LOVERS, uses data lines, but instead of printing graphic characters, it prints letters to make up the instructions for the game.

The animation in SPEED SHIFTER is created by printing the character code (140) to show speed on the speedometer. If the speed drops, it prints at a lower position, which erases the characters in front of it. Data reading can be used for a great many games, such as secret code games or business applications such as the MACHINE SERVICE RECORDS program.

SPEED SHIFTER is unique in the way it prints out instructions for the game by reading data lines containing the codes for the words in the instructions. Wow! Is a sentence that long legal?

This is a popular game with my quality control inspectors (my sons TOM and WALLY). The object of the game is to see how fast you can reach a speed of 90 mph. The problem is that if you don't shift gears at the right speed, you might blow the transmission or first gear.

The up arrow is used for shifting gears, and the right arrow is for the gas. I got the idea for this game from the speedometer in my car. They don't make 'em like they used to! Have you ever tried to get parts for a '67 Continental?

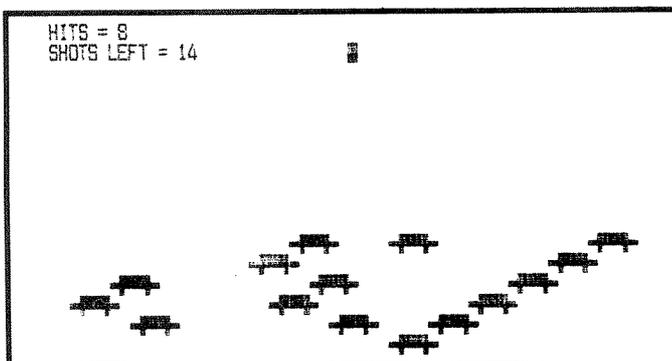
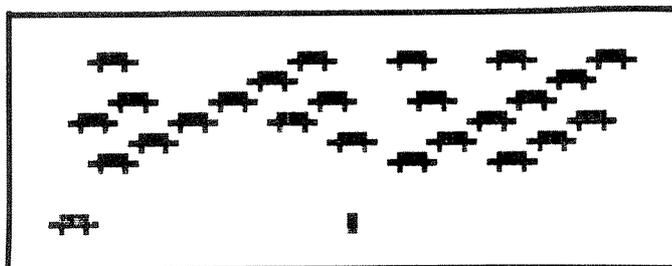


Figure 13.3 Speed Shifter Program

```

100 CLS
110 FOR X = 1 TO 7:
    PRINT :
    NEXT X
120 PRINT CHR$(23)
130 READ A:
    IF A = 46
    THEN 170 ; .....
140 IF A = 1
    THEN 190 ; .....
150 PRINT CHR$(A);
160 GOTO 130
-----
170 FOR X = 1 TO 1000:
    NEXT
180 GOTO 100
-----
190 IS = INKEY$:
    IF IS = "N":
    THEN 190 ; .....
200 X = 1:
    CLS
210 AS = "0 10 20 30 40 50 60
    70 80 90"
220 BS = CHR$(140)
230 S = 896
240 PRINT @832, AS
250 IF S < 896S = 896
260 IF S > 958S = 958
270 PRINT @S, BS
280 PRINT @810, "TIME = ";L;
290 L = L + 1
300 IF S = 958 PRINT @448, "YOU WON THE SPEED SHIFTER R
    ACE IN ";L - 1; "SECONDS":
    END
310 IF P < 30 AND X = 3 PRINT @448, "GOING TOO SLOW FOR
    3RD GEAR...DROPPED YOUR TRANSMISSION":
    END
320 C = S
330 IF P > 26 AND X = 1 PRINT @448, "YOU JUST BLEW YOUR
    ENGINE IN 1ST GEAR":
    END
340 IF P > 48 AND X = 2 PRINT @448, "THERE GOES SECOND
    GEAR":
    END
350 IF PEEK (14400) = 32
    THEN S = C:
    GOTO 270
360 IF PEEK (14400) = 64
    THEN S = S + 1:
    P = P + 1:
    GOTO 250
370 IF PEEK (14400) = 8X = X + 1
380 IF X = 1 PRINT @768, "1ST GEAR";
390 IF X = 2 PRINT @768, "2ND GEAR";
400 IF X = 3 PRINT @768, "3RD GEAR";
410 S = S - 1:
    PRINT @C, "
420 IF S < 896S = 896
430 IF S > 958S = 958
440 P = P - 1:
    IF P < 0P = 0
450 GOTO 270
-----

```

Speed Shifter

460 DATA 84, 72, 73, 83, 32, 73, 83, 32, 84, 72, 69, 32, 83, 80, 69, 69  
68, 32, 83, 72, 73, 70, 84, 69, 82, 32, 82, 65, 67, 69, 46, 83, 72,  
73, 70, 84, 32, 71, 69, 65, 82, 83, 32, 65, 83, 32, 89, 79, 85, 32, 6  
8, 82, 73, 86, 69, 46, 72, 73, 84, 32, 84, 72, 69, 32, 85, 80, 32, 65  
82, 82, 79, 87, 32, 84, 79, 32  
470 DATA 83, 72, 73, 70, 84, 32, 71, 69, 65, 82, 83, 46, 65, 78, 68, 32  
84, 72, 69, 32, 82, 73, 71, 72, 84, 32, 65, 82, 82, 79, 87, 32, 70,  
79, 82, 32, 84, 72, 69, 32, 71, 65, 83, 46, 83, 69, 69, 32, 72, 79, 8  
7, 32, 70, 65, 83, 84, 32, 89, 79, 85, 32, 67, 65, 78, 32, 82, 69, 65  
67, 72, 32, 57, 48, 77, 80, 72, 46  
480 DATA 72, 73, 84, 32, 69, 78, 84, 69, 82, 32, 84, 79, 32, 80, 76, 65  
, 89, 46, 1

---

---

# 14

---

## **Pushing BASIC to Its Limits**

Pushing BASIC code to its limits would be a good name for SPACE ARCADE! In this game there are 30 little robots on the screen.

## **Defining the Robots**

The program starts by defining one robot in three different positions: once with both feet down, once with one leg raised and last with the other.

This is done in lines 140 to 170. Then we jump to lines 200 to 260 and print 10 robots across the screen in the first position. Also, we point a string containing 128 characters to the top two lines of our screen.

This string contains 10 robots which all look alike. We do this for each of the original robots. After this is done, we have three new strings, each containing 10 robots in different positions.

## **Marching Robots**

We now print the first string at position 0 on the screen, next the second string, and finally the third. Then we print a clear string at position 0 and move the robots down. This creates the appearance of robots marching down the screen. We next convert all 30 robots into an array and begin the game.

## **Checking the Robots**

As each robot moves out of the array, we keep checking the array and screen locations so that we don't try to move down a robot which doesn't exist any more. This is taken care of in line 320. If the random robot selector in line 300 picks a robot position that doesn't exist, we go back and select another position.

### Moving and Shooting

The player moves back and forth across the bottom of the screen by pressing the left and right arrows and can shoot back at the robots by using the up arrow key. If you shoot all the robots, 30 new robots come marching down the screen.

### I'm Working as Fast as I Can!

BASIC has quite a time trying to follow this code, but not half as much as the programmer had! After completing this game, I made a few changes in certain sections of the code, and then I compiled this program for greater speed. The changes were a major undertaking, but the end results were worth it.

In other chapters, we are going to investigate some machine-language subroutines and also some utility programs. These will make your BASIC programs very fast and at the same time will help you eliminate some of the work of writing programs.

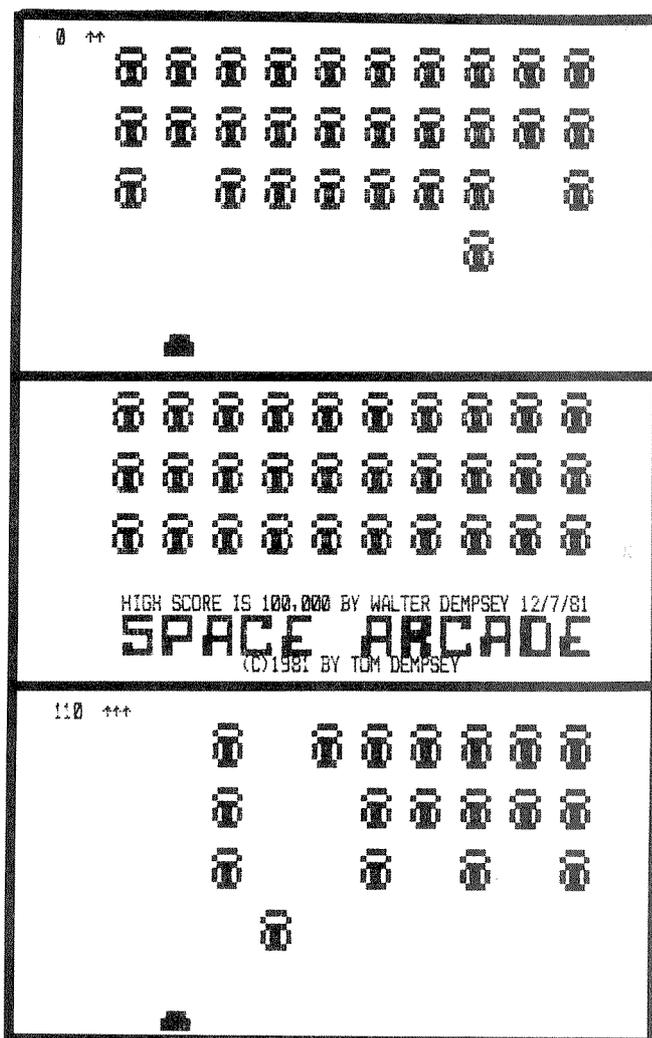


Figure 14.1 *Space Arcade Program*

```
100 CLEAR 2000:
    PL = 3:
    DIM S(30)
110 CLS :
    GOSUB 780:
    PRINT @768,A1$;:
    N = 30:
    E = 16328
120 PRINT @712,"HIGH SCORE IS 100,000 BY WALTER DEMPSEY
    12/7/81";
130 FOR X = 15360 TO 16383:
    POKE X,128:
    NEXT
140 M1$ = CHR$(166) + CHR$(179) + CHR$(153) + CHR$(2
    6) + STRING$(3,24) + CHR$(165) + CHR$(159) + CHR$(
    142)
150 M2$ = CHR$(166) + CHR$(179) + CHR$(153) + CHR$(2
    6) + STRING$(3,24) + CHR$(141) + CHR$(175) + CHR$(
    154)
160 M3$ = CHR$(166) + CHR$(179) + CHR$(153) + CHR$(2
    6) + STRING$(3,24) + CHR$(165) + CHR$(191) + CHR$(
    154)
170 M4$ = CHR$(128) + CHR$(128) + CHR$(128) + CHR$(2
    6) + STRING$(3,24) + CHR$(128) + CHR$(128) + CHR$(
    128)
180 GOSUB 530:
    GOSUB 730
190 A1$ = STRING$(128,128):
    B4$ = A1$
200 FOR X = 7 TO 55 STEP 5:
    PRINT @X,M1$;:
    NEXT
210 M = VARPTR(A1$):
    POKE M + 1,0:
    POKE M + 2,60:
    B1$ = A1$
220 FOR X = 7 TO 55 STEP 5:
    PRINT @X,M2$;:
    NEXT
230 M = VARPTR(A1$):
    POKE M + 1,0:
    POKE M + 2,60:
    B2$ = A1$
240 FOR X = 7 TO 55 STEP 5:
    PRINT @X,M3$;:
    NEXT
250 M = VARPTR(A1$):
    POKE M + 1,0:
    POKE M + 2,60:
    B3$ = A1$:
```

Space Arcade Program

```
T = 384
260 FOR X = 0 TO T STEP 64:
    PRINT @X,B1$;:
    A = USR 0(1111):
    PRINT @X,B2$;:
    A = USR 0(1111):
    PRINT @X,B4$;:
    NEXT
270 PRINT @X,B3$;:
    FOR S = X + 51 TO X + 5 STEP - 5:
        S(N) = S + 2:
        N = N - 1:
    NEXT
280 T = T - 192:
    IF T < 0
    THEN 290: .....
    ELSE 260: .....
-----
290 X = 0:
    C = 0:
    I = 0:
    M = 0:
    U = 0
300 RANDOM :
    R = RND (30):
    M = 15360 + S(R):
    P = PEEK (M)
310 IF PL = - 1
    THEN 480: .....
    ELSE IF TS > = 30
        THEN 500 .....
320 IF P < > 179
    THEN 300: .....
    ELSE X = M
330 IF S(R) < 255
    THEN ZZ = 128:
    ELSE IF S(R) > 447
        THEN ZZ = 128:
        ELSE ZZ = 64
340 I = RND (2):
    IF I = 2 AND PEEK (M + 192) = 179
    THEN I = 0
350 IF PEEK (14400) = 32 POKE E - 1,128:
    POKE E,128:
    POKE E + 1,128:
    E = E - 5
360 IF PEEK (14400) = 64 POKE E - 1,128:
    POKE E,128:
    POKE E + 1,128:
    E = E + 5
370 IF E < 16328
```

```

      THEN E = 16328:
      ELSE IF E > 16373
          THEN E = 16373
380  IF PEEK (14400) = 8
      THEN W = E - 128:
          A = USR 0(5151):
          GOSUB 460
390  POKE E - 1,190:
      POKE E,191:
      POKE E + 1,189
400  IF I = 2 GOTO 590 .....
410  IF X > 0
      THEN P = PEEK (X):
          POKE X,92
420  IF X = E
      THEN PL = PL - 1:
          FOR C = 1 TO 10:
              A = USR 0(7979):
              A = USR 1(0):
              NEXT :
              PRINT @0, STRING$ (64,128);:
              IF PL = - 1
                  THEN 480 .....
-----
430  PRINT @0,F1;" "":
      IF PL > 30
          THEN PRINT STRING$ (30,"[");:
      ELSE PRINT STRING$ (PL,"[");
440  IF X > 0
      THEN POKE X,P:
          X = X + ZZ:
          IF X > 16383
              THEN X = 0:
          ELSE GOTO 350 .....
-----
450  GOTO 290 .....
-----
460  J = PEEK (W):
      IF J = 179
          THEN PRINT @W - 15361,M4$;:
              F1 = F1 + 10:
              IF W > 15999
                  THEN RETURN : .....
              ELSE TS = TS + 1:
                  RETURN .....
470  POKE W,91:
      POKE W,J:
      W = W - 192:
      IF W < 15424
          THEN RETURN : .....
      ELSE 460 .....
-----

