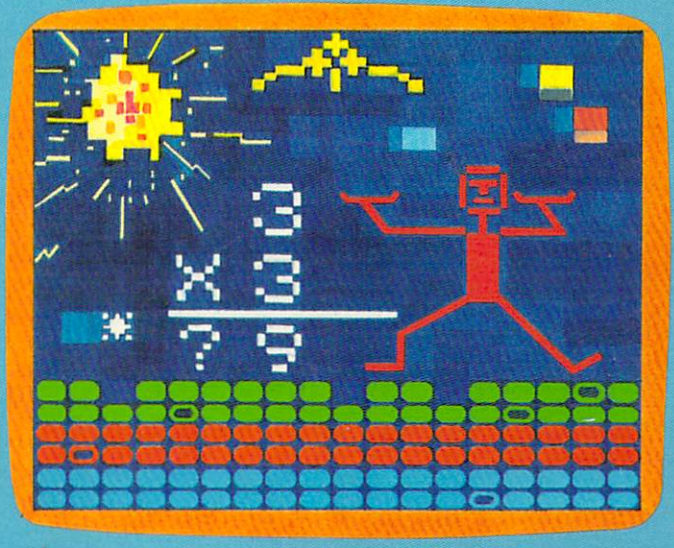
COMPUTE!'s FIRST BOOK OF

VIC GAMES

Twenty-four games for the VIC-20,* ready to type in and enjoy. Includes never-before-published games and chapters on how to develop your own games.



A **COMPUTE! Books** Publication

$12.95

COMPUTE!'s FIRST BOOK OF

VIC GAMES

COMPUTE!™ Publications, Inc.
A Subsidiary Of American Broadcasting Companies, Inc.
Greensboro, North Carolina

The following article was originally published in *COMPUTE!* Magazine, copyright 1981, Small System Services, Inc.: "Maze Generator" (December). The following articles were originally published in *COMPUTE!* Magazine, copyright 1982, Small System Services, Inc.: "Outpost" (June), "Programming Your First Game" (October), "Superchase" (October), "MathMan" (October), "Hidden Maze" (December). The following articles were originally published in *COMPUTE!* Magazine, copyright 1983, Small System Services, Inc.: "Thunderbird" (January), "Juggler" (January), "Copy Cat" (February), "Writing an Arcade Game" (February), "Closeout" (March), "Air Defense" (April), "Deflector" (May), "Jumping Jack" (May), "Hawkmen of Dindrin" (June). The following articles were originally published in *COMPUTE!* Magazine, copyright 1983, COMPUTE! Publications, Inc.: "Time Bomb" (July), "Writing a Simulation Game" (July). The following articles were originally published in *COMPUTE!'s Gazette*, copyright 1983, COMPUTE! Publications, Inc.: "Word Hunt" (July), "Sky Diver" (July).

Printed in the United States of America

ISBN 0-942386-13-2

10 9 8 7 6 5 4 3 2 1

COMPUTE! Publications, Inc., Post Office Box 5406, Greensboro, NC 27403, (919) 275-9809, is a subsidiary of American Broadcasting Companies, Inc., and is not associated with any manufacturer of personal computers. VIC-20 is a trademark of Commodore Electronics Limited.

Table of Contents

Special Requirements: J = joystick M = memory expansion

Special Requirements: J = joystick M = memory expansion

Foreword

When the first huge computers were built, games were not what the owners had in mind. Millions of dollars were invested in every machine. Computer time was valuable, and not to be wasted.

As computers shrank in size and increased in power, however, it was inevitable that weary programmers would begin exploring and programming, devising the forerunners of *Pac-Man* and *Donkey Kong*. Today, a vast number of the world's computers are built for one purpose only — to play games with whoever puts in a quarter.

Your VIC-20 is not a dedicated game machine — it is much more versatile than that. But the VIC's designers knew that one of the most common uses of the machine would be play. Like the arcade machines, the VIC can give you experiences and entertainment that you could never find anywhere except in the worlds the computer can create.

This book serves a double purpose. First, it provides you with a variety of games which you merely type into the computer, save on tape or diskette, and then play again and again as often as you like. Second, because the program is printed, you can see exactly how the game's creator brought off the effects you like. It will be fairly easy for you to learn techniques that you can use in your own programs.

In fact, to make this book as useful as possible, many of the games are accompanied by explanations of how the program works. Chapters at the beginning and end of the book will also help you learn how to write your own games.

Much of the value of this book comes from its variety. Besides being fun, "MathMan," for instance, is educational; "Jumping Jack" can improve eye-hand coordination.

There are games that are simple and slow enough for small children. There are also games as fast and challenging as anything in the arcades.

No matter what level of programming skill you have reached, there will be programs from which you can learn techniques, ranging from fairly simple BASIC games to a sophisticated all-machine-language game like "Richthofen's Revenge."

Even if you are a subscriber to *COMPUTE!* Magazine, there are things here you haven't seen before. One-third of the games in this book have never been published before and some of the others have been since refined and improved.

Some of the games here were originally programmed on other computers, and were "translated" for the VIC. Computer translation often requires as much creativity as the original program, since the refinements and features of computers can be very different.

Part 1

Playing Games With Your VIC

VIC Features: Color, Graphics, Sound, etc.

Dan Carmichael

Your VIC-20 has just completed another hard day's work. It has revamped your household budget, calculated the savings gained from your new attic insulation and checked the kids' homework. It's ready for a break and you are, too; so let the games begin!

Your VIC is especially well suited to games because of a number of exciting features:

The VIC can display up to 128 different screen and border color combinations. Characters can be printed in eight different colors. You can turn normal letters and numbers into spaceships or goblins, people or animals, just by changing the character set.

There are three separately controlled tone-generating speakers which, together, can reach a total of five octaves. A "white noise" generator can create special sound effects. A volume control system lets you vary the volume of the sounds produced by four "speakers."

A realtime clock is built into the VIC, and you can use it for timing games. And the joystick port will accept the standard Atari or VIC-20 joystick or paddles.

Also, there are some advanced features to make games on the VIC even more colorful and exciting:

Eight additional auxiliary character colors.

Multicolor graphics modes.

High-resolution graphics which lets you independently control each dot of color on the TV screen.

Complete machine language access for super-fast programs and routines in the VIC's native language.

All of this combines to make the VIC as entertaining as an arcade.

128 Colors

Here are some short programs that will demonstrate the features of the VIC-20. Before typing in each program, type NEW and press RETURN. After entering the program, type RUN and press RETURN.

The VIC is capable of producing 128 screen and border color combinations. This short program will display all 128 combinations:

```
10 FORA=0TO255:POKE36879,A:FORT=1TO100:N
   EXTT:NEXTA:POKE36879,27:END
```

As a demonstration of the eight possible character colors, type in these simple PRINT statements:

```
10 PRINT"{CLR}{2 DOWN}{6 RIGHT}{BLK}T
   {RED}H{CYN}E {PUR}V{GRN}I{BLK}C{BLU}-
   {YEL}2{BLK}0"
20 PRINT"{DOWN}{9 RIGHT}{RVS}{BLK}HAS
   {OFF}"
30 PRINT"{DOWN}{4 RIGHT}{BLK}C{YEL}O
   {BLU}L{PUR}O{CYN}R{RED}S {GRN}G{BLK}A
   {RED}L{BLU}O{PUR}R{YEL}E{BLK}!"
```

Programmable Characters

Programmable characters are one of the big pluses of the VIC-20. Instead of being restricted to the characters on the VIC's keyboard, you may, if you wish, program your own special characters into the VIC. "Balloons," one of the games found in this book, uses programmable characters to draw the balloons. The following short program also shows how this feature works.

```
10 POKE51,0:POKE52,28:POKE55,0:POKE56,28
   :POKE36869,255:PRINT"{CLR}{2 DOWN}"
20 FORA=7168TO7223:READB:POKEA,B:NEXT
30 PRINT"{CLR}{6 DOWN}{7 RIGHT}AB@@@CD"
40 GETA$:IFA$=""THEN40
100 DATA0,0,0,0,0,0,0,0,24,28,127,192,89
    ,67,192,127,0,0,240,8,238,193,1,254
110 DATA7,8,18,16,20,19,8,7,224,16,72,8,
```

```
    40,200,16,224,8,8,20,20,54,119,20,20
    ,20,20,20
120 DATA54,54,119,28,20
```

Sound

Sound isn't just decoration. It's also the game programmer's best tool to let you know that something has happened. A fanfare to let you know you've just won an extra turn or a click every time two objects collide — these sounds help you know what's going on in the game without having to take your eyes off the main activity.

Sounds also create moods and help build the excitement. The throbbing sound that gets faster and faster in the arcade game *Asteroids* is a good example of this, and musical themes in other games set a pace or help give you the feeling of rhythm and speed in your playing. Just try turning down the volume on a game that uses sound, and see how much harder it is to play — and how much less fun.

This short BASIC program written by John Heilborn (from *COMPUTE!'s Second Book of VIC*) will let you hear the sounds your VIC can make.

```
10 POKE 36875, 240
20 FOR K = 0 TO 10
30 FOR I = 0 TO 1
40 FOR R = 0 TO 15 STEP 5
50 POKE 36878, R
60 POKE 36878, 0: NEXT
70 FOR M = 0 TO 30: NEXT
90 NEXT
100 FOR D = 0 TO 300: NEXT
110 FOR W = 0 TO 1000: NEXT
120 GOTO 20
```

Keeping Time

The VIC-20 also has a realtime clock, which can be used to do anything from simply telling the time to counting down a timed round in a game. The following short program will turn your VIC-20 into a timepiece. When asked to enter the current time, enter it in this format:

HHMMSS

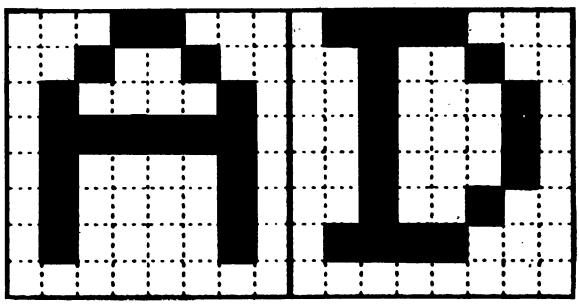
HH=hours, MM=minutes, and SS=seconds. Then press RETURN. For example, to enter 12 noon (12:00:00) enter 120000; to enter 1:45:50, enter 014550.

```
10 PRINT"{CLR}{2 DOWN}ENTER THE TIME IN"
   :PRINT"{DOWN}FORMAT:{2 SPACES}HHMMSS"
20 INPUT"{DOWN}";TI$
30 PRINT"{CLR}{6 DOWN}{4 RIGHT}{GRN}
   {RVS} THE TIME IS: {BLU}{DOWN}"
40 PRINT"{8 RIGHT}"TI$"{UP}":GOTO40
```

Graphics

High-resolution graphics allows you to program games that have arcade-style pictures with smooth movement of characters across the screen.