```

Space Arcade Program

```

480     PRINT @0,"SCORE =",F1;" PRESS ENTER TO PLAY AGA
        IN";
490     I$ = INKEY$ :
        IF I$ = CHR$ (13)
        THEN RUN : .....
        ELSE 490 .....
-----
500     PL = PL + 1:
        TS = 0:
        T = 384:
        W = 0
510     FOR X = 0 TO T STEP 64:
        PRINT @X,B1$;;
        A = USR 0(1111):
        PRINT @X,B2$;;
        A = USR 0(1111):
        PRINT @X,B4$;;
        NEXT :
520     PRINT @X,B3$;
        T = T - 192:
        IF T < 0
        THEN 290: .....
        ELSE 510 .....
-----
530     M$ = CHR$ (205) + CHR$ (127) + CHR$ (10) + CHR$
        (77) + CHR$ (68) + CHR$ (62) + CHR$ (1) + CHR$
        (105) + CHR$ (211) + CHR$ (255) + CHR$ (45) + CHR$
        (32) + CHR$ (253) + CHR$ (60) + CHR$ (105) + CHR$
        (211) + CHR$ (255) + CHR$ (45) + CHR$ (32) + CHR$
        (253) + CHR$ (13) + CHR$ (16) + CHR$ (238) + CHR$
        (175)
540     M$ = M$ + CHR$ (211) + CHR$ (255) + CHR$ (201)
550     I = VARPTR (M$)
560     M = PEEK (I + 1) + PEEK (I + 2) * 256:
        IF M > 32767
        THEN M = M - 65536
570     DEF USR 0 = M
580     RETURN .....
-----
590     PRINT @S(R) - 1,M4$;
600     FOR C = S(R) - 1 TO 639 STEP 64:
        PRINT @C,M1$;;
        PRINT @C,M2$;;
        A = USR 0(5959):
        PRINT @C,M4$;;
        NEXT
610     IF C + 15680 < E
        THEN PRINT @C,M4$;;
        C = C + 5:
        IF C > 700
        THEN I = 0:

```

```

                                GOTO 420 .....
-----
620      IF    C + 15680 > E
        THEN PRINT @C,M4$;:
            C = C - 5:
            IF    C < 640
            THEN I = 0:
                GOTO 420 .....
-----
630      PRINT @C,M3$;:
        A = USR 0(5959)
640      FOR    U = 15553 + C TO 16383 STEP 128:
            P = PEEK (U):
            POKE U,92:
            POKE U,P
650      IF    U = E PRINT @C,M4$;:
            PL = PL - 1:
            FOR    C = 1 TO 10:
                A = USR 0(7979):
                A = USR 1(0):
                NEXT :
                PRINT @0, STRING$ (64,128);:
                TS = TS + 1:
                IF    PL = - 1
                THEN 480: .....
                ELSE 290 .....
660      IF    PEEK (14400) = 32 POKE E - 1,128
            :
            POKE E,128:
            POKE E + 1,128:
            E = E - 5
670      IF    PEEK (14400) = 64 POKE E - 1,128
            :
            POKE E,128:
            POKE E + 1,128:
            E = E + 5
680      IF    E < 16328
        THEN E = 16328:
        ELSE IF    E > 16373
        THEN E = 16373
690      IF    PEEK (14400) = 8
        THEN W = E - 128:
            A = USR 0(5151):
            GOSUB 460
700      IF    W > 15871 AND I = 2 AND PEEK (14
        400) = 0
        THEN I = 0:
            F1 = F1 + 90:
            W = 0:
            U = 16383:
            PRINT @C,M4$;:

```

Space Arcade Program

```

                                C = 0:
                                TS = TS + 1:
                                GOTO 290 .....
710      POKE E - 1,190:
                                POKE E,191:
                                POKE E + 1,189
720      NEXT U:
                                GOTO 610 .....
-----
730      V$ = CHR$ (1) + CHR$ (1) + CHR$ (4) + CHR$
                                (11) + CHR$ (33) + CHR$ (255) + CHR$ (59) +
                                CHR$ (35) + CHR$ (126) + CHR$ (47) + CHR$
                                (203) + CHR$ (255) + CHR$ (203) + CHR$ (18
                                3) + CHR$ (119) + CHR$ (11) + CHR$ (120) +
                                CHR$ (177) + CHR$ (32) + CHR$ (243) + CHR$
                                (201)
740      I = VARPTR (V$)
750      M = PEEK (I + 1) + PEEK (I + 2) * 256:
                                IF M > 32767
                                THEN M = M - 65536
760      DEF USR 1 = M
770      RETURN .....
-----
780      A1$ = " " + CHR$ (183) + CHR$ (179)
                                + CHR$ (179) + " " + CHR$ (151) + CHR$ (13
                                1) + CHR$ (171) + " " + CHR$ (170) + CHR$
                                (131) + CHR$ (171) + " " + CHR$ (151) + CHR$
                                (131) + CHR$ (143) + " " + CHR$ (170) + CHR$
                                (131) + CHR$ (131) + " " + CHR$ (170) +
                                CHR$ (131) + CHR$ (131) + " " + CHR$ (171) + " " + CHR$ (191)
790      A1$ = A1$ + CHR$ (179) + CHR$ (191) + " " +
                                CHR$ (151) + CHR$ (131) + CHR$ (143) + " "
                                + CHR$ (170) + CHR$ (131) + CHR$ (171) + "
                                " + CHR$ (191) + CHR$ (131) + CHR$ (169) +
                                " " + CHR$ (170) + CHR$ (131) + CHR$ (131)
                                + " " + CHR$ (188) + CHR$
                                (176) + CHR$ (186) + " " + CHR$ (191)
800      A1$ = A1$ + CHR$ (131) + CHR$ (131) + " " +
                                CHR$ (191) + CHR$ (131) + CHR$ (171) + " "
                                + CHR$ (181) + CHR$ (176) + CHR$ (184) + "
                                " + CHR$ (191) + CHR$ (179) + CHR$ (176) +
                                " " + CHR$ (191) + CHR$ (131) + CHR$ (1
                                71) + " " + CHR$ (191) + CHR$ (131) + CHR$
                                (189) + " " + CHR$ (181) + CHR$ (176) + CHR$
                                (184)
810      A1$ = A1$ + " " + CHR$ (191) + CHR$ (131) +
                                CHR$ (171) + " " + CHR$ (191) + CHR$ (176)
                                + CHR$ (186) + " " + CHR$ (191) + CHR$ (17
                                9) + CHR$ (176) + "
                                " + CHR$ (40) + CHR$ (67) + CHR$ (41

```

```

) + CHR$ (49) + CHR$ (57) + CHR$ (56) + CHR$
(49) + " " + CHR$ (66) + CHR$ (89) + " "
820  A1$ = A1$ + CHR$ (84) + CHR$ (79) + CHR$ (
77) + " " + CHR$ (68) + CHR$ (69) + CHR$ (
77) + CHR$ (80) + CHR$ (83) + CHR$ (69) +
CHR$ (89) + "
830  RETURN .....

```

---

Line 730 checks for the enter key, which returns us to the main move mode of the program.

Line 800 tells the computer where in memory to find the start of string\$, which will become our new program created from the screen.

Line 810 takes the values that are in the screen positions, and from line 830 we get the values that are in the program lines. If this value is not an 88, we increment (A) and check again. If we find an 88, which is character (X), we POKE the screen value into this memory location.

Line 815 checks to assure we have saved past a certain screen position before printing the message "SAVING DATA."

Line 850 deletes the drawing program, because there is no reason to save it. Lines 1000 to 1050 are the dummy string\$ waiting to be filled.

### Easy Does It

All the BASIC code for graphics in these games was written by my computer. Of the rest of the code, 99% was done by merging separate sections from a disk file. It's not that I'm lazy; it's just that I'm the worlds slowest typist, so I knew I had to teach my computer to do all the typing of BASIC program code for me!

You can use these methods for writing your own games. Whenever you write a subroutine that works, save it. Then you'll never have to write it again!

When you write a utility program to accomplish a task, try to write it in such a way as to make it adaptable to more than just that one need. Whenever you find yourself repeating certain steps or the same code over and over again, — that's the time to write a program so the computer can do it for you!

Don't worry about making mistakes. You can't hurt the computer. The worst you can do is lose a program. To make mistakes is to learn! But not to learn is a bad mistake.

### SERVICE Records Program

SERVICE records inventory was written for a friend of mine in the office-machine repair business. He wanted a program which would list his customer accounts by name, make of machine, serial number and date of service call, as well as list all accounts. He wanted a program that was in BASIC with an automatic edit feature built in. I included the program in this book because of one nice feature it contains.

## Program explanation

When you ask the program to list all accounts, it prints the line number of the data statements. This feature makes it very easy to edit any data you wish. The thing to remember is that the data lines should read 260, 270, 280, in increments of ten. This can be changed by adjusting lines 140 and 230.

The most important feature of this program is the use of a counter on the data lines. This gives us a means of keeping track of each data line as it's being read, — then displaying on the screen for use in editing the customer files. Remember that counting is one of the things the computer is best at, so why not make the most of its ability?

Try to let your computer do the work for you whenever possible. After all, this is what we have them for, — right?

### Program Explanation

Line 10 clears screen.

Line 20 generates large print.

Lines 30 to 130 list command options.

Line 140 sets L to start of data lines and asks for user input.

Lines 170 to 180 read data lines.

Lines 190 to 250 print control logic.

Lines 260 and following are data lines.

---

Figure 14.2 *Service Records Program*

```
10 CLS
20 PRINT CHR$( 23)
30 PRINT @452, "MACHINE SERVICE RECORDS"
40 FOR X = 1 TO 1000:
NEXT
50 CLS
60 PRINT "DATA MAY BE LISTED BY : "
70 PRINT
80 PRINT "
90 PRINT "
100 PRINT "
110 PRINT "
120 PRINT "
A LINES"
130 PRINT
140 L = 500:
PRINT :
INPUT "HOW SHOULD I LIST";I$
141 IF I$ = "C" PRINT "$";T:
FOR X = 1 TO 1000:
NEXT
145 CLS
146 PRINT "DATA ON ";I$
DATE"
ACCOUNT NAME"
MACHINE"
SER. NUMBER"
LIST ALL / EDIT DAT
```

Program Explanation

```
150 READ D$,N$,M$,S$,C$,C
160 IF I$ = D$ PRINT N$,M$,S$,C$
170 IF I$ = N$ PRINT D$,M$,S$,C$
180 IF I$ = M$ PRINT D$,N$,S$,C$
190 IF I$ = S$ PRINT D$,N$,M$,C$:
      LET T = C
200 IF I$ = "ALL" PRINT L;D$,N$,M$,S$:
      L = L + 10
210 IF D$ = "XXX"
220 THEN 1000 .....
      GOTO 150 .....
-----
500 DATA 3-24-80,ACEAUTO,IBM,455357,SERVICECONT,90.
      00
999 DATA XXX,X,X,X,X,0
1000 RESTORE :
      GOTO 140 .....
```

---

# NOTES

---

# 15

---

## Graphic Utility Programs

In this chapter, we'll take a look at some ways to speed up our program writing. We will also take a closer look at some methods of letting the computer do the hard work for us, so we can spend more time on ideas instead of typing on the keyboard all day. After you become used to these utility programs, you should be able to complete a game program in a day or a week that would otherwise take up to a month or even three months to complete!

### LSET 1K

The following subroutine can be used to transfer 1K of packed strings from your program to the screen. This is possible by using the LSET command in disk BASIC and is very fast. To use this subroutine, first pack a full screen using the AUTOGRAF COMPILER. Then insert the following lines of code:

```
1060 GOSUB 50000
1070 GOTO 1070
```

Next, add the LSET1K subroutine listed below.

---

Figure 15.1 *LSET1K Routine*

```
50000 A$ = A1$:
      M = VARPTR (A$):
      POKE M + 2,60:
      POKE M + 1,0:
      LSET A$ = A1$:
      POKE M + 1,192:
      LSET A$ = A2$:
      POKE M + 1,128:
      POKE M + 2,61:
      LSET A$ = A3$:
      POKE M + 1,64:
```

## String/Fix

```
POKE M + 2,62:  
LSET A$ = A4$:  
POKE M + 1,0:  
POKE M + 2,63:  
LSET A$ = A5$:  
POKE M,64:  
POKE M + 1,192:  
POKE M + 2,63:  
LSET A$ = A6$:  
RETURN
```

---

---

The following is an explanation of the LSET1K subroutine.

First we set the variable A\$, which is the routine's working variable, equal to A1\$. Next we find the memory address of A\$ and we change that address to the first position on our screen by POKEing the least significant value of 15360 into M + 1 and the most significant value into M + 2. We then give the computer the LSET command, which tells the computer to move A\$ to its new address which (if everything works) is now the first three lines of our screen. We continue doing this for A2\$, A3\$, A4\$ and A5\$, — each time incrementing the screen address by 192. When we get to A6\$, we change the size of A\$ to 64 bytes and LSET it to the last line of the screen.

## STRING/FIX

The subroutine called STRING/FIX can be used to alter any 1K of packed STRING\$ that you might like to make last minute changes in. To use this program, add a GOSUB after the FRAME or 1K of packed STRING\$ and run your program. Use the arrow keys to draw and the arrow plus space bar to move around the screen. If you do not wish to make any changes, press the clear key, and the program will go on to the next frame (if any).

After all changes in the frame have been completed, press enter. STRING/FIX will make the changes in the packed strings of your program and continue to the next frame. After all changes have been made, hit the break key to exit and delete the STRING/FIX program from memory.

Figure 15.2 *STRING/FIX Program*

```

50000 A$ = A1$:
      M = VARPTR (A$):
      POKE M + 2,60:
      POKE M + 1,0:
      LSET A$ = A1$:
      POKE M + 1,192:
      LSET A$ = A2$:
      POKE M + 1,128:
      POKE M + 2,61:
      LSET A$ = A3$:
      POKE M + 1,64:
      POKE M + 2,62:
      LSET A$ = A4$:
      POKE M + 1,0:
      POKE M + 2,63:
      LSET A$ = A5$:
      POKE M,64:
      POKE M + 1,192:
      POKE M + 2,63:
      LSET A$ = A6$
50010 X = 0:
      Y = 0
50020 SET (X,Y)
50030 IF PEEK (14400) = 8Y = Y - 1
50040 IF PEEK (14400) = 16Y = Y + 1
50050 IF PEEK (14400) = 32X = X - 1
50060 IF PEEK (14400) = 64X = X + 1
50070 IF PEEK (14400) = 136 RESET (X,Y):
      Y = Y - 1
50080 IF PEEK (14400) = 144 RESET (X,Y):
      Y = Y + 1
50090 IF PEEK (14400) = 160 RESET (X,Y):
      X = X - 1
50100 IF PEEK (14400) = 192 RESET (X,Y):
      X = X + 1
50110 IF PEEK (14400) = 1
      THEN 50150 .....
50120 IF PEEK (14400) = 2 RETURN .....
50130 IF X < 0
      THEN X = 127:
      ELSE IF X > 127
          THEN X = 0:
          ELSE IF Y < 0
              THEN Y = 47:
              ELSE IF Y > 47
                  THEN Y = 0
50140 GOTO 50020
-----
50150 S = 15360
50160 A$ = A1$:

```

## Packed Strings to CHR\$ Converter

```
GOSUB 50170:
A$ = A2$:
GOSUB 50170:
A$ = A3$:
GOSUB 50170:
A$ = A4$:
GOSUB 50170:
A$ = A5$:
GOSUB 50170:
A$ = A6$:
GOSUB 50170:
RETURN

-----
50170 M = VARPTR (A$)
50180 A = PEEK (M + 1) + PEEK (M + 2) * 256:
      IF A > 32767
      THEN A1 = - 1 * (65536 - A):
           A = A1
50190 B = A + LEN (A$):
      IF B > 32767
      THEN A1 = - 1 * (65536 - B):
           B = A1
50200 FOR X = A TO B - 1
50210     P = PEEK (S):
           POKE X,P
50220     S = S + 1:
      NEXT X:
      RETURN

-----
```

The following is an explanation of the STRING/FIX program.

Line 50000 is our good old LSET1K subroutine. In lines 50010 through 50140, we have a short drawing program. At line 50160, we set A\$ to each of the variables in our frame of packed strings. Next, we GOTO 50170 to find out where A\$ is in memory (we are really getting the addresses for each one of the strings A1 through A6 in our program). When we get an address, we save it in (A), scan the video screen for the new graphic values for A\$ and POKE them into memory. Remember that A\$ is now at the same place in memory where A1\$ is. When we have completed this for the length of A\$, we go back and do the same for the rest of the strings and then return to move another 1K of packed strings to the screen.

### Packed Strings to CHR\$ Converter

The following utility will unpack your packed strings, convert them into CHR\$ or STRING\$ and list them out on a line printer. Each line starts like this:

```
1000 A1$
```

This utility is nice when you have a program full of packed strings and you want to send someone a listing. Be careful when you use it because the printer doesn't care how long your lines are, and the computer won't take any lines longer than 255 bytes. Don't try to take the listing from the printer and enter it into your machine if it's too long.

Figure 15.3 UNPACK Program

```

50000 '          PACKED / STRINGS CONVERTER
50010 A$ = A1$:
      L$ = "1000 A1$="
50020 Z1$ = "CHR$(":
      Z2$ = "STRING$(":
      Z3$ = ")":
      Z4$ = "+"
50030 FOR X = 1 TO LEN (A$):
      Q = 0
50040   P = ASC ( MID$ (A$,X,1)):
      IF X < LEN (A$)
      THEN Q = ASC ( MID$ (A$,X + 1,1))
50050   IF Q = P
      THEN T = T + 1
50060   IF T > 0 AND Q = P
      THEN 50120 .....
50070   IF T > 0 AND Q < > P
      THEN T = T + 1:
          L$ = L$ + Z2$ + MID$ ( STR$ (T),2,3) + ","
          + MID$ ( STR$ (P),2,3) + Z3$:
          T = 0:
          GOTO 50090 .....
50080   IF T = 0 AND Q < > P
      THEN L$ = L$ + Z1$ + MID$ ( STR$ (P),2,3) + Z3$

50090   IF X < LEN (A$)
      THEN L$ = L$ + Z4$
50100   IF X < LEN (A$) LPRINT L$;:
      L$ = ""
50110   IF X = LEN (A$) LPRINT L$:
      L$ = ""
50120 NEXT :
      END
-----

```

Below is the explanation for UNPACK.

In line 50010, we put the address of A1\$ (or the string we want to unpack) into A\$. Then we make L\$ equal "1000 A1\$=".

Line 50020 initializes Z1\$ to "CHR\$(", Z2\$ to "STRING\$(", Z3\$ to ")" and Z4\$ to "+".

At line 50030 we start our loop for A\$.

In line 50040, we use the MID\$ function to set the variable P equal to the ASC value of the first character in A\$ and Q equal to the next character in A\$.

At line 50050, if Q is equal to P, we increment the counter variable (T).

In line 50060, if the second character is the same as the first, then we drop down to line 50120 and get the next value.

## Basic Line Mod

At line 50070, if our counter is larger than zero and our second value in A\$ is different from the first value, we increment our counter and add the string "STRING\$(“ plus the value of (T) plus “,” plus the value of (P) plus “)”. Next, we GOTO 50090 to see if X is still smaller than the length of A\$. If it is, we add the string “+” to the end of L\$. Then we print L\$ and go back for more.

In line 50080, if our counter equals zero and Q is different from P, it means that there is only one of the characters and we add "CHR\$(“ plus the value of P plus “)” to L\$. Then we GOTO 50090 and print L\$.

At line 50090, if we aren't done, we add “+” to the end of L\$.

Line 50100 prints L\$, and line 50110 prints L\$ only if we are all finished. Line 50120 is the completion of our FOR NEXT loop.

### BASIC LINE MOD

This subroutine will search your BASIC program for a certain value and change it to a new value. This comes in handy if you want to change all those LPRINT commands to PRINT commands or perhaps REMARK lines into DATA lines! More on that later.

To use this program, merge it with your BASIC program and RUN 50000.

---

Figure 15.4 *LINEMOD Program*

```
500000 '                BASIC LINE MOD
500100 CLS :
      INPUT "FIND #";F:
      INPUT "CHANGE TO #";C:
      CLS
500200 Z = PEEK (16549) * 256 + PEEK (16548)
500300 IF Z > 32767
      THEN Z1 = - 1 * (65536 - Z):
           Z = Z1
500400 PRINT @0,"SEARCHING LINE #"; PEEK (Z + 2) + PEEK (Z +
      3) * 256:
           Q = Z:
           Z = Z + 4
500500 P = PEEK (Z):
      IF P = F
      THEN T = T + 1:
           PRINT @30,"LOCATED ";T;" TIME(S)":
           POKE Z,C
```

```

50060 IF P < > 0
      THEN Z = Z + 1:
          GOTO 50050 .....
50070 Z = Q:
      IF PEEK (Z + 2) + PEEK (Z + 3) * 256 = 50000 END .....
50080 Z = PEEK (Z) + PEEK (Z + 1) * 256:
      GOTO 50030
-----

```

---

The following is a breakdown of LINEMOD.

Line 50010 gets the old and new values from the user.

Line 50020 finds the start of the BASIC program we are going to change.

At line 50030, we make sure it's not larger than 32767.

In line 50040, we tell the operator which line we are looking at in memory. At line 50050, if we find the old value, we POKE in the new value and notify the operator of the change.

Line 50060 checks for the end of line. If not, then we increment the memory position and keep going.

At line 50070, if we run into the LINEMOD subroutine, we stop.

In line 50080, we jump to the next line number and keep going.

## **LINEFINDER**

This subroutine will find a particular line number and return the starting memory address of that line. To use it, set L to the number of the line you're looking for. When it returns, L will contain the memory address of that line number. It would look like this in the command mode:

```
L=1000:GOSUB50000:PRINTL
```

---

Figure 15.5 *LINEFINDER Program*

```
50000 ' LINE FINDER
50010 Z = PEEK (16549) * 256 + PEEK (16548)
50020 IF Z > 32767
    THEN Z1 = - 1 * (65536 - Z):
        Z = Z1
50030 IF PEEK (Z + 2) + PEEK (Z + 3) * 256 = L
    THEN L = Z:
        RETURN .....
50040 IF PEEK (Z + 2) + PEEK (Z + 3) * 256 > L STOP
50050 Z = PEEK (Z) + PEEK (Z + 1) * 256:
    GOTO 50020
```

---

If you want to get really fancy with your DATA lines and read only certain lines, you can change line 50030 of the LINEFIND subroutine to this:

```
50030 IFPEEK(Z+2)+PEEK(Z+3)*256=L THENPOKEZ+4,147:RETURN
```

With a little effort in programming, you should be able to change data lines into remark lines or back any time you wish!

### DISKFILE

This program will help you keep track of your disks, and is very easy to use. It starts by listing options on the screen in this way:

```
1 = READ DISK <> TOTAL PROGRAMS IN FILE = 0
2 = LIST PROGRAMS
3 = PROGRAM SEARCH
4 = HARD COPY
5 = SAVE DISK FILE
```

The first step you must take to make use of this program is to number all your disks from 1, 2, 3, . . . etc. After this is done, you're ready to begin the diskfile program.

**SPECIAL NOTE:** The DISKFILE program should be saved on one of your operating system disks and placed in drive 0.

Now we're ready to run the program. GOTO DISK BASIC and type RUN"DISKFILE. Then follow the steps listed below.

1. Place disk number 1, side 1 in drive 1.
2. Press key number 1, then press number 1 again and enter to enter the disk number.
3. When the screen clears and the menu returns, turn the disk over. Now press 1 and 1 again for the disk number; then hit enter.
4. Repeat these steps for each disk you want to save in your file.

After all disks are in the file (the limit is 1000 programs), press number 5 to save the file on disk. Now break the program and run again. Press number 2 to list the file. First pressing 4 and then 2 gives a hard copy. Pressing number 3 will let you search the file for a certain program name.

This program is short and has no safeguards built in. For instance, if you try to read a disk that has not been formatted or has no directory, the program will crash. I'll leave it up to you to make additions or changes to suit your needs. I seldom find programs (even ones sold for large amounts of money) to be just what I'm looking for and usually end up making alterations to them to suit my own needs.

The following paragraphs are the line comments for the program; the program listing appears at the end of this section.

At line 10, we clear sufficient string space and dimension our array to hold about 1000 program names. Lines 20-110 are the menu lines, and lines 120 and 130 check whether the printer is on when the user wants a hard copy. Line 150 gets the disk number for the file information, and line 160 calls up the disk directory.

Lines 170 to 240 scan the screen and place the program names into the array A\$(N). Lines 260 - 280 save the array A\$(N) on disk, and in lines 290 - 340, the program opens the file "END" and gets the value of (N) (the total number of programs that have been saved to disk). We next load in the data and print it to the screen and lineprinter.

Lines 360 - 470 search the file for the program we want and then print out the disk number that it is on. This is the program search mode. There are many different ways of writing this type of program; this is but one example. I hope it will give you ideas for other programs that can be used to control program files on the disk system.

Figure 15.6 DISKFILE Program

```

10 CLS :
   CLEAR 10000:
   DIM A$(1000):
   N = 1
20 CLS :
   PRINT "1 = READ DISK <> TOTAL PROGRAMS IN FILE =";N -
   1
30 PRINT "2 = LIST PROGRAMS"
40 PRINT "3 = PROGRAM SEARCH"
50 PRINT "4 = HARD COPY"
60 PRINT "5 = SAVE DISK FILE"
70 IS = INKEY$:
   IF IS = "" :
   THEN 70 :
   IF IS = "1" :
   THEN 150 :
   IF IS = "2" :
   THEN 290 :
   IF IS = "3" :
   THEN 360 :
   IF IS = "5" :
   THEN 260 :
   IF IS = "4" AND PEEK(14312) <> 63 PRINT @220,"PR
   INTER OFF";:
   HS = "N":
   GOTO 70 :
130 IF IS = "4" AND PEEK(14312) = 63 PRINT @220,"PRIN
   TER READY";:
   HS = "Y":
   GOTO 70 :
140 GOTO 70
-----
150 CLS :
   INPUT "DISK NUMBER";D
160 CMD "DIR :1"
170 M = 15488
180 P = PEEK (M):
   IF P = 32 :
   THEN 210 :
190 A$(N) = A$(N) + CHR$(P)
200 M = M + 1:
   GOTO 180
-----
210 A$(N) = STR$(D) + " " + A$(N)
220 N = N + 1
230 P = PEEK (M):
   IF P <> 32 :
   THEN T = 0:
   GOTO 180 :
240 M = M + 1:
   T = T + 1:
   IF T > 30 :
   THEN T = 0:
   GOTO 20 :
250 GOTO 230
-----
260 OPEN "O",1,"LIST":
   FOR X = 1 TO N:
   PRINT #1,A$(X) :
   NEXT :
   CLOSE
270 OPEN "O",1,"END":
   PRINT #1,N:
   CLOSE
280 GOTO 20
-----
290 CLS :
   OPEN "I",1,"END":
   INPUT #1,N:

```

```

CLOSE
300 OPEN "I",1,"LIST":
FOR X = 1 TO N - 1:
INPUT #1,A$(X):
NEXT X:
CLOSE
310 FOR X = 1 TO N - 1:
PRINT "DISK NUMBER ";A$(X)
320 IF H$ = "Y" LPRINT "DISK NUMBER ";A$(X)
330 Q = Q + 1:
IF Q = 15
THEN Q = 0:
PRINT @960,"WAITING";:
INPUT Z
340 NEXT X
350 PRINT @960,"WAITING";:
INPUT Z:
CLS:
Q = 0:
H$ = "N":
GOTO 20

```

---

```

360 CLS:
OPEN "I",1,"END":
INPUT #1,N:
CLOSE
370 INPUT "ENTER PROGRAM WANTED";B$
380 E$ = B$
390 OPEN "I",1,"LIST"
400 FOR X = 1 TO N - 1:
INPUT #1,A$(X)
FOR T = 1 TO LEN (A$(X))
420 IF ASC ( MID$ (A$(X),T,1)) < 64 AND Q1 =
0
THEN 440: .....
ELSE Q1 = 1: .....
430 C$ = C$ + MID$ (A$(X),T,1)
440 NEXT T
450 IF C$ = B$
THEN PRINT "DISK NUMBER ";A$(X):
CLOSE:
GOTO 350 .....
460 B$ = E$:
C$ = "":
Q1 = 0
470 NEXT X:
CLOSE:
PRINT "PROGRAM NOT LOCATED":
GOTO 350

```

---

## Enlarge

### ENLARGE

This program is more a subroutine than a program, although it can be used by itself. The idea is to draw a small picture on the screen and then let the computer enlarge it by about 4 times.