If you'll type the letter *A*, then take a close look at your TV screen, you'll notice that the character is constructed of many tiny dots. These dots are called *pixels*. Each standard VIC character is composed of eight rows and eight columns of pixels, for a total of 64 in each letter "block." Some pixels are turned on, to show the letter; others are off, so that the background color is displayed. In effect, then, when you type the letter *A* you are moving a block of 64 pixels to the screen, some turned off and some turned on, in the pattern shown in this diagram:



The letters A and D as drawn by the VIC.

When you use high-resolution graphics, you aren't limited to controlling the pixels through character patterns. Instead, you control each pixel on the screen individually. This lets you create anything from beautiful, detailed pictures to animated figures that move smoothly around the screen.

Machine Language

Another very nice feature of the VIC-20 is the ability to program in VIC's native language — *machine language.* Most commercial games you buy for the VIC are written in machine language, because it's much faster than BASIC, and because it uses less memory than the same program would use even if it *could* be written in BASIC.

As a demonstration of just how much faster machine language can be, enter these two short programs. First, type in the following BASIC program, and then RUN. It will fill the screen with periods (.), then tell you how long it actually took to do it.

```
1 B=7680:C=38400:IF4*(PEEK(36866)AND128)
  +64*(PEEK(36869)AND112)=4096THENB=4096
  :C=37888
10 PRINT"{CLR}":D=505:TI$="000000":FORA=
   0TOD:POKEB+A,81:POKEC+A,6:NEXT:PRINTT
   I/60:END
```

As you can see, this BASIC program took about four seconds to fill the screen.

The next BASIC program will POKE a short machine language program into memory, then RUN it. This program will also fill the screen with periods, then print the time, in seconds, the task took. (Be careful when entering the DATA statements: an error can cause the program to do very odd things.)

```
5 FORA=828TO859:READB:POKEA,B:NEXT
10 POKE251,0:POKE252,30:POKE253,0:POKE25
   4,150
20 IF4*(PEEK(36866)AND128)+64*(PEEK(3686
   9)AND112)=4096THENPOKE252,16:POKE254,
   148
30 PRINT"{CLR}{DOWN}PRESS ANY KEY,":PRIN
   T"{DOWN}BUT DON'T BLINK!!!"
```

```
32 GETA$:IFA$=""THEN32
35 TI$="000000":SYS828:PRINTTI/60
100 DATA160,0,169,81,145,251,200,208,251
    ,230,252,145,251,200,208,251,169,6,1
    45,253,200
110 DATA208,251,230,254,145,253,200,208,
    251,96,234
```

You will notice that this machine language program runs quite a bit faster than the BASIC program.

The games are fun for their own sake. But one of the best things about typing in programs yourself is that you can see exactly how another programmer created the effects you want to use in your *own* games. You may soon find that the best computer game of all is programming games for other people to play!

Writing Your First Game

Richard Mansfield

Richard Mansfield, Senior Editor of COMPUTE! Publications, explains the details of a simple game. A beginning programmer can learn a great deal by studying this short program.

If you are tempted to write your own games, go ahead. It's a good way to learn to program. Games are basically the same as any other kind of programming.

Computer games fall into two broad categories: imitations of old standards (checkers, Othello) and games which could not be played without a computer. This second category is more difficult to program for several reasons. For one thing, you've got to think up a whole new, and entertaining, concept and then adjust the action until it is just hard enough to be challenging but not so difficult that people want to give up.

This category (basically "arcade" games) is especially hard to program precisely because a good computer-only game exploits all of the computer's special attributes: speed, color, sound. To do this well, to make things look and respond just the way you imagine them, requires a good bit of programming experience. Usually, too, several things are happening *at once* in an arcade game. This often means that such a program must be written in machine language, which is far faster than BASIC.

High Card Slice

Old standards, on the other hand, can often be the best way to get started programming games. You already know the game concept, and cards or dice or game boards are fairly easily constructed and manipulated on your computer screen. To illustrate, let's take a look at a simple simulation of one of the oldest card games, High Card. The rules are simple: you place a bet, and then you draw a card from the deck. The

computer, your opponent, draws a card too, and the highest card wins the money.

One simplification here is that there is no attempt to represent the cards on the screen. The entire game relies simply on words ("Ace of Spades," for example) when cards are drawn.

Like most computer programs, the program can be visualized as having four distinct zones: initialization, main loop, subroutines, data tables.. We can go through the steps in programming this game by looking at each zone separately.

Initialization

From lines 10 through 80 we are "teaching" the computer some basics about this game. Initialization is the activity which must take place before any of the action can begin. Computers are so fast that they will zip up through these lines and start things off in the main loop at line 100 in a flash. However, as programmers, we are aware that several preliminary events took place inside before anything else.

In line 20, the computer discovers that there is a variable called "dollars" which is to equal 500. It sets aside a section (like a small box) in its memory which it labels "dollars." When the game is running, it will add or subtract from this "box" (lines 230-240) to keep a running total of how much money you have left to bet. From time to time (line 110), it will check the box and report to the player how much he has. The box labelled "dollars" is called a *variable* because during the game the amount in it will vary.

Lines 30 through 60 are simple enough — they ask the player to give his or her name. The computer "memorizes" it in another "box" called "name$" and can now speak more personally to the player in lines 140 and 230. Also, the computer prints the rules of the game in line 60.

Line 70 "reads" four names (the face cards) from the data tables in lines 510 on. It also makes a "mental note" that it already READ four items. So, when it's asked to READ again (line 80), it will start with the next unread item of data which will be "clubs." By now, the computer has "memorized" a variety of important facts: the player's name, the amount of his or her betting purse, the names of the face cards, and the suits of a standard deck. In less than a second, the computer

has grasped and filed away the necessary facts to go on to the main loop where all the action takes place.

The Main Loop

After checking that the player has money to bet, the computer asks for the bet, checks again that the bet is possible, and then runs through one cycle of the game starting in line 160. At this point, a programmer might find it worthwhile to visualize the steps involved in the game.

Draw a card for the player.
Draw for the computer.
Decide who won.
Adjust the player's purse.

Since both draws are essentially identical actions (the only difference will be that we say "Bob draws a ..." instead of "The computer draws"), we don't need to program the draw twice. This is where subroutines come in handy.

The Subroutine

Twice in the main loop, we GOSUB 300. First the player, then the computer, draws. Line 310 randomly picks two numbers, the card and the suit. If line 320 finds that this selection matches the one drawn just before by the player, it goes back for another draw. Line 330 makes the *name* of the card be the number if it wasn't a number higher than 10 (a face card).

Then line 340 announces the draw using three variables. The first variable (player$) is set up in either line 160 or 190 as appropriate. Then the card$ and suit$ variables are selected from the lists that were "memorized" back in the initialization phase (lines 70-80). The subroutine then RETURNs to the main loop.

Lines 210-240 decide and announce the winner of this round. First, if the variable "card" (the computer's card) is greater than (>) "yourcard," the computer is declared the winner in line 240, the purse is adjusted, and the main loop is restarted (GOTO 100). If the cards are equal, nothing happens to the purse and the next round begins. Notice that we don't need to say "IF YOURCARD > CARD" at the start of line 230 to test if the player has won. It's the only possible thing if the computer has gotten this far.

Once you've solved a particular problem, you'll find you can use the solution in many future games. This subroutine

which draws cards, for instance, would work just as well for Poker, or Blackjack, or dozens of other games. Subroutines are handy not only because they can be used repeatedly within a program, but because they can be saved and used repeatedly in future programs. So think up a simple, traditional game and teach it to your computer. There is probably no more pleasurable way to learn programming than to write a game.

Program 1-1. High Card Slice

```
10 REM*NECESSARY INITIAL INFORMATION*
20 DOLLARS=500
30 PRINT " WITH WHOM DO I HAVE THE PLEAS
   URE"
40 PRINT " OF PLAYING HIGH CARD SLICE?"
50 INPUT NAME$
60 PRINT " HIGH CARD WINS IN THIS GAME!"
70 DIM SUIT$(4),CARD$(14):FOR I=11 TO 14
   :READ CARD$(I):NEXT I
80 FOR I=1 TO 4: READ SUIT$(I):NEXT I
90 REM
100 REM*MAIN PROGRAM LOOP*
110 PRINT:PRINT" YOU HAVE $" DOLLARS
120 IF DOLLARS<=0 THEN PRINT" THE GAME I
    S OVER. YOU ARE OUT OF CASH.":END
130 PRINT"WHAT IS YOUR BET";:INPUT BET
140 IF DOLLARS<BET THEN PRINT" YOU ONLY
    HAVE $"DOLLARS" TO BET,"NAME$:GOTO 1
    30
150 YOURCARD=0:YURSUIT=0
160 PLAYER$=NAME$
170 GOSUB300
180 YOURCARD=CARD:YURSUIT=SUIT
190 PLAYER$=" THE COMPUTER"
200 GOSUB300
210 IF CARD>YOURCARD THEN GOTO 240
220 IF CARD=YOURCARD THEN PRINT " A TIE!
    ":GOTO 100
230 PRINT NAME$ " WINS": DOLLARS = DOLLA
    RS + BET:GOTO 100
240 PRINT " THE COMPUTER WINS": DOLLARS=
    DOLLARS-BET:GOTO 100
```

```
290 REM
300 REM*SUBROUTINE TO DRAW THE CARDS*
310 CARD=INT(RND(5)*13)+2:SUIT=INT(RND(5
    )*4)+1
320 IF CARD=YOURCARD AND SUIT=YURSUIT TH
    EN 300:REM NO IDENTICAL DRAWS
330 IF CARD<11 THEN CARD$(CARD)=STR$(CAR
    D)
340 PRINT PLAYER$ " DRAWS THE " CARD$(CA
    RD) " OF " SUIT$(SUIT)
350 RETURN
490 REM
500 REM* DATA TABLE*
510 DATA JACK,QUEEN,KING,ACE
520 DATA CLUBS,DIAMONDS,HEARTS,SPACES
```

Writing A Simulation Game

Richard Mansfield

A simulation is an imitation of life. It can be the most difficult type of game to create. Thought, rather than fast action, is important. Try the short simulation offered here, then see if you can write one of your own.

There are three basic types of computer games: arcade, adventure, and simulation games. Let's briefly look at the characteristics of arcade and adventure games and then write a simulation.

Realtime Action

Arcade games feature what's called *realtime* action. Unlike chess or bridge, things happen fast. You can't sit back and plan your next move; you must react immediately to the space invaders. In other words, events take place at the same speed as they would in reality: realtime.

Arcade games also have a strong appeal to the eye and ear. There is much animation, color, and sound. In fact, your ability to respond quickly and effectively depends in part on all the clues you get from the graphics and sound effects. Strategy, while often an aspect of arcade play, is clearly secondary. These games are a new kind of athletics: the fun of man versus machine. Like auto racing, arcade games are essentially isometric exercises — you don't run around; you just stay in one place flexing and unflexing your muscles, tensing and relaxing.

Story and Strategy

Strategy, however, is more important in "adventure" games. The emphasis is on planning ahead and solving riddles. It can be like living inside an adventure novel. There is drama, characterization, and plot. You might start out, for example, in

a forest with a shovel and a trusty, if enigmatic, companion parrot. As you try to figure out what to do next, the parrot keeps saying "piny dells, piny dells." After wandering aimlessly through the trees, it suddenly comes to you that the bird is saying "pine needles" and you dig through them and find a treasure map.

Your "character" will travel, meet friends and enemies, and have the opportunity to pick up or ignore potentially useful items such as food, magic wands, and medicine. It's customary that you cannot haul tons of provisions. You'd have to decide whether or not to leave the shovel in the forest. Yet you might be sorry that you'd dropped it if you're involved in a cave-in later in the game.

In any case, adventure games are fundamentally verbal. The computer displays the words:

YOU ARE IN A BOAT ON A LAKE. NIGHT IS FALLING.

to which you can respond in any number of ways. You might type:

DIVE OFF BOAT.

and the computer would reply that you now see an underwater cave or whatever. You move through the scenes the way a character moves through a novel. There is generally no penalty if you take time to plan your next move. It's not *realtime.*

Imitations of Life

The third category, simulation, is the least common kind of computer game. This is because to really imitate something, to *simulate* it effectively, you need lots of computer memory to hold lots of variables. However, memory has recently become far less expensive so we can expect to see increasingly effective simulation games. *Star Trek* and *Hammurabi,* both simulations, have long been popular home computer games. Although they are similar to adventure games, simulations are random. That is, there is no secret to discover, no puzzle to solve, no plot. Like real life, things happen with unpredictable, complex results.

Here's a program which simulates investing. The key to simulating is to arrange realistic *interactions* between variables. Look at line 600. If there is "international unrest," the price of gold (PGLD) goes up and the price of Bundtfund stock (PB) goes down. This relationship between gold, stock, and an in-

ternational crisis is true to life. Alternatively, stock goes up and gold goes down on line 700 during a "market rally."

The game allows you to make investment decisions, and then a "month" passes during which the value of your investments will go up or down. In line 510, three variables are given random values. Stock can gain or lose up to 10 points (variable X), and gold can change by $20 an ounce (Y). Variable Z will be used to simulate flipping a coin. Also notice lines 520 and 525. In 520, we determine whether or not there will be unrest. The variable CH is just a counter. Each "month," CH is raised by one. Two conditions are required for unrest to happen: in a given month, CH must be greater than 4 and it must be less than whatever X turns out to be. If both these conditions are met, CH is reset to zero and we've got international unrest. This has the effect of creating unrest roughly every four to six months. Likewise, another rhythm is set up in line 525 to cause market rallies. In both cases, however, you cannot be certain exactly when to invest in gold or in stocks.

The decision to raise or lower stock prices is made in line 530 and based on the coin toss variable, Z. Again, stocks move in opposition to gold. Prices will rise about 50 percent of the time, but you can never know what will happen in a given month.

Suggested Complications

This is the core, a rough sketch, of an investment simulation game. There is much you can do to make it a more effective simulation and thereby a more enjoyable game. The more variables in a simulation, the better. For example, add leverage and additional "incidents" which affect prices, improve the randomizing, and include other types of investments. You could even use a separate counter which, every five years, causes the X and Y variables to swing more widely to reflect recession/recovery cycles.

As you can see, a simulation should be lifelike. It has interdependent cycles and a degree of unpredictability. Its realism derives from including a sufficient number of variables. And those variables must interact in plausible ways and with just the right amount of randomness. A simulation is a little world you create. You can define cause and effect and then fine-tune the whole thing until it seems well-balanced. Ad-

venture and arcade games are certainly enjoyable, but this investment simulation can be built up to the point where it's just as much fun as any other kind of game.

Mixing Styles

Of course, these three categories — arcade, adventure, and simulation — are somewhat arbitrary. Some of the best games contain elements of each. There are adventure games with graphics — you see the forest, the shovel, the pine needles. After you say DIVE, your character jumps into a lake and the screen transforms into an underwater scene. Likewise, arcade games can include the different "settings" so characteristic of adventure games. Popular arcade games such as *Tron* and *Donkey Kong* change the playfield as you earn more points.

There are several ways to add to the appeal of our investment simulation, beyond just making it a more complex, more accurate simulation. You could add the visuals and sound of arcade games. Try creating a ticker tape across the top of the screen to show price changes and news events. Maybe add a bell sound to indicate the end of further transactions. And from adventure games you could borrow two elements: riddles and the necessity of planning ahead. One easy way to incorporate these two elements would be to make paying taxes a part of the game. After all, the closer it is to real life, the better the simulation.

Program 1-2: Investment Simulation

```
5 PRINT"{CLR}"
10 CASH=100000:PGLD=400
15 POKE 36869,242:REM SHIFT TO LOWER CAS
   E
20 PB=80
31 PRINT: PRINT"BUNDTFUND IS $"PB" PER S
   HARE.YOU HAVE "B"{4 SPACES}SHARES. --
    $"PB*B
33 PRINT" GOLD IS{4 SPACES}$"PGLD" PER O
   UNCE.{2 SPACES}YOU HAVE "GLD" OUNCES.
    -- $"GLD*PGLD
34 T=PB*B+GLD*PGLD
35 PRINT:PRINT" TOTAL INVESTMENTS -- $"T
36 PRINT:PRINT" YOU HAVE $"CASH" TO SPEN
   D."
```

```
40 PRINT:PRINT"GRAND TOTAL":PRINT"(INVES
   TMENTS + CASH){4 SPACES}$"T+CASH
45 IFCK=1THEN500
50 PRINT: PRINT"1.BUY{2 SPACES}2.SELL
   {2 SPACES}3.DONE"
60 INPUTA:IFA=3THENCK=1:GOTO31
100 PRINT"WHICH?{3 SPACES}1.GOLD
    {2 SPACES}OR{4 SPACES}2.STOCK"
110 INPUTF
120 PRINT"HOW MANY (SHARES OR{3 SPACES}O
    UNCES)?"
130 INPUTN
140 IFF=1THEN160
150 PRICE=PB*N:IFA=1THENCASH=CASH-PRICE:
    B=B+N:GOTO400
155 CASH=CASH+PRICE:B=B-N:GOTO400
160 PRICE=PGLD*N:IFA=1THENCASH=CASH-PRIC
    E:GLD=GLD+N:GOTO400
170 CASH=CASH+PRICE:GLD=GLD-N
400 GOTO50
500 PRINT"PRESS ANY KEY TO CONT" ;
503 GET C$:IF C$=""THEN 503
505 CK=0:PRINT:PRINT"{CLR}ONE MONTH LATE
    R ...":FORT=1TO700:NEXTT:PRINT
510 X=INT((RND(1)*100)/10):Y=INT((RND(1)
    *200)/10):Z=RND(1)
520 CH=CH+1:IFCH>4ANDCH<XTHENCH=0:GOTO60
    0
525 IFCH=2GOTO700
530 IF Z>.5THENPB=PB+X:PGLD=PGLD-Y:GOTO3
    1
540 PB=PB-X:PGLD=PGLD+Y:GOTO31
600 PRINT"INTERNATIONAL UNREST...":PGLD=
    PGLD+2*Y:PB=PB-2*X:GOTO31
700 PRINT"MARKET RALLY ...{2 SPACES}":PG
    LD=PGLD-2*Y:PB=PB+3*X:GOTO31
```

Writing An Arcade Game

Richard Mansfield

Using the memory-mapped video *could help you create faster moving games. The sample program here will assist you in designing your own fast-moving game.*

When you bring home your computer, usually the first thing everyone expects you to do is to write an arcade game. Who's "everyone"? It could be your children, your friends, even you — anybody who is tired of spending lots of money and wants you to program a game to play at home for free.

The best defense is to politely point out that:

1. Arcade games are among the hardest types of software to write.
2. Professionals, working in teams, can take a year to write one.

However, it is well worth trying to write action games. You might not be able to duplicate the speed or complexity of professional games, but you can create very entertaining games of your own. After you've spent a few weeks getting familiar with BASIC and have typed in a few games, you are ready to take up the challenge. This is one of the best ways to learn some important programming techniques and to explore the graphics and sound capabilities of your computer.

Ten Million IF/THENs

Your main problem is going to be speed. BASIC, though fast enough for most jobs, is pretty slow when it has to keep track of ten aliens, two mother ships, torpedoes, stars, and the player's position. All these things are in motion at once. You need to have a way to control players, to detect collisions, to score points, etc. We at *COMPUTE!* received a letter from reader John Anderson which touches on these problems:

In order to make a fast, effective "arcade-style" game, I would like to know how to let my computer know where a large number of things are on the screen (like walls in a maze) without 10,000,000 IF/THEN statements. I would also like to know how to keep things, like the little figures racing around during a game, from plowing through walls and wiping them out or coming back onto the other side of the screen.

As Anderson points out, the first solution that comes to mind is to use an IF/THEN test for every possible event in the game. IF the ball hits the target, THEN raise the score. IF the ball misses the target, THEN let it move one more space. And on and on. This quickly slows the action down to a crawl.

POKE Ping-Pong

One of the simpler arcade games is a simulation of Ping-Pong. You need to keep track of only three things: two paddles and one ball. Let's start off by solving the hardest problem. How can we bounce a ball around the screen both quickly and accurately?

The key to the problem is the fact that many computers have an area set aside in RAM which is an *image* of what you see on screen. This is called *memory-mapped video* and most computers have it. It means that if you POKE into that area of RAM, a character will appear on the screen. The next RAM byte address is the next space on screen, and so on. You can use this built-in "map" to tell what is where by using the fast "PEEK" command, and you can move things quickly with POKEs.

The example program will work on all VICs.

SCR =The address where screen RAM memory starts.
LN =The length of one screen line.
WALL =A solid square that appears when this number is POKEd anywhere into SCR.
BLANK =A blank space character that returns the screen to normal if POKEd into SCR on top of a WALL or FIGURE.
FIGURE =A character that, when POKEd into SCR, looks like a ball.

The memory cells holding the screen image are located in different places. The VIC determines where it starts by using

the formula in line 100. First, draw a border around your screen like a picture frame. Perhaps print reversed spaces all around. (See lines 250-310.) This border is very useful. It will let you know when your ball has hit the edge.