The program gives you a protected field to draw in. Next it scans the x,y coordinates, converts them into POKE values and duplicates the image on the screen (but in much larger graphics).

I have spent a great deal of time writing programs (or, I should say utilities and subroutines) that perform many different tasks. Examples are reverse video, reduce or enlarge, block moves of memory or screen, — even a program that writes game programs for me (or at least 90% of the program). I still have to push a few buttons. Many of these subroutines are included in this book. One of the projects I'm working on at the moment is a greatly improved version of the program that writes games. When it's completed and working properly, all the operator will have to do is enter the description of the game and the action that he wants to take place, the type of characters or graphic objects, and then the computer will create the game for you! Needless to say, this is quite a project and I have been gathering the code and subroutines for this program for about a year now.

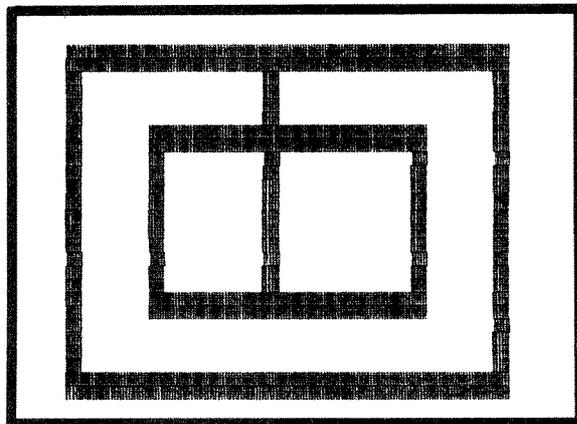
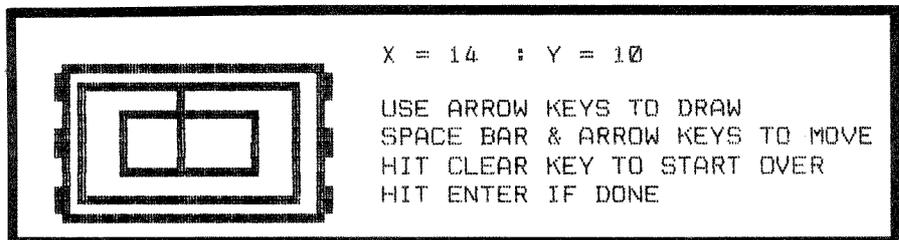


Figure 15.7 ENLARGE Program

```

51190 CLS :
      FOR M = 1 TO 32:
        SET (M,2):
      NEXT :
      FOR M = 1 TO 32:
        SET (M,18):
      NEXT :
      FOR M = 2 TO 18:
        SET (1,M):
      NEXT :
      FOR M = 2 TO 18:
        SET (32,M):
      NEXT :
      POKE 15424,191:
      POKE 15552,191:
      POKE 15680,191:
      POKE 15440,191:
      POKE 15568,191:
      POKE 15696,191:
51200 PRINT @148,"USE ARROW KEYS TO DRAW";
51210 PRINT @212,"SPACE BAR & ARROW KEYS TO MOVE";
51220 PRINT @276,"HIT CLEAR KEY TO START OVER";
51230 PRINT @340,"HIT ENTER IF DONE";:
      X = 2:
      Y = 3:
      A = 14400
51240 SET (X,Y):
      P = PEEK (A)
51250 PRINT @20,"X =";X - 2;" : Y =";Y - 3;
51260 IF P = 8
      THEN Y = Y - 1
51270 IF P = 16
      THEN Y = Y + 1
51280 IF P = 32
      THEN X = X - 1
51290 IF P = 64
      THEN X = X + 1
51300 IF P = 1
      THEN 51380 .....
51310 IF P = 2
      THEN 51190 .....
51320 IF P = 136 RESET (X,Y): .....
      Y = Y - 1
51330 IF P = 144 RESET (X,Y):
      Y = Y + 1
51340 IF P = 160 RESET (X,Y):
      X = X - 1
51350 IF P = 192 RESET (X,Y):
      X = X + 1
51360 IF X < 2X = 2:
      ELSE IF X > 31X = 31:
      ELSE IF Y < 3Y = 3:
      ELSE IF Y > 17Y = 17
51370 GOTO 51240
-----
51380 M = 31400:
      PRINT @448,"SCANNING DATA";
51385 FOR Y = 3 TO 17:
      FOR X = 2 TO 31
51390         IF POINT (X,Y)
          THEN N = 191:
          ELSE N = 128
          POKE M,N:
          M = M + 1
51400         NEXT X
51410      NEXT Y
51430 CLS :
      M = M - 1

```

## How to Use Copy

```
51440 S = 15360:
      T = S
51450 FOR X = 31400 TO M:
      O = PEEK (X)
51460   POKE S, O:
      S = S + 1:
      Z = Z + 1:
      IF Z = 30 Z = 0:
      S = T + 64:
      T = T + 64
51470 NEXT X
51480 GOTO 51480
```

---

## COPY

Did you ever have need for a machine-language subroutine but didn't want to take the time to learn machine-language? Look no further! The COPY program enables you to convert anything on the screen into a machine-language subroutine which you can call by the A=USR(0) command or by simply pressing the down arrow on the keyboard.

This program allows you to stop any program execution at any time, take a picture of whatever is on the screen, save it and recall it any time you wish. This is done by making a 1K block move and takes less than 1 second.

To take the picture, press the up arrow. When you wish to display the picture, press the down arrow. The ease of doing this allows you to save important data or pictures and recall them later without having to run the program again from the start.

### How to Use COPY

To load the program, set memory size to 31488 and type SYSTEM (enter). Then type COPY (enter). After the program loads, press (/) (enter), which brings you back to BASIC. Now you can load any program you wish.

After your BASIC program is running, you can press the up arrow key to take a picture of anything you like. This takes everything on the screen and saves it in memory. To recall the picture, press the down arrow.

If you would like to call this screen from your BASIC program, insert the following lines in your program.

```
POKE 16526,41:POKE 16527,123
A=USR(0)
```

Each time the computer encounters the A=USR(0), you will have the picture on the screen.

Figure 15.8 COPY Program

```

7B00          00100          ORG          7B00H
7B00 2A1640   00110  START  LD          HL,(4016H)
7B03 22427B   00120          LD          (JUMPER),HL
7B06 210F7B   00130          LD          HL,ENTRY
7B09 221640   00140          LD          (4016H),HL
7B0C C3CC06   00150          JP          06CCH
7B0F 014038   00160  ENTRY  LD          BC,14400
7B12 0A       00170          LD          A,(BC)
7B13 E608     00180          AND         8
7B15 281F     00190          JR          Z,DOWN
7B17 11003C   00200          LD          DE,15360
7B1A 21D47B   00210          LD          HL,31700
7B1D 010004   00220          LD          BC,1024
7B20 1A       00230  LOOP  LD          A,(DE)
7B21 77       00240          LD          (HL),A
7B22 23       00250          INC         HL
7B23 13       00260          INC         DE
7B24 0B       00270          DEC         BC
7B25 78       00280          LD          A,B
7B26 B1       00290          OR          C
7B27 20F7     00300          JR          NZ,LOOP
7B29 11003C   00310  COPY  LD          DE,15360
7B2C 21D47B   00320          LD          HL,31700
7B2F 010004   00330          LD          BC,1024
7B32 EDB0     00340          LDIR
7B34 280B     00350          JR          Z,AWAY
7B36 014038   00360  DOWN  LD          BC,14400
7B39 0A       00370          LD          A,(BC)
7B3A E610     00380          AND         16
7B3C 2803     00390          JR          Z,AWAY
7B3E C3297B   00400          JP          COPY
7B41 C3       00410  AWAY  DEFB       0C3H
7B42 E3       00420  JUMPER DEFB       0E3H
7B43 03       00430          DEFB       003H
7B00          00440          END          START
00000 TOTAL ERRORS
AWAY       7B41
COPY       7B29
LOOP       7B20
DOWN       7B36
ENTRY      7B0F
JUMPER     7B42
START      7B00

```

### Program Execution

Lines 100 - 150 set up the basic keyboard entry points.  
Line 160 loads the BC register with keyboard address.  
Line 170 loads A register with value in memory address 14400.  
Line 180 checks for up arrow.  
Line 190 If no, then go check for down arrow.  
Line 200 loads DE register with screen starting address.  
Line 210 loads HL register with start of save data address.  
Line 220 uses BC register as counter for screen positions.  
Line 230 loads A register with what was found in DE.  
Line 240 load HL with value in A register.  
Line 250 Add 1 to HL register.  
Line 260 Add 1 to DE register.  
Line 270 Subtract 1 from counter register BC.  
Lines 280 - 290 load A with value in counter.  
Line 300 If not 0, go back to loop and do it all again.  
Line 310 loads DE register with screen starting location.  
Line 320 loads HL register with starting address of data.  
Line 330 loads our counter again.  
Line 340 Ok, move 1k of data starting at 31700 and put it on the screen starting at 15360.  
Line 350 When you're done, go back to BASIC and start checking for an up or down arrow again.  
Line 360 loads BC register with keyboard address.  
Line 370 loads the value at 14400 into the A register.  
Line 380 Is it a down arrow?  
Line 390 If not, go back to BASIC.  
Line 400 If it is, then jump over to copy and do what it says.  
Lines 410 - 440 define our address for BASIC and keyboard.

### PCOPY

This program is practically the same as COPY, except that PCOPY sends the contents of your screen to a printer.

#### How to Use PCOPY

Load the program using the system command. Don't forget to set memory size to 31488. After typing SYSTEM (enter), type the word PCOPY (enter). After the program has loaded, press (/) (enter), and you will return to BASIC.

Now you are ready to load your BASIC program. Any time you wish to save the screen contents, press the up arrow. PCOPY will stop the BASIC program and send the screen contents to your printer. After this is done, control returns to the BASIC program. You can do this as often as you wish. Before using PCOPY, remember to set your printer controls from BASIC first.

Figure 15.9 PCOPY Program

```

9A00          00100          ORG          9A00H
9A00 2A1640   00110  START  LD          HL, (4016H)
9A03 22479A   00120          LD          (JUMPER), HL
9A06 210F9A   00130          LD          HL, ENTRY
9A09 221640   00140          LD          (4016H), HL
9A0C C3CC06   00150          JP          06CCH
9A0F 014038   00160  ENTRY  LD          BC, 14400
9A12 0A        00170          LD          A, (BC)
9A13 E608     00180          AND         8
9A15 282F     00190          JR          Z, AWAY
9A17 018038   00191          LD          BC, 14464
9A1A 0A        00192          LD          A, (BC)
9A1B E601     00193          AND         1
9A1D 2827     00194          JR          Z, AWAY
9A1F 11003C   00200          LD          DE, 15360
9A22 210004   00210          LD          HL, 1024
9A25 CD349A   00220  LOOP   CALL         TEST
9A28 1A        00230          LD          A, (DE)
9A29 32E837   00240          LD          (37E8H), A
9A2C 13        00250          INC         DE
9A2D 2B        00260          DEC         HL
9A2E 7C        00270          LD          A, H
9A2F B5        00280          OR          L
9A30 20F3     00290          JR          NZ, LOOP
9A32 2812     00300          JR          Z, AWAY
9A34 3AE837   00310  TEST   LD          A, (37E8H)
9A37 FE3F     00320          CP          03FH
9A39 20F9     00330          JR          NZ, TEST
9A3B 0E0A     00340          LD          C, 0AH
9A3D 0600     00350  DELAY1 LD          B, 0
9A3F 10FE     00360  DELAY2 DJNZ        DELAY2
9A41 0D        00370          DEC         C
9A42 C23D9A   00380          JP          NZ, DELAY1
9A45 C9        00390          RET
9A46 C3        00400  AWAY   DEFB        0C3H
9A47 E3        00410  JUMPER DEFB        0E3H
9A48 03        00420          DEFB        003H
9A00          00430          END          START
000000 TOTAL ERRORS
DELAY2  9A3F
DELAY1  9A3D
TEST    9A34
LOOP    9A25
AWAY    9A46
ENTRY   9A0F
JUMPER  9A47
START   9A00

```

# NOTES

# 16

---

## **Teaching Your Computer New Tricks**

From the first day I started programming, I always believed that the computer was designed to do the work and that we, the operators and programmers, only have to tell it what we want it to do!

If you believe that too, this chapter will be of great interest to you.