LOC is a variable in the program that's always changing whenever the ball changes. It keeps track of the current location of the ball. What you do is keep another variable (VECTR, in this example) which holds the direction and distance of the ball's current motion. When VECTR is added to LOC, we know where to move the ball next.

There are four possible directions to go in the simplest kind of animated games. Traveling up, VECTR =-LN since you subtract the number of spaces in one screen line to move the ball to the line above. Going down is +LN, right is +1, left is -1.

Notice line 180. That is how the computer tells if the ball has reached a border. The next position the figure is supposed to be POKEd into is checked to see if the WALL variable is sitting there. If not, the figure is moved (lines 200-220). If there is a wall, line 190 reverses the figure's direction.

If you type in the example program, you'll be on your way to making a Ping-Pong game that will be as fast as you could want. What's left is to play around with VECTR to get different angles of bounce off walls so the ball can go anywhere. Then add two movable pieces of wall (paddles) and scorekeeping.

Program 1-3: Ping-Pong

```
100 SCR=4*(PEEK(36866)AND128)+64*(PEEK(3
    6869)AND120):COL=37888+4*(PEEK(36866
    )AND128)
110 WALL=160:REM WALL CHARACTER, SOLID S
    QUARE.TRY OTHER CHARACTERS.
120 LN=22
130 GOSUB 260:REM DRAW BORDER
140 LOC=SCR+LN*10+LN/2:CLOC=COL+LN*10+LN
    /2:REM SCREEN AND COLOR LOCATION AT
    FIRST
150 VECTR=LN:REM ALSO TRY -1,+1,LN-1,LN+
    1,ETC.
160 BLANK=32
```

```
170 FIGURE=81:REM "BALL"CHARACTER.
180 IF PEEK(LOC+VECTR)<>WALL THEN 200
190 VECTR=-VECTR:REM REVERSE DIRECTION
200 POKE LOC,BLANK:REM ERASE OLD BALL
210 LOC=LOC+VECTR:CLOC=CLOC+VECTR:REM CA
    LCULATE NEW POSITION
220 POKE LOC,FIGURE:POKECLOC,6:REM PLACE
     BALL
230 GOTO 180
240 END
250 REM BORDER SUBROUTINE
260 PRINT"{CLR}";:REM CLEAR SCREEN.
270 FOR I=0 TO LN-1:POKE SCR+I,WALL:POKE
    COL+I,2:NEXTI:REM TOP
280 FOR I=0 TO LN-1:POKE SCR+LN*22+I,WAL
    L:POKECOL+LN*22+I,2:NEXTI:REM BOTTOM
290 FOR I=0 TO 22:POKESCR+I*LN,WALL:POKE
    COL+I*LN,2:NEXT I:REM LEFT
300 FOR I=0 TO 22:POKE SCR+LN-1+I*LN,WAL
    L:POKECOL+LN-1+I*LN,2:NEXTI:REM RIGH
    T
310 RETURN
```

Part 2

Maze Games

Time Bomb

Doug Smoak

"Time Bomb" for the unexpanded VIC creates a maze which is three screens long. The game uses machine language to create the maze and allows scrolling up and down.

You play "Time Bomb" against the clock. You start at the bottom of a maze which is about three times the size of the VIC's screen. At the top of the maze is a time bomb ticking away. The closer it gets to blowing up, the higher pitched the ticking becomes. If you reach the bomb, you must steer the pointer into it to defuse it. If you are successful, you have a go at the same maze, but with the bomb in a different place and with a shorter fuse. This continues until you run out of time. If you fail to defuse it, you get a new maze and a new bomb with a longer fuse.

Time Bomb is quite challenging to a player's memory of spatial relationships. People who are at first intimidated by seeing only a portion of the maze quickly become accustomed to thinking ahead and remembering the dead ends and clear paths through the maze. An ability to recall the good and bad moves is crucial to getting into the later rounds.

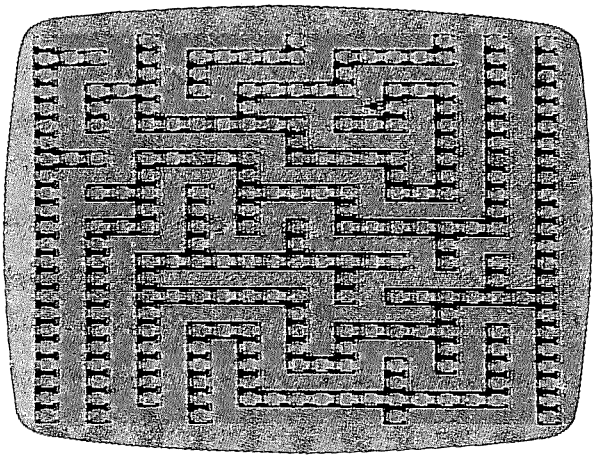
My aim was to create something more challenging than a single screen maze. I then hit on the idea that makes this game so entertaining: to make the maze larger than the screen and bring it on and off the display by scrolling it out of a much larger block of memory.

How the Idea Came

It sounded great, but how would I do it? The secret lies in a short machine language routine that is called to update the display whenever the player goes up or down in the maze. It does this so quickly that I used the BASIC joystick routine from *COMPUTE!'s First Book of VIC* just to keep things at a reasonable pace.

There are actually three separate machine language routines that are represented by the DATA statements. One fills

the maze area with the proper character, another fills the screen's color RAM with the proper color, and the third one scrolls the maze. I could have used BASIC POKEs to do all these things, but the time consumed would be too great. It would be impossible to use POKEs to scroll the maze with enough speed to be any fun at all.



Maze fans will find "Time Bomb" quite a challenge.

When typing in the program, be sure to SAVE it before you RUN it, since a typo in the DATA statements could cause you to lose the whole program. Be very careful as you enter the DATA statements. If you have a bug in the program, it is most likely in the DATA statements, so look there first.

When you do RUN it, there will be a slight pause while the machine language parts are POKEd into the cassette buffer. Then the screen should clear, and the words "Making Maze" should appear. Because of the size of the maze, the VIC needs almost a minute to draw it, so be patient. When the maze is complete, a musical announcement alerts you to begin playing. Don't give up if you are eliminated on the first round; it takes a while to get used to looking ahead in the maze and planning your route.

You can give yourself more time to reach the bomb by making these changes in the program:

```
6 K=K+1:ON-(K/2 < > INT(K/2))GOTO8:IFK>840THEN37
7 FORT=1TO2:POKEV,T*4:POKES+1,128+K/7:NEXT:POKES+1,0
```

Program 2-1. Time Bomb

```
2  POKE56,24:POKE55,103:GOSUB29
3  D=37154:P1=D-3:P2=D-2:DF=3072Ø:V=36878
   :S=V-4:M1=3Ø:X=5Ø:GOTO19
4  FORT=24ØTO2Ø8STEP-4:POKES,T:FORTT=ØTO3
   Ø:POKEV,TT/2:NEXT:NEXTT:POKES,Ø:ME=793
   2
5  POKEOM,32:POKEOM+DF,1Ø:POKEME,M1:POKEM
   E+DF,6:IFFTHEN4Ø
6  K=K+1:ON-(K/2<>INT(K/2))GOTO8:IFK>6ØØT
   HEN37
7  FORT=1TO2:POKEV,T*4:POKES+1,128+K/5:NE
   XT:POKES+1,Ø
8  POKED,127:P=PEEK(P2)AND128:JØ=-(P=Ø)
9  POKED,255:P=PEEK(P1):J1=-((PAND8)=Ø):J
   2=-((PAND16)=Ø):J3=-((PAND4)=Ø)
1Ø IFJØTHENC=1:M1=62:GOTO14
11 IFJ1THENC=22:M1=22:GOTO14
12 IFJ2THENC=-1:M1=6Ø:GOTO14
13 IFJ3THENC=-22:M1=3Ø
14 OM=ME:ME=ME+C:C=Ø
15 IFPEEK(ME)<>32ANDPEEK(ME)<>42THENME=O
   M
16 IFPEEK(ME)=42THENF=1:GOTO5
17 ON-(ME>7921)GOTO18:SYS887:ME=ME+22:GO
   TO5
18 ON-(ME<7944)GOTO5:SYS9Ø5:ME=ME-22:GOT
   O5
19 DIMA(3):A(Ø)= 2:A(1)=-44:A(2)=-2:A(3)
   =44:WL=2Ø9:HL=32:SC=6228:A9=6943
2Ø SYS861:PRINT"{CLR} {DOWN}MAKING MAZE"
21 FORT=SC+21TO7679STEP22:POKET,32:NEXT:
   FORT=SCTOSC+21:POKET,32:NEXT
22 J=INT(RND(1)*4):X3=J
23 B=A9+A(J)
24 IFPEEK(B)=WLTHENPOKEB,J:POKEA9+A(J)/2
   ,HL:A9=B:GOTO22
25 J=(J+1)*-(J<3):IFJ<>X3THEN23
```

```
26 J=PEEK(A9):POKEA9,HL:IFJ<4THENA9=A9-A
   (J):GOTO22
27 TB=SC+INT(RND(Ø)*2Ø)+22Ø:ON-(PEEK(TB)
   <>32)GOTO27:POKETB,42
28 SYS83Ø:POKE828,2Ø4:POKE829,28:SYS923:
   GOTO4
29 FORI=83ØTO974:READA:POKEI,A:NEXT:RETU
   RN
3Ø DATA169,238,141,15,144,169,Ø,133,251,
   169,15Ø,133,252,16Ø,Ø,169,1Ø,145,251,
   2ØØ,2Ø8
31 DATA251,23Ø,252,165,252,2Ø1,152,2Ø8,2
   41,96,169,84,133,251,169,24,133,252,1
   6Ø,Ø,169
32 DATA2Ø9,145,251,2ØØ,2Ø8,251,23Ø,252,1
   65,252,2Ø1,3Ø,2Ø8
33 DATA241,96,173,6Ø,3,56,233,22,176,3,2
   Ø6,61,3,141,6Ø,3,56,176,19,234,173,6Ø
   ,3,24,1Ø5
34 DATA22,144,3,238,61,3,141,6Ø,3,24,144
   ,1,234,169,Ø,133,Ø,169,3Ø,133,1,173,6
   Ø,3,133
35 DATA254,173,61,3,133,255,169,Ø,133,25
   3,16Ø,Ø,177,254,164,253,145,Ø,132,253
   ,23Ø,253
36 DATA234,2Ø8,2,23Ø,1,23Ø,254,2Ø8,2,23Ø
   ,255,169,32,197,1,2Ø8,227,96
37 POKEV,15:FORT=255TO127STEP-2:POKES,T:
   POKEV-9,255:FORG=1TO1Ø:NEXT
38 POKEV-9,242:FORG=1TO1Ø:NEXT:POKEV-9,2
   4Ø:NEXT:POKEV-1,22Ø:FORG=15TOØSTEP-.Ø
   5
39 POKEV,G:POKEV+1,G*1Ø:NEXT:POKEV-1,Ø:P
   OKEV+1,238:GOSUB42:RUN
4Ø POKETB,32:POKEV-1,253:FORG=3ØTOØSTEP-
   .15:POKEV,G/2:NEXT:X=X+5Ø:IFX>449THEN
   X=45Ø
41 POKEV-1,Ø:F=Ø:K=X:R=R+1:GOSUB42:GOTO2
   7
42 PRINT"{HOME}ROUND"R"{LEFT} ":PRINT"
   {DOWN}PRESS F7 ":A$="":GETA$:ON-(PEEK
   (197)<>63)GOTO42:RETURN
```

Hidden Maze

Gary Boden
Translated for the VIC by Charles Brannon.