### **Programming Tip 1**

For any task or program code that will be repeated over and over again, you should write the code in such a way that it can fit into any program you write. Then you only have to write it once and just merge it into your program each time you need that certain operation.

### **Programming Tip 2**

Try, if possible, to write code that will automatically adjust for the computer size and system that the program might be used in. This way, you won't have to rewrite it each time you want to use it in a different size machine.

### **Programming Tip 3**

Use the fastest code or subroutine that you can write and try to keep your subroutines limited to one line, if possible.

### **Programming Tip 4**

Break your program into small sections so the BASIC interpreter won't have to jump from line 10 to line 50000 all the time.

## **Make Your Computer Do It**

If you use a lot of string packing in your programs, you should write a utility program that will allow you to draw graphics on the screen and then convert them into packed strings.

## Unpacking Packed Strings

Listed below are some subroutines you can use to speed up your graphics.

### Convert PRINT@ to X,Y Positions

```
1000 Z1=FIX(S/64):Z2=64*Z1:X=(S-Z2)*2:Y=FIX(S/64*3):RETURN
```

To use this subroutine, just make the variable (S) equal to the screen PRINT@ position, and when it returns, X,Y will be the X,Y coordinates of the same location.

### Convert X,Y to POKE Positions

```
1000 XQ=INT(X/2):XR=X-(XQ*2):YQ=INT(Y/3):YR=Y-(YQ*3)
1010 PO=15360+YQ*64+XQ:RETURN
```

To use this, load X,Y with the values, and on return, PO will equal the POKE position.

### Convert POKE Positions to PRINT@

```
1000 PR=PO-15360:RETURN
```

Subtract 15360 from the POKE position and you have the PRINT@ position. To find the POKE position, just add 15360 to the PRINT@ position.

### Find the Start of the BASIC Program

```
1000 PRINTPEEK(16549)*256+PEEK(16548)
```

### Find the End of the BASIC Program

```
1000 PRINTPEEK(16634)*256+PEEK(16633)
```

### Find the Computer Memory Size

```
1000 PRINT PEEK(16561)+PEEK(16562)*256
```

### Making Programs Unlistable

```
0 POKE16396,175:POKE16397,201:POKE27208,0:POKE27209,0
```

Enter line zero shown above. Next, enter the following line in the command mode:

```
POKE27206,254:POKE27207,255
```

Now try to list your program; then try to run it!

### Tape or Disk?

```
1000 IFPEEK(16396)=201 THEN TAPE SYSTEM ELSE DISK SYSTEM
```

### Unpacking Packed Strings

```
1000 FORX=1 TO LEN(A$):PRINTASC(MID$(A$,X,1));NEXT
```

## Machine-language Subroutine Packer

---

Figure 16.1 *Machine-language Packer Subroutine*

```

1000 L=50000
1010 Z=PEEK(16549)*256+PEEK(16548)
1020 IFZ>32767THENZ1=-1*(65536-Z):Z=Z1
1030 IFPEEK(Z+2)+PEEK(Z+3)*256=LTHENL=Z+4:GOTO1060
1040 IFPEEK(Z+2)+PEEK(Z+3)*256>LSTOP
1050 Z=PEEK(Z)+PEEK(Z+1)*256:GOTO1020
1060 P=PEEK(L):IFP<>88THENL=L+1:GOTO1060
1070 READD:IFD=-1THEN2000ELSEPOKEL,D:L=L+1:GOTO1070
1080 DATA 1,1,4,11,33,255,59,35,126,47,203,255,203,183
1090 DATA 119,11,120,177,32,243,201,-1
2000 GOSUB50000:DELETE1000-2000
50000 M$="XXXXXXXXXXXXXXXXXXXXX"
50010 Z=VARPTR(M$):DEFUSR0=PEEK(Z+1)+256*PEEK(Z+2):RETURN

```

---

To use this, enter the data values for the machine-language routine into line 1080 and insert the same number of X's into line 50000. Remember not to use any 0's because this will terminate the string.

Now run the program, list the program, make the USR call and the video should reverse. Tape users will have to change DEFUSR to their usual POKE16526,LS:POKE16527,MS. To do this, change line 50010 to the following:

```
50010 Z=VARPTR(M$):POKE16526,PEEK(Z+1):POKE16527,PEEK(Z+2)
```

---

# NOTES

# 17

---

## Machine-language Subroutines

The use of machine-language subroutines becomes more necessary as we get deeper into BASIC programming. The speed at which they run is needed to an even greater extent when we start working with graphics.

Out of this need for faster and more complex programs, a need for easier and simpler programming methods also arises.

As we learn more about machine-language and writing programs, we find that many of our problems really amount to only one or two, which are most critical. First is keeping our BASIC and machine-language codes from killing each other, and the second is how much memory we have to use to accomplish a given task.

Let's talk about the last condition first. If you have done your homework and investigated using one of the many machine-language compilers on the market, you soon realize that these programs are great for compiling large, complete programs. However, if you just need a one-line subroutine compiled to speed up your basic code, you realize that this is not the way to do it!

First of all, these compilers take too much memory to use them for one-liners. Most of them don't make use of very compact code, and some even dump about 1K of control code onto your compiled BASIC code, which may be just 20 bytes. This doesn't mean that the compiler is bad; it's just that most compilers weren't designed to compile one-line subroutines.

Getting back to the problem of keeping our machine code and BASIC code from destroying each other brings us to a real problem.

## There Must Be a Better Way

There is a better way; in fact, there are probably many better ways. The first is to learn machine-language! Now don't get all excited about that last statement. When I say learn machine language, I don't mean learn how to write inventory control programs in machine-language.

## One Step at a Time

We don't have to know that much machine-language. In fact, for writing one-line subroutines to speed up our BASIC programs, we don't have to know very much machine code at all!

What we *do* need is a way to convert our BASIC programming ideas into the machine-language logic. The first thing we have to do is start thinking in smaller terms. Thinking big is not for machine-language programmers!

### One Step at a Time

There are actually very few machine-language instructions we need to know to write just about any subroutine we need. Don't get me wrong. It's nice if you understand all the instructions, but you won't need them all for this type of programming.

Let's take a simple BASIC subroutine and convert it step by step into a machine-language subroutine. Take the following subroutine, for example.

```
10 FORX=15360TO16383:POKEX,191:NEXT:RETURN
```

Below is how the same subroutine would look, expressed in BASIC, but looking more like machine-language.

---

Figure 17.1 *BASIC Subroutine*

```
10 HL=15360
20 DE=1024
30 A=191
40 POKEHL,A
50 HL=HL+1
60 DE=DE-1
70 A=DE
80 IFA>0THEN30
90 RETURN
```

---

This BASIC subroutine will POKE a graphic character 191 into each screen position. When it has completed this task, it returns to the code that called it.

Now let's take this subroutine a line at a time and convert it into a machine-language subroutine. The following is how the program could be expressed in assembly-language.

---

Figure 17.2 *Machine-language Subroutine*

```
00100          ORG      32000          ;START OF PROGRAM
00110          LD       HL,15360       ;FOR X=15360
00120          LD       DE,1024        ;TO 16383
00130 NEXT     LD       A,191          ;A=191
00140          LD       (HL),A         ;POKEX,191 (A)
00150          INC      HL             ;X=X+1
00160          DEC      DE             ;DE=16383-X
```

```

00170      LD      A,D          ;LS BYTE
00180      OR      E          ;MS BYTE
00190      JR      NZ,NEXT    ;IFDE>0THEN NEXT X
00200      RET     ;RETURN
00210      END     ;END
    
```

All those who see any similarity between the BASIC and the machine-language programs, please raise your hands! They look a lot like each other to me, but the machine-language version deals with much smaller operations than the BASIC version. The following are a few commands in machine-language and BASIC code which are very close in meaning.

Figure 17.3 Machine-language and BASIC Commands

```

LD HL,15360 = LET HL=15360
LD (HL),A   = POKE HL,A
RET         = RETURN
END        = END
DEC HL     = HL=HL-1
INC HL     = HL=HL+1
JP        = GOTO
CALL      = GOSUB
JR NZ,NAME = NOT ZERO THEN GOTO NAME
JR NZ,BC00H = NOT ZERO THEN GOTO MEMORY LOCATION
    
```

One of the best ways to learn machine-language is by looking at programs in magazines and comparing the op-codes with the remarks of the programmer.

### The Merging of BASIC and Machine-code

The following programming utility will make things a little easier for you. It takes machine-code subroutines that you find or write and converts them into packed strings.

There are a couple of points I should mention before we continue. The first is that when you select machine-language code for use in your packed strings, you have to avoid using any 0's or 34's because these numbers will terminate a packed string faster than a herd of turtles!

Now comes the tricky part. How do we express a number like 15360 in our packed string? This number in our data line should look like this:

```
100 DATA 33,0,60
```

This means LD HL,15360. To do this without using any 0's is easy. All we have to do is use DEC or INC, depending on which way we want to work. Now our data line would look like the following:

```
100 DATA 33,1,60,43
```

## Machine-language Subroutine Packer

All we are doing is loading HL with 15361 and then immediately decrementing HL so that HL holds 15360, but our string contains a 1 instead of a 0 byte, which would terminate the packed string. Now the machine-language program will look like the following when completed.

---

Figure 17.4 *Machine-language/Packed String Converter*

```
00100      ORG      32000
00110      LD       HL,15361
00120      DEC      HL
00130      LD       DE,1025
00140      DEC      DE
00150 NEXT  LD       A,191
00160      LD       (HL),A
00170      INC      HL
00180      DEC      DE
00190      LD       A,D
00200      OR       E
00210      JR       NZ,NEXT
00220      RET
00230      END
```

---

Another rule we need to remember is not to use the same string variable anywhere else in our program.

## Machine-language Subroutine Packer

---

Figure 17.5 *Machine-Language Subroutine*

```
1000 L=50000
1010 Z=PEEK(16549)*256+PEEK(16548)
1020 IFZ>32767THENZ1=-1*(65536-Z):Z=Z1
1030 IFPEEK(Z+2)+PEEK(Z+3)*256=LTHENL=Z+4:GOTO1060
1040 IFPEEK(Z+2)+PEEK(Z+3)*256>LSTOP
1050 Z=PEEK(Z)+PEEK(Z+1)*256:GOTO1020
1060 P=PEEK(L):IFP<>88THENL=L+1:GOTO1060
1070 READD:IFD=-1THEN2000ELSEPEEK(L,D):L=L+1:GOTO1070
1080 DATA 1,1,4,11,33,255,59,35,126,47,203,255,203,183
1090 DATA 119,11,120,177,32,243,201,-1
2000 GOSUB50000:DELETE1000-2000
50000 M$="XXXXXXXXXXXXXXXXXXXXX":GOTO60000
60000 Z=VARPTR(M$):DEFUSR0=PEEK(Z+1)+256*PEEK(Z+2):RETURN
```

---

To use this, just enter the data values for the machine-language routine into line 1080 and insert the same number of X's into line 50000. Remember not to use any 0's in the data lines because that will terminate the string.

Now RUN the program, LIST it, make the USR call, and the video should reverse. Tape users will have to change the DEFUSR to your usual POKE16526,LS:POKE16527,MS. To do this, you can change line 60000 to the following:

```
60000 Z=VARPTR(M$):POKE16526,PEEK(Z+1):POKE16527,PEEK(Z+2)
:RETURN
```

The best way to use this utility is to save it after you have packed your first machine-language subroutine. Each time you write a new routine, renumber lines 50000 to 50010, 50020, etc., so that each subroutine is in increments of 10.

After you have saved the first subroutine, you no longer need to have line 60000 in the following subroutines, so delete this line for each subsequent one. Each time you want to write a program, just load in the first subroutine and merge in all the rest. Each subroutine can use the same line 60000.

If your system won't allow you to merge programs, you can use the code below. First find where your BASIC program starts in memory. In the command mode, enter:

```
PRINTPEEK(16548);PEEK(16549)
```

Next, save these two numbers to POKE back into the second command statement below. Load the first subroutine and in the command mode, type:

```
POKE16548,PEEK(16633)-2:POKE16549,PEEK(16634) <ENTER>
```

Now load in the next subroutine and type:

```
POKE16548,233:POKE16549,66 <ENTER>
```

Repeat these steps for each subroutine you want to load. Better yet, why not write a machine-language subroutine to do this for you?

The following are a few machine-language subroutines that have been converted to be compatible with string packing.

### LDVIDEO

This subroutine lets you fill the screen with any character you wish; it is especially handy for a graphic clear character 128. To call this routine, load the USR call with the value to pass to the routine, such as A=USR(128).

```
180 DATA 205,127,10,229,209,33,1,60,43,1,255,3,3,115,11,
35,120,177,32,249,201
```

### REVIDEO

This subroutine does a reverse video and can be called by A=USR(0).

```
180 DATA 1,1,4,11,33,255,59,35,126,47,203,255,203,183,119,
11,120,177,32,243,201
```

## SAVETAPE

### SOUND

This subroutine can be called by  $A = \text{USR}(\text{XXXX})$ , where XXXX is any number between 0 and 9999.

```
180 DATA 205,127,10,77,68,62,1,105,211,255,45,32,253,60,  
105,211,255,45,32,253,13,16,238,175,211,255,201
```

### RSCROLL

This subroutine does a right-video scroll and can be called by  $A = \text{USR}(0)$ .

```
180 DATA 33,1,251,17,1,60,27,1,1,4,11,237,176,201
```

### SCRMEM48

This subroutine transfers the screen to high memory for a 48K machine and can be called by  $A = \text{USR}(0)$ .

```
180 DATA 33,1,60,43,17,1,251,1,1,4,11,237,176,201
```

### MEMSCR48

This subroutine moves high memory back to the screen for a 48K machine and is called by  $A = \text{USR}(0)$ .

```
180 DATA 33,1,251,17,1,60,27,1,1,4,11,237,176,201
```

### SCRMEM16

This subroutine transfers the screen to high memory for 16K machines and is called by  $A = \text{USR}(0)$ .

```
180 DATA 33,1,60,43,17,254,123,1,1,4,11,237,176,201
```

### MEMSCR16

This subroutine moves the high memory back again to the screen for 16K machines and is called by  $A = \text{USR}(0)$ .

```
180 DATA 33,254,123,17,1,60,27,1,1,4,11,237,176,201
```

### LOADTAPE

This subroutine loads a screen dump from tape back into the screen and is called by  $A = \text{USR}(0)$ .

```
180 DATA 175,205,18,2,205,150,2,33,1,60,43,1,1,4,11,205,53,  
2,119,35,11,120,177,32,246,205,248,1,201
```

### SAVETAPE

This subroutine dumps the screen to tape and can be called by  $A = \text{USR}(0)$ .

```
180 DATA 175,205,18,2,205,135,2,33,1,60,43,1,1,4,11,126,205,  
100,2,35,11,120,177,32,246,205,248,1,201
```

Keep adding to this file and before you know it, you'll have a subroutine to do anything you want in machine-language, and you will never have to write the code again!