The maze is there — you just can't see it! This maze game adds a new twist to maze puzzles.

Mazes present a challenge different from arcade-type "shoot-out" games, but the appeal of a maze can quickly fade once it has been solved. I have enhanced its challenge by hiding the complete maze from the player and showing only a realistically limited view from any position inside it. Although the view is from above rather than ground level, the player still gets a claustrophobic feeling similar to that of actually being inside the maze and groping along the corridors.

Playing Hidden Maze

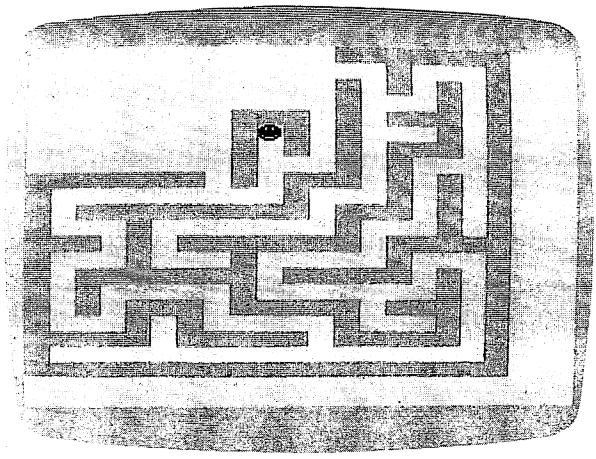
The objective is simply to find a way out of the maze in the least amount of time. Realism is added by showing at most only seven cells in any of the four possible directions of movement. This simulates holding up a lantern and peering down various avenues of escape — at a certain point the light either illuminates a wall or disappears into the gloom.

Moves are made by pushing the joystick in a particular direction. If no wall obstructs, the smiling face advances one cell and a new limited view is displayed. Time ticks on relentlessly whether the player is moving or thinking. Hitting the fire button reveals a quick glimpse of the whole maze, but the player is "paralyzed" for a few seconds as a penalty.

Program Description

The maze is constructed (lines 480-580) in the manner described in Appendix A, but by filling color memory with the value for the screen color (line 520), the maze is made invisible. As the player proceeds through the maze, the color memory for the characters around the player is POKEd with values which make walls visible (lines 170-200). The player always starts in the center and tries to reach "home" in the upper-left corner.

Four player characters are defined (lines 360-470), one each for the smiling face looking left, right, up, and down. In this way, the player is always facing the direction of the last move. The joystick is read (line 210) to determine the direction to move (line 215) and which player character to use (line 220). If the fire button is pressed, a quick view of the maze is given and the player is "frozen" for a few seconds (line 240).



The more success you have, the more maze you will see in "Hidden Maze."

Line 250 checks if a wall has been hit, and line 270 checks whether the end of the maze has been reached. Line 280 flashes the screen when the player reaches home. Scoring is based on the time taken to travel the maze, using the VIC's built-in clock (lines 160, 300-310).

Program 2-2. Hidden Maze

```
100 REM HIDDEN MAZE:
110 PRINT"{CLR}";:GOSUB 360:GOSUB480
120 PP=253
130 POKE SCR+PP,5:POKE CMEM+PP,2
140 DIM DIR(3)
150 DIR(0)=22:DIR(1)=23:DIR(2)=21:DIR(3)
    =1
160 T=TI
170 FOR I=0 TO 3
```

```
180 POKE CMEM+PP+DIR(I),5
190 POKE CMEM+PP-DIR(I),5
200 NEXT I
210 POKE37154,127:X=(NOTPEEK(37151))AND6
    0-((PEEK(37152)AND128)=0):POKE37154,
    255
211 IFX=0THEN210
215 TP=PP-22*((XAND8)>0)+22*((XAND4)>0)-
    ((XAND1)>0)+((XAND16)>0)
220 CHR=-(3*((XAND16)>0)+4*((XAND1)>0)+5
    *((XAND4)>0)+6*((XAND8)>0))
230 IFCHR<3ORCHR>6THENCHR=5
240 IF(XAND32)THENPOKECC,8:FORW=1TO2000:
    NEXT:POKECC,27:FORW=1TO2000:NEXT
250 IF PEEK(SC+TP)<>32 THEN 270
260 POKE SCR+PP,32:POKE SCR+TP,CHR:POKE
    CMEM+TP,2:PP=TP
270 IF PP<>23 THEN 170
280 FORI=1TO100:POKE CCTRL,255*RND(0):NE
    XT:POKECCTRL,27
290 PRINT"{CLR}{RVS}{PUR}YOU DID IT!":PO
    KE36869,240
300 SEC=INT((TI-T)/60)
310 PRINT"{GRN}IN";SEC;"SECONDS
320 PRINT:PRINT"{CYN}PRESS{RED}{RVS}SPAC
    E{2 OFF}{CYN} TO":PRINT"PLAY AGAIN.
    {BLU}"
340 GETA$:IFA$=""THEN340
350 RUN
360 REM LOAD CHARACTER SET
365 CHSET=7168:POKE51,240:POKE52,CH/256-
    1:POKE55,240:POKE56,CH/256-1
370 FORI=0TO7:POKECH+256+I,0:NEXT
380 READA:IFA=-1THENRETURN
390 FORJ=0TO7:READB:POKECHSET+A*8+J,B:NE
    XTJ
400 GOTO380
410 DATA3,56,124,174,174,254,186,68,56
420 DATA4,56,124,234,234,254,186,68,56
430 DATA5,56,84,214,254,254,186,68,56
440 DATA6,56,124,254,214,214,186,68,56
441 DATA7,255,255,255,255,255,255,255,255
```

```
470 DATA-1
480 POKE36869,255
485 PRINT"{CLR}{22 DOWN}{RVS}GENERATING
    MAZE{HOME}{OFF}";
490 SC=7680:CMEM=38400:CCTRL=36879
500 DIMA(3):A(0)=2:A(1)=-44:A(2)=-2:A(3)
    =44
510 A=SC+23:WL=7:HL=32
520 FORI=1TO21:PRINT"{WHT}GGGGGGGGGGGGG
    GGGGG":NEXT:POKEA,5
530 J=INT(RND(1)*4):X=J:POKESC+505,J+128
    :POKECM+505,8*RND(0)
540 B=A+A(J)
550 IFPEEK(B)=WLTHENPOKEB,J+1:POKEA+A(J)
    /2,HL:A=B:GOTO530
560 J=-(J+1)*(J<3):IFJ<>XTHEN540
570 J=PEEK(A):POKEA,HL:IFJ<5THENA=A-A(J-
    1):GOTO530
575 PRINT"{HOME}{22 DOWN}{17 SPACES}
    {HOME}";:POKESC+505,32
580 RETURN
```

Superchase

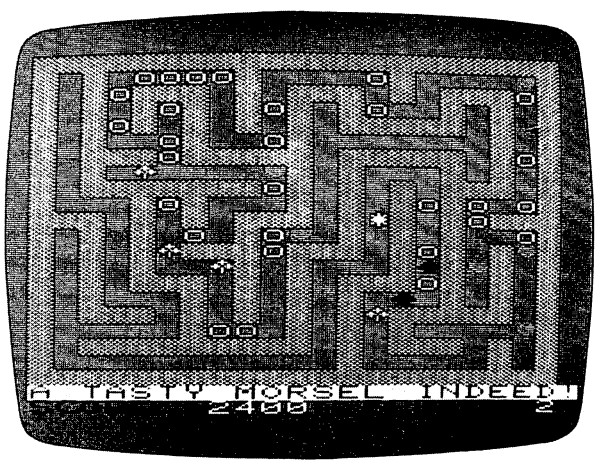
Anthony Godshall

Fast action and strategy make this maze game fun for all ages.

"Superchase" is an arcade-style game where you try to find all the treasures before the monster of dungeons gets you! Sounds easy, doesn't it? Well, it isn't quite that simple. The faster you go, the faster he goes. What's the point in going fast? If you go fast, you get more points.

Here's how the game works. You get to choose your skill level. Hit a key between 1 and 9. After you choose your skill level, the maze is drawn, the treasures are put in, and you appear in the upper left-hand corner. Take off!

The monster will follow in your exact footsteps, so you can duck into a side passageway and let him go past if you know where you have been. If you are trapped, try to make him accelerate. When he is accelerating, you can run past him. Do this by moving your joystick back and forth as fast as you can.



Getting away is difficult in "Superchase."

If you clear the maze of all the treasures, you will receive a bonus, depending on your skill level and score, and will start a new screen with a higher skill level. Don't be disappointed if you don't get a good score the first time. I find that most people learn quickly.