---

# 18

---

## Computer Animation

The word *animation* has always had a sort of mystical sound to me, — a word with a little magical meaning to it. The dictionary defines animation as “the state of possessing life or spirit, to give natural life to, to look alive.”

The object of this chapter is to use a method of programming to make our graphics look alive, or give animation to our graphic characters. The chapter will also provide a method of creating our characters to make it easier for us to give them animation.

## Character Generators

Character generators are programs that help us create our graphic characters. In order to create the appearance of animation, we must be able to see our character in slightly different positions on the screen and at the same time keep changing the appearance of the character.

Perhaps many of you have made things called flip cards, in which you take a small pad of paper and draw pictures on each page, changing each picture just a little. By flipping the pages slowly, you could create the appearance of the character moving. This illusion is animation, and we will be trying to re-create this effect using our computers instead of a pad of paper.

There are three qualities we shall be dealing with a lot in our programming. One is speed of execution, another is ease of programming and last (but not least) is realistic-looking characters. These three attributes are not the most compatible to try using at the same time, but we'll do our best using only BASIC code.

## The Graphics Generator

The creation of graphic characters for animation is perhaps one of the most time-consuming parts of programming. To help with this task, let's first take a look at some facts about characters that we would most likely use in our games.

The size of our characters is usually going to stay within 5 or 6 print positions across by as high as 3 lines on the video. This is all the more true if we want more than one character on the screen at a time. These dimensions will give a character just about the size shown below.

```
XXXXXX  
XXXXXX  
XXXXXX
```

This is quite large enough to create almost any character we might need. If you don't believe it, just take a look at some of the best-selling games you have purchased lately.

All we need now is a program we can use to draw the characters and another to give our characters animation. How about a program that does both? Well step right up, — it has already been written.

### **STRING WRITER**

STRING WRITER can create up to ten graphic characters for us. At the same time, we don't have to worry about all the embedded control codes such as line feeds and backspaces. Even more importantly, we can edit the lines if we need to! STRING WRITER also gives us animation for our characters.

Remember any of my favorite sayings? If you don't, I'll tell you. It's, "Make the computer do the hard work."

To use the STRING WRITER utility, press the arrow keys for the direction in which you want to draw. Use the arrow and space bar to move or erase with. Below is a list of the control keys and what each does.

- (1) Write graphics into memory. Shows line number and string number. This control does not clear your picture. The reason for this is that most of your character will remain the same from frame to frame. You will only be making slight changes to each picture to create animation.
- (2) Clear the picture. This key won't be used much because it's usually more convenient to remove only the parts you don't want by pressing the arrow and space bar.
- (3) Start over. This key clears the string counter and memory address to allow you to start again at line 1000. You might notice that the strings I used to print the title "STRING WRITER" on the screen are the same strings that will eventually hold your graphic characters. This method of programming helps me get more program into less memory.
- (4) Exit STRING WRITER program. This control removes everything from memory, leaving only your new graphic characters.

As you draw your characters, the program will display the X,Y coordinates on the screen. Most of the time, you will want to define one of your characters as a null so that when you want to move your character, you will have a blank character ready to remove the original character from the screen. To make a blank, or null character, hold down the shift key while you press the (1) key.

For an example of how this all works, let's copy the bird character from the video display sheet (see Figure 18.1) starting with the top left corner (bird @0) and working down, next the middle row from top to bottom, and last the end row from top to bottom. After you have saved the last bird, hold down the shift key and press the (1) key to make the last character a null.



We have now used all 10 characters, so the screen should clear and we can list our new program. You now have new graphic characters, labeled from A\$(0) through A\$(9). If you want to see the animation of your character, add the following code.

---

Figure 18.2 *Animation Code*

```
2000 CLS:S=460
2010 FORX=0TO8:PRINT@S,A$(X)
2020 SET(24,19):SET(30,14):SET(35,19)
2030 RESET(24,19):RESET(30,14):RESET(35,19)
2040 NEXT:GOTO2010
```

---

A circus dodo bird? If you would like to see him fly, enter the following code.

---

Figure 18.3 *Flight Code*

```
2000 CLS:S=350:FORX=0TO127:SET(X,0):SET(X,47):NEXT
2010 FORX=0TO47:SET(0,X):SET(1,X):SET(126,X):SET(127,X):NEXT
2020 A=RND(4):IFA=1THENZ=-64ELSEIFA=2THENZ=64
2030 IFA=3THENZ=-1ELSEZ=1
2040 IFA<3THENT1=0:T2=2ELSEIFA=3THENT1=3:T2=5ELSET1=6:T2=8
2050 FORX=T1TOT2:PRINT@S,A$(X);:NEXT:PRINT@S,A$(9);:L=S:S=S+Z
2060 IFPEEK(S)>128ORPEEK(S+15366)>128ORPEEK(S+15552)>128THENS=L
2070 GOTO2020
```

---

I hope the above gives you some idea of what can be done using the STRING WRITER utility program.

---

# NOTES

---

# 19

---

## **Compu / Mation**

Compu / Mation is actually the prototype of a program that writes complete games. Compu / Mation was created over three years ago and has now evolved into my machine-language game-generator. This version listed here has gone through about 10 revisions over the course of three years.

After learning how to use this version, you should be able to write about 80% of your game programs without ever having to type one line of BASIC code. Three years of learning the best ways to make use of Compu / Mation make it hard (if not impossible) for me to instruct you in all the possibilities this program has. In fact, I am currently writing a complete book about the newest version of this program. I will try to give you the information that you need to start using Compu / Mation, and perhaps you will be in a better position to work with the new version by the time my next book is done.

Compu / Mation is more than just a drawing program; it's a completely new method of getting fast graphics on your screen and animation into games. It's actually more a programming method than a stand-alone program.

## **What's in a Name?**

The name, Compu / Mation, means computer animation. I can honestly tell you (I have the publishers to prove it) that every game that Compu / Mation helped to write (over 80 complete games) has been sold! Let's see, that's 80 divided by 2 years, — that comes to 40 games a year. The reason I'm so slow is that in those two years I also had to learn how to program.

Now don't start thinking that I'm trying to impress everyone, — I'm not that kind of person. All this means is that I have devoted over 2 years to this program, and I'm proud of it. It's like getting a puppy and living with it day and night, watching it grow; you know what I mean?

## Operating Instructions

The only way to learn Compu / Mation is by using it, but I'll do my best to show you my way of using Compu / Mation. When running the program, the Compu / Mation name will appear on the screen. Please wait, because the program is loading in all the machine-language subroutines at this time.

After a few moments, the menu will appear on the screen. This is the main command mode! Let's go through each of the commands one at a time and see what they do.

### 1 = SAVE ON TAPE

This is the tape save mode. The screen will prompt you to press enter when ready to save graphics. You must have the tape recorder ready and in record mode before you press enter! The first thing that happens is that Compu / Mation moves your drawing from its working memory position on the screen. The graphics are saved on tape in a condensed format for speed and to use less room. The tape recorder will stop and the screen will clear and return you to the command mode.

### 2 = READING TAPE

Pressing the number (2) key puts you in the load from tape mode. Put the tape recorder in the play mode; then press the <ENTER> key. The graphics will load into screen memory and then be transferred into the working memory location.

### 3 = LINE PRINTER

This will dump graphics from the screen to your line printer. If you go into any mode and change your mind, just press the (C) key to clear the command and return to the menu.

### 4 = COMPU / MATION

This is the animation mode. Compu / Mation lets you save up to 14 frames or complete screens in memory at one time. If you press the number (4) key and wish to save the picture you're working on, then just press the number of the frame you wish to save it in. If this is the first frame in your animation, press number (1) and it will be saved in the first location.

If you wish to run the frames in memory like a motion picture, press the (0) Key and enter the total number of frames you wish to run.

To exit from the animation mode, wait until the frame you wish to work on is on the screen, hold down the <ENTER> key and continue pressing until you return to the menu. If you would like to adjust the speed of the animation, press the up arrow to go faster or the down arrow to decrease the speed.

### 5 = GRAPHIC MOVE

To move an object from one place to another on the screen, move the first flashing cursor with the arrow key and position it at the top, left corner of the object you wish to move. Press the <CLEAR> key, and you will see a second flashing cursor. Move this cursor to the bottom, right corner of the object to be moved and press the <CLEAR> key again.

After a few seconds the screen will become fuzzy, signifying that the object is ready to be moved. Use the arrow keys to move the object. When done, press the <ENTER> key to return to the menu. Each time you press the <ENTER> key, Compu / Mation will save the graphics on the screen to the working memory location and return you to the command menu.

**6 = TEXT INSERT**

This mode lets you type words into your graphics. Move the cursor with the arrow keys to one position before the spot where you wish to enter text. Next, press the <SPACE> bar to enter the text enter mode. Enter the line of text and press <ENTER> when done.

**7 = DRAWING MODE**

Pressing the number (7) key places you into the drawing mode of Compu / Mation. Use the arrow keys to draw, and use the arrow and space bar to move or erase. In the drawing mode, you have some additional control keys, described below.

Pressing the (Q) key will print the X and Y positions of the cursor.

Pressing the (L) key, moving the cursor to a new location and pressing the (L) key again will draw a line between two points on the screen.

Pressing the (F) key invokes the auto fill command. To use this command, draw a square on the screen and then place the cursor one pixel down from the top center of the square. Now press the (F) key. The object will be filled in, and the cursor will return to the top left corner of your screen. The auto fill can be used to fill any shape you wish. The only requirement is that the cursor must be able to drop straight down through the object it's filling without running into any lines.

As soon as the cursor reaches a bottom line, it cancels the command and returns to the top left corner of the screen. This command can also be used to draw a line across the screen from the left to right side.

Pressing the <CLEAR> key clears the screen and returns you to the drawing mode.

**8 = INVERT VIDEO**

After pressing the number (8) key and then the letter (R) key, the graphics will be inverted or reversed on the screen. Pressing (R) again restores them to the original configuration.

**9 = (DISK I/O)**

This mode lets you either save any frame to disk or load any frame from disk by filename. You can also read the directory on drive 1. This is a very fast disk save/load and l uses only one gran for each picture saved.

**0 = LARGE::PRINT**

## Operating Instructions

Pressing the (0) key places the picture in working memory onto the screen. The large print mode is good for making fancy-looking titles for your game programs. You can print 15 letters across the screen and 5 down. Pressing the (1) key moves you down one line or row of letters.

**: = SAVE : TOTAL**

This mode allows you to save all frames in memory at one time using the tape recorder.

**P = PACK STRINGS**

This is the heart of the Compu / Mation program. This mode will convert the graphics in working memory into packed strings starting at lines 3000 to 3050 using A1\$ through A6\$. After the screen has been converted, it will clear, and you will return to BASIC with only the packed strings left in memory. You can run these by adding the following lines:

```
1060 CLS:PRINTA1$;A2$;A3$;A4$;A5$;A6$;  
1070 GOTO 1070
```

You can save them to merge later into one large program. Now you're ready to explore all the different ways to use Compu / Mation. How about eye-catching store window displays to attract customers into the stores? I'll leave the rest up to your imagination.