Program 2-3. Superchase

```
40 GOSUB14000
42 POKE1,0:POKE2,0
45 GOSUB12000:CLR:SK=PEEK(0):P=PEEK(1)*2
   56+PEEK(2)
100 GOTO10000
1000 M$=""
1110 POKEDD,127:P1=PEEK(D1)ANDAD:P2=PEEK
     (D2)
1120 IFP1=58THENM$="+{LEFT}{UP}":PRINTM$
     ;:Y=Y-1:C$="{DOWN}":CX=0:CY=1:GOTO1
     160
1130 IFP2=119THENM$="+{LEFT}{RIGHT}":PRI
     NTM$;:X=X+1:C$="{LEFT}":CX=-1:CY=0:
     GOTO1160
1140 IFP1=46THENM$="+{2 LEFT}":PRINTM$;:
     X=X-1:C$="{RIGHT}":CX=1:CY=0:GOTO11
     60
1150 IFP1=54THENM$="+{LEFT}{DOWN}":PRINT
     M$;:Y=Y+1:C$="{UP}":CY=-1:CX=0:GOTO
     1160
1155 GOTO1300
1160 IFFNCH(S)=WLTHENPRINTC$;:X=X+CX:Y=Y
     +CY:GOTO1300
1170 F$=F$+RIGHT$(M$,1)
1180 IFFNCH(S)=DITHENP=P+100*(EL-S):PC=P
     C+1
1190 IFFNCH(S)=SPTHENP=P+50*(EL-S):PC=PC
     +1
1200 IFFNCH(S)=CLTHENP=P+30*(EL-S):PC=PC
     +1
1210 IFFNCH(S)=HETHENP=P+20*(EL-S):PC=PC
     +1
1220 IFFNCH(S)=CITHENP=P+10*(EL-S):PC=PC
     +1
```

```
1250 J$=STR$(P*SK):FORJ=1TOLEN(J$):POKES
     C+J+489,ASC(MID$(J$,J,1)):NEXT
1300 PRINT"Q{LEFT}";
1310 IFPC>=61THENPRINTDN$"NO MORE TREASU
     RE.";:GOTO7000
1900 RETURN
2000 IFLEN(F$)>=30THENGOSUB3000
2005 FM=FM+1:IFFM/S<>INT(FM/S)THENRETURN
2006 FORH=1TOSKL:
2007 POKEFNPLOT(0),32
2010 J$=LEFT$(F$,1):F$=MID$(F$,2)
2030 POKEV,15:POKES1,254-LEN(F$):FORM=1T
     010:NEXT:POKE36875,0
2100 IFJ$="{UP}"THENYF=YF-1:GOTO2200
2110 IFJ$="{RIGHT}"THENXF=XF+1:GOTO2200
2120 IFJ$="{DOWN}"THENYF=YF+1:GOTO2200
2130 IFJ$="{LEFT}"THENXF=XF-1:GOTO2200
2150 GOTO2200
2200 POKEFNPL(0),42
2205 NEXT
2210 RETURN
3000 POKEFNPL(0),32
3007 S=S-1:IFS<1THENS=1
3008 J$=STR$(EL-S):FORJ=1TOLEN(J$):POKES
     C+J+502,ASC(MID$(J$,J,1)):NEXT
3010 FORC=1TO10:J$=MID$(F$,C,1):IFJ$="
     {UP}"THENYF=YF-1:GOTO3100
3020 IFJ$="{DOWN}"THENYF=YF+1:GOTO3100
3030 IFJ$="{RIGHT}"THENXF=XF+1:GOTO3100
3040 IFJ$="{LEFT}"THENXF=XF-1:GOTO3100
3100 POKEFNPL(0),42
3150 IFC/SK=INT(C/SK)THENGOSUB1000
3310 FORM=CTOC+2:POKES2,M*3+130:FORN=1TO
     10:NEXT:NEXT:POKES2,0
3350 POKEFNPLOT(0),32
3400 NEXT
3500 F$=MID$(F$,EL):RETURN
4000 IFX=XFANDY=YFTHENPRINTDN$"A TASTY M
     ORSEL INDEED!";:GOSUB6000:GOTO11000
4500 RETURN
6000 POKE36877,220:FORL=15TO0STEP-1:POKE
     36878,L:FORM=1TO300:NEXT:NEXT:POKE3
     6877,0:POKE36878,15
```

```
6010 RETURN
7000 FORK=1TO30
7005 POKE36876,220:FORL=1TO5:NEXT:POKE36
     876,0:FORL=1TO5:NEXT:POKE36876,200:
     FORL=1TO5:NEXT
7010 POKE36876,0:FORL=1TO5:NEXT:NEXT
7100 J=INT(P/256):POKE1,J:POKE2,P-J*256
7200 SK=PEEK(0)+1:POKE0,SK:GOTO45
7999 GOTO7000
8000 FORM=1TO500:GOSUB1000:IFLEN(F$)<20T
     HENNEXT
8010 FORJ=8142TO8142+20:POKEJ,32:NEXT
8100 GOSUB1000:GOSUB2000:GOSUB4000:GOTO8
     100
10000 DN$="{HOME}{21 DOWN}{RVS}{WHT}"
10030 S=10:PC=0:SC=7680:RO=22
10050 DEFFNPL(XX)=(YF*RO+XF)+SC
10060 DEFFNCH(XX)=PEEK((Y*RO+X)+SC)
10077 SO=10:POKE36878,15
10100 DD=37154:D1=37151:D2=37152:AD=63
10110 WL=102:DI=90:SP=65:CL=88:HE=83:CI=
      87:EL=11
10120 V=36878:S1=36875:S2=36876
10500 TI$="000000"
10600 PRINTDN$"{7 SPACES}GO !!!!!!
      {5 SPACES}"
10700 PRINTDN$"{OFF}{DOWN}{BLK}SCORE:
      {WHT}{7 SPACES}{BLK}SPEED:{WHT} 1
      {HOME}"
10800 PRINT"{HOME}{RIGHT}{DOWN}";:X=1:Y=
      1:XF=1:YF=1
10900 GOTO8000
11000 REM GAME OVER
11010 POKE37154,255
11105 PRINT:PRINT"{DOWN}TIME WAS ";MID$(
      TI$,3,2);" MINUTES, ";RIGHT$(TI$,2
      );" SECONDS"
11110 PRINT"PLAY AGAIN ? {@}{LEFT}
      ";
11120 GETJ$:IFJ$=""THEN11120
11130 PRINTJ$:IFJ$="N"THENEND
11140 IFJ$="Y"THENRUN
```

```
11150Ø PRINT:PRINT"{UP}";:GOTO11111Ø
12000Ø DIMA(3):A(Ø)=2:A(1)=-44:A(2)=-2:A(
       3)=44:WL=1Ø2:HL=32:SC=768Ø:A=SC+23
       :J=RND(-TI)
12Ø1Ø POKE36879,11Ø
12100Ø PRINTCHR$(142)"{CLR}{YEL}{OFF}";:F
       ORI=1TO21:PRINT"{21 +}":NEXT:POK
       EA,4
12200Ø J=INT(RND(1)*4):X=J
12205 B=A+A(J)
12210Ø IFPEEK(B)=WLTHENPOKEB,J:POKEA+A(J)
       /2,HL:A=B:GOTO12200Ø
12240Ø J=(J+1)*-(J<3):IFJ<>XTHEN12205
12250Ø J=PEEK(A):POKEA,HL:IFJ<4THENA=A-A(
       J):GOTO12200Ø
12300Ø PRINT"{HOME}{DOWN}{RIGHT}V"
12305 READJ,K,C:IFJ<ØTHEN12500Ø
12310Ø FORA=1TOJ
12320Ø B=INT(RND(1)*41Ø):IF(B-21)/22=INT(
       (B-21)/22)THEN12320Ø
12330Ø IF PEEK(B+77Ø2)<>32THEN12320Ø
12340Ø POKEB+77Ø2,K:POKEB+38422,C
12350Ø NEXT:GOTO12305
12400Ø DATA2,9Ø,1,4,65,Ø,7,88,5,9,83,2,39
       ,87,3,-1,Ø,Ø
12450Ø PRINT"12450:P="P
12500Ø RETURN
14000Ø POKE36879,46
14010Ø PRINTCHR$(14)"{CLR}{WHT}{6 DOWN}
       {2 RIGHT}{4 SPACES}{1Ø @}
       {12 SPACES}{RVS}SUPERCHASE{OFF}"
14050Ø PRINT"{5 DOWN} SKILL LEVEL (1-9):
       {RVS} {OFF}{2 LEFT}";
14060Ø GETJ$:IFJ$=""THEN14060Ø
14070Ø SK=VAL(J$):IFSK<1ORSK>9THEN14060Ø
14075 POKEØ,SK
14080Ø PRINTSK:POKEØ,SK:RETURN
```

Lochinvar's Maze

Clark and Kathryn H. Kidd

This maze keeps changing. The object is to move across the screen as efficiently as possible. Points are deducted from your score when you run into a wall. Caution is more important than speed.

The Legend of the Maze

For thousands of years, leprechauns have wandered the hills and valleys of the little country of Ireland. Irishmen look for leprechauns because leprechauns are rich and magical. If an Irishman catches a leprechaun, the leprechaun must grant his wishes and make him rich.

Each leprechaun has his own pot of gold, but the leprechauns want to keep their gold and not share it with the big people, the human population of Ireland. So leprechauns have become very tricky. Very few Irishmen have ever seen a leprechaun, and nobody has ever been able to keep a leprechaun's pot of gold.

Lochinvar is king of the leprechauns. He is king because he's the trickiest of all the Little People. He doesn't have a pot of gold, however. Lochinvar's treasure is in diamonds.

Lochinvar has hidden his diamond treasure at the end of a maze. The maze is built of tall hedges, like the hedge mazes in many gardens throughout the world. Lochinvar's maze is different, though: it moves. The maze changes shape as people try to go through it, making it hard to reach Lochinvar's diamonds.

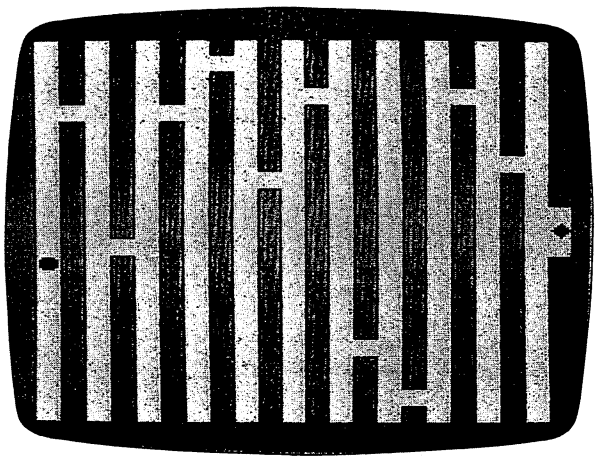
Although Lochinvar is tricky, you are fast. If you are fast enough, you may be able to reach Lochinvar's diamonds before the maze confuses you and you become lost. Don't go too fast, however. The maze is built of thorns, and whenever you run into a hedge wall you'll get a skin full of prickles.

How to Play

"Lochinvar's Maze" is a game of skill and speed. You must move through the maze and find the diamonds. If you run into a wall, a beep will sound, and points will be deducted from your score.

You'll begin the game on the left side of the maze. To move upward, press F1. To move down, press F3. Press the letter A to move across the board. Move as quickly as you can without touching the sides or walls of the maze. The game continues until you touch the diamond on the right side of the maze.

At the end of the game, you will receive a score ranked from 1 to 100. The score is based on the time it took you to reach the diamond treasure, minus the number of times you ran into the moving hedge. A score of 80 or above qualifies you as an expert diamond hunter. If you reach 90, you may want to move to Ireland to take on the other leprechauns.



The maze will continue to change as you try to cross the screen.

Program 2-4: Lochinvar's Maze

```
100 P1=4*(PEEK(36866)AND128)+64*(PEEK(368
    69)AND120):P2=37888+4*(PEEK(36866)AN
    D128)
```

```
125 POKE36879,27
130 PRINT"{CLR}{DOWN}WELCOME TO...
    {2 DOWN}":PRINT"{5 SPACES}LOCHINVAR'
    S{2 SPACES}":PRINT"{2 DOWN}
    {7 SPACES}M A Z E"
140 INPUT "{4 DOWN}INSTRUCTIONS (Y/N)";X$
150 IFX$="N"THEN500
160 IFX$="Y"THEN5000
170 GOTO140
500 POKE 36879,28:PRINT"{CLR}"
505 PC=0
510 FORX=1TO21STEP2
520 Z=INT(RND(1)*23)
530 FORY=0TO22
540 IFY=ZANDX<>21THEN560
550 POKEP1+X+(Y*22),160:POKEP2+X+(Y*22),4
560 NEXTY:NEXTX
570 POKEP1+242,81:POKEP2+242,0
580 POKEP1+241,32:POKEP2+241,1:POKEP1+263
    ,90:POKEP2+263,0:POKEP1+285,32:POKEP
    2+285,1
590 X=11:Y=0:MN=220:MD=50:GOSUB700:MN=230
    :MD=100:GOSUB700:GOTO1000
700 POKE36878,15:POKE36876,MN:FORQQ=1TOMD
    :NEXTQQ
710 POKE36878,0:POKE36876,0:RETURN
1000 AA=INT(TI/60)
1005 LC=0
1010 Z=PEEK(197)
1020 IFZ=17THEN1100
1040 IFZ=39THEN1300
1050 IFZ=47THEN1400
1060 LC=LC+1
1070 GOTO4000
1100 X1=X:Y1=Y+1:GOTO2000
1300 X1=X-1:Y1=Y:GOTO2000
1400 X1=X+1:Y1=Y
2000 IFX1<0ORX1>22ORY1<0ORY1>21THEN2080
2010 Z=PEEK(P1+(X1*22)+Y1)
2020 IFZ=90THEN3000
2030 IFZ<>32THEN2080
2040 POKEP1+(X*22)+Y,32:POKEP2+(X*22)+Y,1
```

```
2050 X=X1:Y=Y1
2060 POKEP1+(X*22)+Y,81:POKEP2+(X*22)+Y,0
2070 GOTO1010
2080 MN=128:MD=70:GOSUB700:PC=PC+1:GOTO10
     10
3000 ZZ=INT(TI/60)
3001 POKEP1+(X*22)+Y,32
3002 MN=220:MD=50:GOSUB700:MN=230:MD=100:
     GOSUB700
3005 GETX$:IFX$<>""THEN3005
3010 PRINT"{CLR}{DOWN}TOTAL SECONDS:"ZZ-A
     A
3015 PRINT"{DOWN}PENALTIES:"PC
3016 X=115-(ZZ-AA)-(PC*5)
3017 IFX>100THENX=100
3018 IFX<1THENX=1
3019 PRINT"{DOWN}RANKING(1-100):"X"
     {4 DOWN}"
3020 INPUT "{2 DOWN}ANOTHER MAZE (Y/N)";X
     $
3030 IFX$="N"THENPOKE36879,27:PRINT"{CLR}
     ":END
3040 IFX$="Y"THEN500
3050 GOTO3020
4000 ONLCGOTO4010,4020,4030,4040,4050,406
     0,4070,4080
4010 L1=INT(RND(1)*10):GOTO1010
4020 L2=INT(RND(1)*23):GOTO1010
4030 L3=P1+1+(L1*2):GOTO1010
4040 L4=0:GOTO1010
4050 IFPEEK(L3+(L4*22))=32THEN1010
4052 L4=L4+1:IFL4>22THEN1005
4054 LC=LC-1:GOTO1010
4060 L6=L3+(L2*22):GOTO1010
4070 IFPEEK(L3+(L4*22))<>32THEN1005
4075 POKE(L3+(L4*22)),160:POKEP2+(L3-P1+(
     L4*22)),4:GOTO1010
4080 POKEL6,32:POKEP2+(L6-P1),1:GOTO1005
5000 PRINT"{CLR}{DOWN}{2 SPACES}THIS MAZE
      IS A TEST {DOWN}OF SKILL AND SPEED
     .{3 SPACES}{DOWN}YOU({BLK}Q{BLU}) M
     UST MOVE"
```

```
5010 PRINT"{DOWN}THROUGH THE MAZE AND
      {2 SPACES}{DOWN}FIND THE DIAMONDS(
      {BLK}Z{BLU}).{DOWN}"
5020 PRINT"THIS IS COMPLICATED BY{DOWN}TH
      E FACT THAT THE MAZE{DOWN}WILL SOME
      TIMES CHANGE "
5030 PRINT"DURING THE GAME."
5040 GOSUB5900
5050 PRINT"{CLR}{DOWN}IF YOU CRASH INTO A
      {3 SPACES}{DOWN}WALL, YOU WILL HEAR
       A {DOWN}'BEEP' AND WILL GET A"
5060 PRINT"{DOWN}PENALTY.{2 SPACES}YOUR F
      INAL{2 SPACES}{DOWN}SCORE WILL BE B
      ASED ON{DOWN}THE TIME SPENT IN THE"
5070 PRINT"{DOWN}MAZE, ADJUSTED FOR
      {4 SPACES}{DOWN}PENALTIES."
5080 GOSUB5900
5090 PRINT"{CLR}{DOWN}THREE KEYS ARE USED
       TO{DOWN}MOVE THROUGH THE MAZE:
      {3 DOWN}"
5100 PRINT"F1 - MOVES YOU UP{5 SPACES}
      {DOWN}F3 - MOVES YOU DOWN{3 SPACES}
      {DOWN} A - MOVES YOU ACROSS"
5110 PRINT"{3 DOWN}G O O D{3 SPACES}L U C
       K !{SHIFT-SPACE}!"
5120 GOSUB5900
5130 GOTO500
5900 PRINT"{3 DOWN}{3 SPACES}(PRESS ANY K
      EY)"
5910 GET X$:IF X$=""THEN5910
5920 RETURN
```

Part 3

Action Games

Closeout

L.L. Beh
Translated for the VIC by Charles Brannon.

"Closeout" uses the concept of artificial intelligence to create an interesting chase game.

"Closeout" uses almost all the memory that the unexpanded 5K VIC can offer. Don't enter any extra spaces or semicolons. It will run on any size VIC. "Floating" memory is handled in lines 180 and 190. Instead of using TX=7680, the start of screen memory on a 5K VIC, the formula in line 180 will return the proper address for any VIC.

Scrambling for Bargains

There's a huge sale going on at a local department store. You arrive at the multistory building hungry for bargains. Boldly, you enter the store and look around — and see bargains galore. A real sale! You start gathering up sale items, but then become aware of a strange group of shoppers. Wherever you go, they follow you around.

The object of Closeout is to snatch up as many sale items as possible while evading the hostile bargain hunters. Using the I,J,K,M keys (I=up, M=down, J=left, K=right) move yourself (represented by the "pi" symbol) around the department store, avoiding the rapacious bargain hunters who mercilessly pursue you. Pick up the various sale items by moving your character over the colored dots. You can ascend and descend escalators to move from floor to floor. If a fellow shopper gets too aggressive, you can deliver a shove that will send him reeling back to the top floor. You start out with three shoves, and you get two more every time you acquire 120 sale items (at which point you move on to a whole new store). Press SPACE to deliver a shove, but be careful not to run out of them. You cannot use your shoves while you're on an escalator. You have no time limit to worry about.

Special Techniques

In an attempt to get the most speed from a BASIC program while using the least amount of memory, there are several

tricks here that may be useful to VIC programmers. First, variables replace much-used constants. For example, the number 22 (number of characters per line) is used extensively in the screen POKE statements. Initially setting Q=22 at the start of the program lets you substitute Q for the constant 22. Aside from the convenience of this, using variables is much faster. Instead of converting the characters "22" to the floating point equivalent, the computer only has to look up the value of Q. It seems trivial, but judicious use of variables can significantly speed up your programs. Also note that the letter "O" is used instead of zero in some places for the same reason.

Artificial Intelligence?

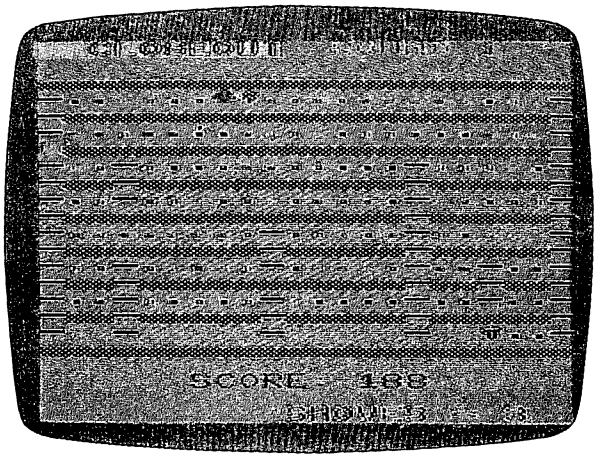
A game like this is the perfect opportunity to fiddle with artificial intelligence. When you write a routine that makes a character chase another, you've simulated a simple animal's "instincts," or predetermined behavior.

The crazed shoppers in Closeout must know how to home in on you. Their behavior must include the ability to get on and off the escalators. One thing you should watch out for when writing games like this is predictability. If your "intelligence" subroutine is too good, your creatures will accurately home in on the victim, but they will act in predictable ways. Sometimes it helps to stir a little randomness in with the RND function (we humans call such "careful" randomness *creativity*).

Since all three pursuers use the same "intelligence" algorithm, we've just created "mini-arrays" that hold important variables for each character: the X and Y position, the character used, the current direction, and the character "underneath" the pursuer. To move a character nondestructively, you have to save and restore the background characters. The routine can move any of the three chasers according to E, the "Enemy index." To create the illusion of simultaneous motion without slowing down the player too much, only one chaser can move for every move the player makes.

Enhancements

If you have more than the normal 5K RAM memory in your VIC, you might want to make some enhancements to the game. You could add custom characters, improved sound effects, and more pursuers.



You'll be grabbing up bargains when you play "Closeout."