Figure 19.1 *Compu/Mation Program*

```

10 DEF USR 1 = - 28672:
    DEF USR 2 = - 28652:
    DEF USR 3 = - 28630:
    DEF USR 4 = - 28603
20 REM *** COMPU / MATION BY TOM DEMPSEY ***
30 CLS :
    CLEAR 1000
40 REM *** REVERSE VIDEO ROUTINE ***
50 DATA 1,0,4,33,255,59,35,126,47,203,255,203,183,119,1
    1,120,177,32,243,201
60 FOR A = - 28672 TO - 28653:
    READ B:
    POKE A,B:
    NEXT
70 REM *** BLOCK MOVE ROUTINE ***
80 FOR A = - 28652 TO - 28634:
    READ B:
    POKE A,B:
    NEXT
90 DATA 17,0,60,33,136,144,1,0,4,26,119,35,19,11,120,17
    7,32,247,201
100 REM *** TAPE READ & SAVE ROUTINE ***
110 FOR A = - 28630 TO - 28577:
    READ B:
    POKE A,B:
    NEXT
120 DATA 175,205,18,2,205,150,2,33,0,60,1,0,4,205,53,2,1
    19,35,11,120,177,32,246,205,248,1,201,175,205,18,2,2
    05,135,2,33,0,60,1,0,4,126,205,100,2,35,11,120,177,3
    2,246,205,248,1,201
130 PRINT CHR$(23)
140 PRINT @462,"COMPU / MATION"
150 PRINT @518,"(C)1981 BY TOM DEMPSEY"
160 FOR X = - 28536 TO - 27513:
    POKE X,128:
    NEXT
170 GOTO 480
-----
180 A = 14400
190 Y = 0:
    X = 0
200 RESET (X,Y):
    SET (X,Y)
210 IF PEEK (14340) = 2
    THEN I$ = "":
        FOR Q = 15360 TO 15423:
            P = PEEK (Q):
            I$ = I$ + CHR$(P):
        NEXT Q:
        PRINT @0,"X=";X;" Y=";Y;:

```

```

FOR Q = 1 TO 1000:
  NEXT Q:
  PRINT @0, I$;:
  I$ = ""
220 IF PEEK (A) = 8Y = Y - 1
230 IF PEEK (A) = 16Y = Y + 1
240 IF PEEK (A) = 32X = X - 1
250 IF PEEK (A) = 64X = X + 1
260 IF PEEK (A) = 136:
  GOSUB 370:
  RESET (X, Y):
  Y = Y - 1
270 IF PEEK (A) = 144:
  GOSUB 370:
  RESET (X, Y):
  Y = Y + 1
280 IF PEEK (A) = 160:
  GOSUB 370:
  RESET (X, Y):
  X = X - 1
290 IF PEEK (A) = 192:
  GOSUB 370:
  RESET (X, Y):
  X = X + 1
300 IF Y < 0Y = 0:
  ELSE IF Y > 47Y = 47:
  ELSE IF X < 0X = 0:
  ELSE IF X > 127X = 127
310 IF PEEK (A) = 2 FOR M = 15360 TO 16383:
  POKE M, 128:
  NEXT :
  X = 0:
  Y = 0
320 IF PEEK (A) = 1
  THEN 460 .....
330 IF PEEK (14337) = 8V = USR 2(X):
  GOSUB 760:
  A = 14400
340 IF PEEK (14337) = 64
  THEN 390 .....
350 IF PEEK (14338) = 16 GOSUB 2730:
  RESET (X, Y):
  FOR T = 1 TO 100:
  NEXT :
  SET (X, Y)
360 GOTO 200 .....
-----
370 IF Y < 0Y = 0:
  ELSE IF Y > 47Y = 47:
  ELSE IF X < 0X = 0:

```

```

                                     ELSE IF X > 127X = 1
                                     27
380 RETURN .....
-----
390 A = X:
400 B = Y
    X = X + 1:
    GOSUB 440:
    IF POINT (X,Y)
    THEN 410: .....
    ELSE SET (X,Y):
    GOTO 400 .....
-----
410 X = A
420 X = X - 1:
    GOSUB 440:
    IF POINT (X,Y)
    THEN 430: .....
    ELSE SET (X,Y):
    GOTO 420 .....
-----
430 X = A:
    Y = Y + 1:
    GOSUB 440:
    IF POINT (X,Y)
    THEN 180: .....
    ELSE SET (X,Y):
    GOTO 400 .....
-----
440 IF X < 0 OR X > 127 OR Y > 47
    THEN 180: .....
    ELSE RETURN .....
-----
450 REM *** SAVE IN MEMORY ***
460 A =USR 2(X)
470 REM *** COMMAND MODE ROUTINE **
    *
480 CLS
490 PRINT "* COMMAND MODE *":
    PRINT
500 PRINT "1 = SAVE ON TAPE"
510 PRINT "2 = READING TAPE"
520 PRINT "3 = LINE PRINTER"
530 PRINT "4 = COMPU/MATION"
540 PRINT "5 = GRAPHIC MOVE"
550 PRINT "6 = TEXT INSERT"
560 PRINT "7 = DRAWING MODE"
570 PRINT "8 = INVERT VIDEO"
580 PRINT "9 = ( DISK I/O )"
590 PRINT "0 = LARGE: :PRINT"
600 PRINT ": = SAVE : TOTAL"

```

Operating Instructions

```

610      PRINT "P = PACK STRINGS"
620      I$ = INKEY$ :
          IF I$ = ""
          THEN 620 .....
630      IF I$ = "P"
          THEN 2660 .....
640      IF I$ = "0"
          THEN 2190 .....
650      IF I$ = ":"
          THEN 2530 .....
660      IF I$ = "7"
          THEN GOSUB 2130:
          GOTO 180 .....
670      ON VAL (I$) GOTO 690,850,910,980
          ,1410,1730,1920,1980 .....
-----
680      GOTO 620 .....
-----
690      REM *** TAPE SAVE ROUTINE ***
700      CLS :
          PRINT "WHEN READY TO SAVE GRAPHI
          CS HIT ENTER"
710      I$ = INKEY$ :
          IF I$ = ""
          THEN 710 .....
720      IF I$ = "C" GOSUB 2130:
          GOTO 180 .....
730      GOSUB 2130
740      A =USR 4(X)
750      GOTO 470 .....
-----
760      CLS
770      INPUT "X,Y CENTER OF CIRCLE";X1,
          Y1
780      INPUT "SIZE OR RADIUS OF CIRCLE"
          ;R
790      GOSUB 2130:
          E = 128 / 48
800      FOR T = 0 TO 6.3 STEP 1 / (R +
          R)
810          X = INT (E * R * COS (T) +
          X1 + 0.5)
820          Y = INT (R * SIN (T) + Y1 +
          0.5)
830          IF X < 0 OR X > 127 OR Y <
          0 OR Y > 47
          THEN NEXT T
840          SET (X,Y):
          NEXT T:
          RETURN .....
-----

```

Operating Instructions

```

850      REM *** TAPE READ ROUTINE ***
860      CLS :
          PRINT "WHEN READY TO READ TAPE H
          IT ENTER"
870      I$ = INKEY$ :
          IF I$ = ""
880      THEN 870 .....
          IF I$ = "C" GOSUB 2130:
890      GOTO 180 .....
          A =USR 3(X)
900      GOTO 450 .....
-----
910      REM *** LINEPRINTER ROUTINE ***

920      CLS :
          PRINT "WHEN READY TO LINEPRINT H
          IT ENTER"
930      I$ = INKEY$ :
          IF I$ = ""
940      THEN 930 .....
          IF I$ = "C" GOSUB 2130:
950      GOTO 180 .....
          GOSUB 2130:
960      LPRINT CHR$(27); CHR$(66)
          FOR X = 15360 TO 16383:
              P = PEEK (X):
              LPRINT CHR$(P);:
          NEXT
970      GOTO 180 .....
-----
980      REM *** COMPU / MATION ***
990      CLS :
1000     PRINT "COMPU / MATION"
          PRINT "ENTER FRAME NUMBER (1-14)
          / ENTER 0 TO RUN"
1010     INPUT I$
1020     IF I$ = "0"
1030     THEN 1230 .....
          GOSUB 2130
1040     B = - 28652
1050     POKE B + 4,0
1060     IF I$ = "1" POKE B + 5,192
1070     IF I$ = "2" POKE B + 5,196
1080     IF I$ = "3" POKE B + 5,200
1090     IF I$ = "4" POKE B + 5,204
1100     IF I$ = "5" POKE B + 5,208
1110     IF I$ = "6" POKE B + 5,212
1120     IF I$ = "7" POKE B + 5,216
1130     IF I$ = "8" POKE B + 5,220
1140     IF I$ = "9" POKE B + 5,224
1150     IF I$ = "10" POKE B + 5,228

```

## Operating Instructions

```

1160      IF  I$ = "11" POKE B + 5,232
1170      IF  I$ = "12" POKE B + 5,236
1180      IF  I$ = "13" POKE B + 5,240
1190      IF  I$ = "14" POKE B + 5,244
1200      A =USR 2(X):
          CLS
1210      POKE B + 4,136:
          POKE B + 5,144
1220      GOTO 470 .....
-----
1230      REM  *** RUN ANIMATION ***
1240      CLS :
          PRINT "* ANIMATION MODE *"
1250      PRINT :
          INPUT "ENTER NUMBER OF FRAMES IN
              MEMORY";I
1260      Q = 0
1270      B = - 28652
1280      POKE B + 9,237:
          POKE B + 10,176:
          POKE B + 11,201
1290      Z = 192:
          POKE B + 4,0:
          T = 500
          CLS
1300      POKE B + 5,Z:
1310      Z = Z + 4:
          A =USR 2(X)
1320      FOR X = 1 TO T:
          NEXT
1330      IF  PEEK (14400) = 8T = T - 100
          :
          IF  T < 0T = 0
1340      IF  PEEK (14400) = 16T = T + 10
          0:
          IF  T > 1000T = 1000
1350      IF  PEEK (14400) = 1
          THEN 1380 .....
1360      Q = Q + 1:
          IF  Q = I
          THEN Q = 0:
          Z = 192
1370      GOTO 1310 .....
-----
1380      Q = 0:
          POKE B + 9,26:
          POKE B + 10,119:
          POKE B + 11,35
1390      POKE B + 4,136:
          POKE B + 5,144
1400      GOTO 450 .....
-----

```

Operating Instructions

```

1410      REM *** GRAPHIC MOVE MODE ***
1420      CLS :
          PRINT "*" GRAPHIC MOVE MODE *":
          FOR X = 1 TO 1000:
          NEXT :
          CLS
1430      GOSUB 2130
1440      A = 15360
1450      P = PEEK (A):
          POKE A,191:
          FOR X = 1 TO 10:
          NEXT :
          POKE A,P
1460      IF PEEK (14400) = 8A = A - 64
1470      IF PEEK (14400) = 16A = A + 64

1480      IF PEEK (14400) = 32A = A - 1
1490      IF PEEK (14400) = 64A = A + 1
1500      IF A < 15360A = 15360:
          ELSE IF A > 16320A = 16320
1510      IF PEEK (14400) = 2
          THEN POKE 14400,0:
          GOTO 1530 .....
1520      GOTO 1450 .....
-----
1530      B = 15360
1540      P = PEEK (B):
          POKE B,191:
          FOR X = 1 TO 10:
          NEXT :
          POKE B,P
1550      IF PEEK (14400) = 8B = B - 64
1560      IF PEEK (14400) = 16B = B + 64

1570      IF PEEK (14400) = 32B = B - 1
1580      IF PEEK (14400) = 64B = B + 1
1590      IF B < 15360B = 15360:
          ELSE IF B > 16320B = 16320
1600      IF PEEK (14400) = 2
          THEN 1620 .....
1610      GOTO 1540 .....
-----
1620      FOR X = A TO B:
          P = PEEK (X):
          M$ = M$ + CHR$ (P):
          IF LEN (M$) > 245
          THEN M$ = "":
          GOTO 450 .....

1630      NEXT X
1640      C = A - 15360:
          PRINT @C,M$;

```

## Operating Instructions

```

1650      IF   PEEK (14400) = 8A = A - 64
1660      IF   PEEK (14400) = 16A = A + 64

1670      IF   PEEK (14400) = 32A = A - 1
1680      IF   PEEK (14400) = 64A = A + 1
1690      IF   A < 15360A = 15360:
1700      ELSE IF  A > 16320A = 16320
1710      IF   PEEK (14400) = 1
1720      THEN M$ = "":
1730      GOTO 450 .....
1740      IF   PEEK (14400) = 2
1750      THEN 1440 .....
1760      GOTO 1640 .....
-----
1730      REM   *** TEXT INSERT MODE ***
1740      CLS :
1750      PRINT "* TEXT INSERT MODE *":
1760      FOR X = 1 TO 1000:
1770      NEXT :
1780      CLS
1790      GOSUB 2130
1800      A = 15360
1810      P = PEEK (A):
1820      POKE A,143:
1830      POKE A,P
1840      B = A
1850      IF   PEEK (14400) = 8A = A - 64
1860      IF   PEEK (14400) = 16A = A + 64

1810      IF   PEEK (14400) = 32A = A - 1
1820      IF   PEEK (14400) = 64A = A + 1
1830      IF   A < 15360A = B:
1840      ELSE IF  A > 16383A = B
1850      IF   PEEK (14400) = 1
1860      THEN 450 .....
1870      IF   PEEK (14400) = 128
1880      THEN 1870 .....
1890      GOTO 1770 .....
-----
1870      I$ = "":
1880      I$ = INKEY$ :
1890      IF   I$ = "":
1900      THEN 1870 .....
1910      IF   PEEK (14400) = 1
1920      THEN 450 .....
1930      POKE A, ASC (I$):
1940      A = A + 1
1950      GOTO 1870 .....
-----
1910      REM   *** CALL REVERSE VIDEO ***
1920      CLS :

```

```

1930      PRINT "* INVERT VIDEO MODE *"
          FOR X = 1 TO 1000:
1940      NEXT
          GOSUB 2130
1950      IF PEEK (14340) = 4A = USR 1(X
          )
1960      IF PEEK (14400) = 1
          THEN 460 .....
1970      GOTO 1950 .....
-----
1980      CLS
1990      PRINT :
          PRINT "PRESS (S) = SAVE (L) = L
          OAD (D) = DIR (C) = COMMAND MO
          DE"
2000      I$ = INKEY$ :
          IF I$ = "S"
          THEN 2010: .....
          ELSE IF I$ = "L"
          THEN 2070: .....
          ELSE IF I$ = "C"
          THEN 480: .....
          ELSE IF I$ = "D"
          THEN CMD "DIR :1"
          :
          : GOTO 1990: .....
          : .....
          ELSE 2000 .....
2010      INPUT "ENTER FILESPEC TO SAVE ";
          B$
2020      F$ = "DUMP " + B$ + ":1 15360 16
          383 26810"
2030      GOSUB 2130
2040      CMD "F$
2050      GOTO 470 .....
-----
2060      REM          LOAD FROM DI
          SK
2070      REM
2080      INPUT "ENTER FILESPEC TO LOAD ";
          B$
2090      CLS
2100      F$ = "LOAD " + B$
2110      CMD "F$
2120      A = USR 2(X):
          GOTO 470 .....
-----
2130      REM *** BLOCK MOVE TO SCREEN **
          *
2140      B = - 28652
2150      POKE B + 9,237:

```

Operating Instructions

```

                POKE B + 10,176:
                POKE B + 11,201
2160           A = USR 2(X)
2170           POKE B + 9,26:
                POKE B + 10,119:
                POKE B + 11,35
2180           RETURN .....
-----
2190           GOSUB 2130:
                ' *** LARGE PRINT MODE ***
2200           CLEAR 1000:
                M = 15360
2210           T = M
2220           A$ = INKEY$ :
                IF A$ = ""
                THEN 2220 .....
2230           IF A$ = "A" POKE M,170:
                POKE M + 1,131:
                POKE M + 2,171:
                POKE M + 64,191:
                POKE M + 65,131:
                POKE M + 66,171
2240           IF A$ = "B" POKE M,191:
                POKE M + 1,131:
                POKE M + 2,149:
                POKE M + 64,191:
                POKE M + 65,179:
                POKE M + 66,187
2250           IF A$ = "C" POKE M,151:
                POKE M + 1,131:
                POKE M + 2,143:
                POKE M + 64,181:
                POKE M + 65,176:
                POKE M + 66,184
2260           IF A$ = "D" POKE M,191:
                POKE M + 1,131:
                POKE M + 2,169:
                POKE M + 64,191:
                POKE M + 65,176:
                POKE M + 66,186
2270           IF A$ = "E" POKE M,170:
                POKE M + 1,131:
                POKE M + 2,131:
                POKE M + 64,191:
                POKE M + 65,179:
                POKE M + 66,176
2280           IF A$ = "F" POKE M,170:
                POKE M + 1,131:
                POKE M + 2,131:
                POKE M + 64,191:
                POKE M + 65,131

```

## Operating Instructions

```

2290      IF  A$ = "G" POKE M,151:
           POKE M + 1,131:
           POKE M + 2,139:
           POKE M + 64,181:
           POKE M + 65,178:
           POKE M + 66,191
2300      IF  A$ = "H" POKE M,191:
           POKE M + 2,170:
           POKE M + 64,191:
           POKE M + 65,131:
           POKE M + 66,191
2310      IF  A$ = "I" POKE M,170:
           POKE M + 64,191:
           M = M - 2
2320      IF  A$ = "J" POKE M,131:
           POKE M + 1,191:
           POKE M + 2,131:
           POKE M + 64,180:
           POKE M + 65,191
2330      IF  A$ = "K" POKE M,191:
           POKE M + 1,160:
           POKE M + 2,159:
           POKE M + 64,191:
           POKE M + 65,131:
           POKE M + 66,189
2340      IF  A$ = "L" POKE M,170:
           POKE M + 64,191:
           POKE M + 65,176:
           POKE M + 66,176
2350      IF  A$ = "M" POKE M,191:
           POKE M + 1,171:
           POKE M + 2,171:
           POKE M + 64,191:
           POKE M + 66,190
2360      IF  A$ = "N" POKE M,191:
           POKE M + 1,175:
           POKE M + 2,170:
           POKE M + 64,191:
           POKE M + 66,191
2370      IF  A$ = "O" POKE M,170:
           POKE M + 1,131:
           POKE M + 2,171:
           POKE M + 64,191:
           POKE M + 65,176:
           POKE M + 66,186
2380      IF  A$ = "P" POKE M,151:
           POKE M + 1,131:
           POKE M + 2,171:
           POKE M + 64,191:
           POKE M + 65,131:
           POKE M + 66,131

```

## Operating Instructions

2390	IF	A\$ = "Q" POKE M,151: POKE M + 1,131: POKE M + 2,149: POKE M + 64,181: POKE M + 65,187: POKE M + 66,181
2400	IF	A\$ = "R" POKE M,191: POKE M + 1,179: POKE M + 2,191: POKE M + 64,191: POKE M + 65,131: POKE M + 66,189
2410	IF	A\$ = "S" POKE M,183: POKE M + 1,179: POKE M + 2,179: POKE M + 64,188: POKE M + 65,176: POKE M + 66,186
2420	IF	A\$ = "T" POKE M,131: POKE M + 1,171: POKE M + 2,131: POKE M + 65,191
2430	IF	A\$ = "U" POKE M,170: POKE M + 2,170: POKE M + 64,191: POKE M + 65,176: POKE M + 66,186
2440	IF	A\$ = "V" POKE M,149: POKE M + 2,186: POKE M + 64,141: POKE M + 65,176: POKE M + 66,143
2450	IF	A\$ = "W" POKE M,191: POKE M + 2,175: POKE M + 64,191: POKE M + 65,186: POKE M + 66,186
2460	IF	A\$ = "X" POKE M,175: POKE M + 1,176: POKE M + 2,159: POKE M + 64,190: POKE M + 65,131: POKE M + 66,189
2470	IF	A\$ = "Y" POKE M,159: POKE M + 2,191: POKE M + 64,179: POKE M + 65,179: POKE M + 66,191
2480	IF	A\$ = "Z" POKE M,131: POKE M + 1,179: POKE M + 2,143:

```

                POKE M + 64,191:
                POKE M + 65,188:
                POKE M + 66,188
2490          IF A$ = CHR$ (13)
                THEN 450 .....
2500          IF A$ = "1" T = T + 192:
                M = T - 4
2510          M = M + 4
2520          GOTO 2220 .....
-----
2530          CLS
2540          INPUT "ENTER TOTAL NUMBER TO SAV
                E";I:
                CLS
2550          Q = 0
2560          B = - 28652
2570          POKE B + 9,237:
                POKE B + 10,176:
                POKE B + 11,201
2580          POKE B + 4,0
2590          Z = 192
2600          FOR X = 1 TO I
2610             POKE B + 5,Z:
                A = USR 2(X)
                A = USR 4(X)
                Z = Z + 4:
                NEXT X
2640          Q = 0:
                POKE B + 9,26:
                POKE B + 10,119:
                POKE B + 11,35
2650          POKE B + 4,136:
                POKE B + 5,144:
                GOTO 470 .....
-----
2660          GOSUB 2130:
                A = - 30053
2670          FOR X = 15360 TO 16382
2680             IF X > 15375 PRINT @0,"I'
                M PACKING ";
2690             B = PEEK (X)
2700             C = PEEK (A):
                IF C < > 88
                THEN A = A + 1:
                GOTO 2700 .....
2710             IF C = 88 POKE A,B:
                A = A + 1:
                NEXT X:
                CLS
2720          DELETE 10 - 2970
2730          IF L1 = 0

```

Operating Instructions

```

THEN L1 = 1:
      A1 = X:
      B = Y:
      RETURN .....
2740 IF L1 = 1
      THEN L1 = 0:
           C = X:
           D = Y
2750 IF A1 = C
      THEN RETURN : .....
      ELSE IF B = D
           THEN RETURN .....
2760 IF ABS (C - A1) < ABS (D -
           B) GOTO 2880 .....
2770 Z = (D - B) / ABS (C - A1)
2780 IF C > A1 GOTO 2830 .....
2790 FOR I = A1 TO C STEP - 1
2800     SET (I,B):
           B = B + Z:
           IF B < 0
           THEN B = 0
2810 NEXT I
2820 RETURN .....
-----
2830 FOR I = A1 TO C
2840     SET (I,B)
2850     B = B + Z:
           IF B < 0
           THEN B = 0
2860 NEXT I
2870 RETURN .....
-----
2880 Z = (C - A1) / ABS (C - B):
2890 IF D > B GOTO 2940 .....
2900 FOR I = B TO D STEP - 1
2910     SET (A1,I)
           A1 = A1 + Z:
           IF A1 < 0
           THEN A1 = 0
2920 NEXT I
2930 RETURN .....
-----
2940 FOR I = B TO D
2950     SET (A1,I)
2960     A1 = A1 + Z:
           IF A1 < 0
           THEN A1 = 0
2970 NEXT I
2980 RETURN .....

```

```

2990      A1$ = "XXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXX"
3000      A2$ = "XXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXX"
3010      A3$ = "XXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXX"
3020      A4$ = "XXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXX"
3030      A5$ = "XXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXX"
3040      A6$ = "XXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"

```

---

# NOTES

---

# List of Games and Graphics

---

ARCADE .....	171
AUTOGRAF .....	113
COMPUMAT .....	219
COPY .....	195
COWBOY .....	52
DATAGRAF .....	116
DISKFILE .....	190
ENLARGE .....	193
HORSE .....	62
INVASION .....	160
JAWS .....	28
LINEFIND .....	188
LINEMOD .....	186
LISA .....	134
LSET1K .....	182
MACHINE .....	23
MAKEMAZE .....	129
MATH .....	144
MEMORY .....	149
MORSE .....	147
NEWSLETT .....	41
PCOPY .....	197
POKER .....	88
RACETRAC .....	67
ROADSTER .....	138
ROBOT .....	37
SAUCER .....	163
SER/REC .....	178
SILVER .....	147

List Of Games

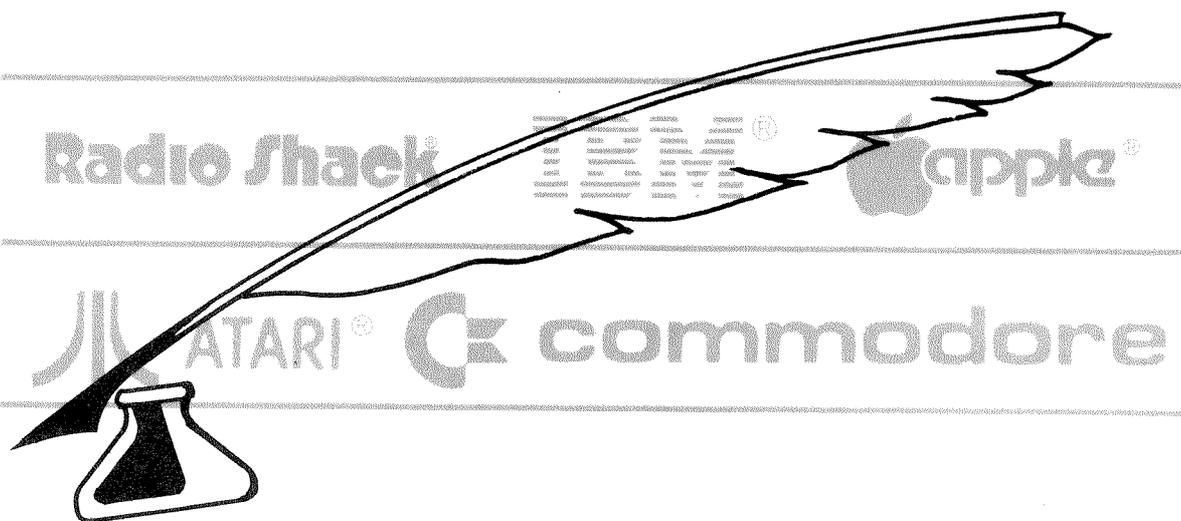
SLOT .....	48
SPEED .....	167
STATES .....	78
STRAW .....	157
STRING/FIX .....	183
TANK .....	58
TREASURE .....	154
TWITS .....	60
UNPACK .....	185
WRITER .....	210
YAHTZEE .....	95

---

# NOTES

# NOTES

# WANTED



## Book & Software AUTHORS

IJG INC. has become a world wide recognized leader in computer publishing. We take pride in publishing only the best in computer oriented books and software. If you have an idea, and really know your subject, we would like to talk with you.

Qualifying manuscripts once submitted, will be read and evaluated by our professional editorial staff (who are themselves published authors), and a few selected writers will be invited in for a personal evaluation of their work.

Contact Mr. Harvard Pennington or Mr. David Moore.



1953 West  
11th Street  
Upland, CA  
91786 (714)  
946-5805

# JG Computer Books & Software



Microsoft trademark Microsoft Corporation  
 Apple trademark Apple Computer Inc.  
 TRS-80 trademark TANDY Corporation  
 Electric Pencil © 1981 Michael Shrayer

## BOOKS

**TRS-80 Disk & Other Mysteries.** H.C. Pennington. The "How to" book of Data Recovery. 128 pages. **\$22.50**

**Microsoft Basic Decoded & Other Mysteries.** James Farvour. The Complete Guide to Level II Operating Systems & BASIC. 312 pages ..... **\$29.95**

**The Custom TRS-80 & Other Mysteries.** Dennis Bathory Kitz. The Complete Guide to Customizing TRS-80 Software & Hardware. 336 pages ..... **\$29.95**

**BASIC Faster & Better & Other Mysteries.** Lewis Rosenfelder. The Complete Guide to BASIC Programming Tricks & Techniques. 290 pages .... **\$29.95**

**Electric Pencil Operators Manual.** Michael Shrayer. Electric Pencil Word Processing System Manual. 123 pages. .... **\$24.95**

**The Custom Apple.** Winfried Hofacker & Ekkehard Floegel. The Complete Guide to Customizing the Apple Software and Hardware. 190 pages ..... **\$24.95**

**Machine Language Disk I/O & Other Mysteries.** Michael D. Wagner. A Guide to Machine Language Disk I/O for the TRS-80 Models I and II. Available October 1982 ..... **\$29.95**

**TRSDOS 2.3 Decoded & Other Mysteries.** James Lee Farvour. Commented Guide to TRSDOS 2.3 for the Model I. Available December 1982 ..... **\$29.95**

## SOFTWARE

**Electric Pencil.** Michael Shrayer. Word Processing System. Available in DISK ..... **\$89.95**  
 STRINGY FLOPPY or CASSETTE ..... **\$79.95**

**Red Pencil.** Automatic Spelling Correction Program. For use with the Electric Pencil Word Processing System. Available in DISK ONLY ..... **\$89.95**

**Blue Pencil.** \* Dictionary - Proofing Program. For use with the Electric Pencil Word Processing System. Available in DISK ONLY ..... **\$89.95**

\* Requires **Red Pencil** program.

**BFBLIB.** Lewis Rosenfelder. Basic Faster & Better Library Disk. 32 Demonstration Programs. Basic Overlays. Video Handlers. Sorts & more for the Model I & II. Available in DISK ONLY ..... **\$19.95**

**BFBDEM.** Lewis Rosenfelder. Basic Faster & Better Demonstration Disk. 121 Functions, Subroutines & User Routines for the TRS-80 Model I & II. Available in DISK ONLY ..... **\$19.95**

Prices Subject to change without notice

Add \$4.00 shipping and handling charge per item.

California residents add 6% sales tax. Canadian residents add 20% for exchange rate.

**JG** 1953 West  
 11th Street  
 Upland, CA  
 91786 (714)  
 946-5805



ISBN 0-936200-10-3