Program 3-1. Closeout

```
100 REM VIC-20 CLOSEOUT
110 DIM EX%(2),EY%(2),EC%(2),EP%(2),PC%(
    2),DX%(2),DY%(2),SV%(21)
120 S1=36874:S3=S1+3:V=36878:POKES1,0:PO
    KEV,0:S2=S1+1
130 R=0:PTS=0:SH=1:O=0:CD$=CHR$(19):FORI
    =1TO23:CD$=CD$+CHR$(17):NEXT
140 PRINTCHR$(147)CHR$(18)CHR$(156)"
    {2 SPACES}CLOSEOUT"
150 R=R+1:PRINTCHR$(19)TAB(12)CHR$(30)"R
    OUND"R
160 SH=SH+2:GOSUB900
170 SC=36879:Q=22:M=O
180 TX=4*(PEEK(36866)AND128)+64*(PEEK(36
    869)AND120)
190 COLOUR=37888+4*(PEEK(36866)AND128)
200 FORI=44TO65:POKETX+I,104:POKECO+I,6:
    NEXT
210 FORI=88TO400STEP44
220 FORJ=ITOI+21
230 POKETX+J,104:POKECO+J,6
240 POKETX+J-Q,46:POKECO+J-Q,6*RND(O)+2
250 NEXT:NEXT
```

```
260 FORI=66TO378STEPQ:POKETX+I,64:POKECO
    +I,O:POKETX+21+I,64:POKECO+21+I,O:NE
    XT
270 FORI=1TO10
280 SX=INT(7*RND(O))*3+3:SY=5+INT(6*RND(
    O))*2
290 LN=2+INT(3*RND(O))*2:IF SY+LN>18 THE
    N 290
300 FORJ=SY*QTO(SY+LN)*QSTEPQ
310 POKETX+SX+J,64:POKECO+SX+J,O
320 NEXT:NEXT
330 EX%(O)=1:EX%(1)=2:EX%(2)=3:FORI=OTO2
    :EY%(I)=3:NEXT
340 EC%(O)=65:EC%(1)=88:EC%(2)=81
350 DX%(O)=1:DX%(1)=1:DX%(2)=-1
360 FORI=OTO2:EP%(I)=46:PC%(I)=INT(6*RND
    (O)+2):NEXT
370 PX=2:PY=17:PC=94:DX=O:DY=O:PK=32
380 GETA$:IFA$<>""THENB$=A$
390 IFB$=" "THENGOSUB800
400 IFB$="I"THENDY=-1:DX=O
410 IFB$="M"THENDY=1:DX=O
420 IFB$="J"THENDX=-1:DY=O
430 IFB$="K"THENDX=1:DY=O
440 CP=TX+PX+Q*PY
450 IFPK<>64ORPEEK(CP+Q*DY)=104THENDY=O
460 IFPEEK(CP+DX)=104THENDX=O:B$=""
470 POKE CP,PK:POKECO+CP-TX,CC
480 PX=PX+DX:PY=PY+DY
490 IFPX<0THENPX=O:B$="":DX=O
500 IFPX>21THENPX=21:B$="":DX=O
510 NP=TX+PX+Q*PY:CC=PEEK(CO+NP-TX)
520 PK=PEEK(NP):IFPK<>46ANDPK<>42THEN560
530 PTS=PTS+1:IFPK=42THENPTS=PTS+49
540 GOSUB980:PK=32:FORI=15TO0STEP-5:POKE
    S3,255-I:POKEV,I:NEXT:POKES3,0
550 M=M+1:IFM=120THEN140
560 IFPK=EC%(O)ORPK=EC%(1)ORPK=EC%(2)THE
    N730
570 POKENP,PC:POKECO+PX+Q*PY,2
580 E=-(E+1)*(E<2)
590 EX=EX%(E):EY=EY%(E):EC=EC%(E):XX=DX%
    (E):YY=DY%(E):EP=EP%(E):C=PC%(E)
```

```
600 POKETX+EX+Q*EY,EP:POKECO+EX+Q*EY,C
610 CP=TX+EX+Q*EY
620 IF(PEEK(CP-Q)=64ORPEEK(CP+Q)=64)ANDR
    ND(1)>.1THENXX=0:YY=SGN(PY-EY)
630 IFPEEK(CP+YY*Q)=104OR(EY=PYANDEY/2<>
    INT(EY/2))THENYY=0:XX=SGN(PX-EX)
640 EX=EX+XX:EY=EY+YY
650 IFEX=OOREX=21THENXX=-XX
660 NP=TX+EX+Q*EY:EP=PEEK(NP):C=PEEK(CO+
    NP-TX)
670 IFEP=PCTHENPOKENP,161:GOTO730
675 IFEP<>EC%(0)ANDEP<>EC%(1)ANDEP<>EC%(
    2)THEN700
680 FORI=0TO2:IFEP<>EC%(I)THENNEXT:STOP
690 EP=EP%(I):C=PC%(I)
700 POKENP,EC:POKECO+NP-TX,3+E
710 EX%(E)=EX:EY%(E)=EY:EP%(E)=EP:PC%(E)
    =C:DX%(E)=XX:DY%(E)=YY
720 GOTO380
730 FORI=128TO255STEP2:POKESC,I:POKES3,I
    :POKEV,(I-128)/8:NEXT
740 POKESC,27:PRINTCHR$(19)CHR$(18)CHR$(
    156)"{2 SPACES}NABBED! "
750 PRINTLEFT$(CD$,23);CHR$(31);"PRESS "
    CHR$(18);
760 PRINTCHR$(156)"SPACE"CHR$(146)CHR$(3
    1)" TO REPLAY";
770 FORI=1TO10:GETA$:NEXT
780 GETA$:IFA$<>" "THEN780
790 RUN
800 REM SHOVE!
810 IFDX=0THENPOKESC,31:POKES2,200:POKEV
    ,10:FORW=1TO100:NEXT:POKESC,27:POKES
    2,0:RETURN
820 IFSH=0THENPOKESC,28:POKES2,255:POKEV
    ,10:FORW=1TO100:NEXT:POKESC,27:POKES
    2,0:RETURN
830 CP=TX+Q*PY:LC=64:B$=""
840 FORI=PXTO-21*(DX>0)STEPDX
850 SV%(I)=PEEK(CP+I):POKECP+I,LC:LC=131
    -LC:POKES1,LC:POKEV,(IAND15)
```

```
860 IFSV%(I)=EC%(0)ORSV%(I)=EC%(1)ORSV%(
    I)=EC%(2)THENGOSUB920
870 NEXTI:POKES1,0:POKEV,0
880 FORI=PXTO-21*(DX>0)STEPDX
890 POKECP+I,SV%(I):NEXT:SH=SH-1
900 PRINTCHR$(156);LEFT$(CD$,23);TAB(10)
    "SHOVES ="SH;
910 RETURN
920 FORJ=0TO2:IFSV%(I)<>EC%(J)THENNEXT:S
    TOP
930 SV%(I)=EP%(J):POKECO+CP+I-TX,PC%(J)
940 EX%(J)=INT(20*RND(0)+1):EY%(J)=3
950 DX%(J)=-1:IFRND(0)>.5THENDX%(J)=1
960 DY%(J)=0:EP%(J)=46
970 PTS=PTS+50:FORJ=0TO15:POKES3,128+J:P
    OKEV,15-J:NEXT:POKES3,0:POKEV,0
980 PRINTCHR$(159)LEFT$(CD$,21)"
    {6 SPACES}SCORE:"PTS;
990 RETURN
```

Marble Hunt

Ronny Ong

"Marble Hunt," for the unexpanded VIC, is a maze game without a maze.

"Marble Hunt" is a unique kind of maze game. It has no maze! Without guiding corridors, the player must exercise joystick control while keeping up with the fast pace of the game.

When the program is run for the first time, there is a short pause as a few custom characters are being defined. Then a title page is presented with mood-setting "music." Be ready, because although the program won't start you (the marble hunter) near the neighborhood bully, the bully will know your location before you know his.

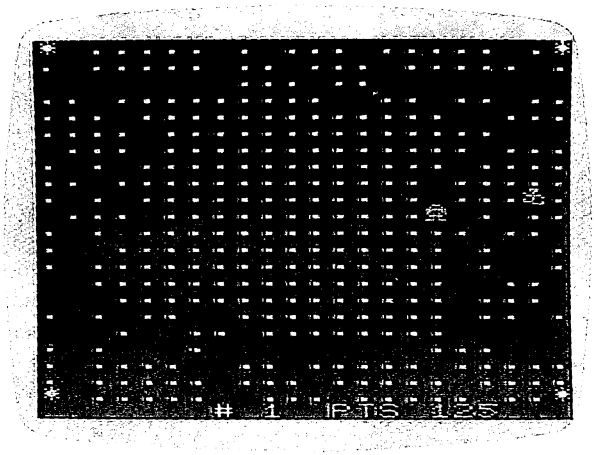
Below the playfield is the status line showing the board number and score. At each corner are vitamins. The white dots littering the rest of the screen are the marbles. The object of the game is for the marble hunter (in yellow) to gather up all the marbles.

The neighborhood bully is in red. Regardless of his location, he is forever trying to pounce on the hunter and steal his marbles. This hyperactivity often causes the bully to miss the marble hunter even when he manages to close in.

Scoring Points

Each marble (white dot) collected gives the hunter one point. If the bully catches him, the marble hunter loses 75 points or more depending on the skill level reached. The game is over when the hunter loses more points than he has accumulated.

Eating a vitamin gives the marble hunter two points plus the ability to scare off the bully — who turns white with fear. The vitamin's effects soon wear off, especially at higher skill levels. A warning appears on the status line and a sound tells you the bully's courage is returning. Catching the bully while he is frightened is worth ten points, and picking up all the marbles is worth five. The board number eventually rolls over, easing the game difficulty should you get so far.



The bully tries to pounce on the marble hunter and steal his marbles.

The RUN/STOP key is disabled. At the end of a game, pressing N enables RUN/STOP, disables the custom character set, and exits the program. Pressing Y or the fire button on the joystick restarts the program.

Game of Strategy

Marble Hunt is a game of strategy as well as hand/eye coordination. There is a full wraparound if you try to go past an edge of the playfield. Since the bully will usually be thrown to the opposite side, an edge row or column is the safest place to be. Right after the bully is transported away from the hunter is the best time to rack up points with the marbles near the edges. Plentiful points, along with careful use of vitamins, will help in the risky process of clearing the marbles in the center of the screen.

There are many other keys to good play that are left for you to figure out.

Program 3-2. Marble Hunt

```
1 IFPEEK(36869)<>255THEN25Ø
4 PRINT"{CLR}{8 DOWN}{5 RIGHT}{YEL}MARBL
  E HUNT!":TI$="ØØØØØØ"
6 Z=Ø:POKE36879,47:C=Z:I=RND(-TI):DEFFNR
```

```
  N(G)=INT(RND(1)*G)+1:POKE36878,8:GOSUB
  500
7 DEFFNJH(G)=((PEEK(37151)AND16)=0)-((PE
  EK(37152)AND128)=0):POKE37154,127:POKE
  36879,9
8 DEFFNJV(G)=((PEEK(37151)AND4)=0)-((PEE
  K(37151)AND8)=0)
10 F=0:P=F:PRINT"{CLR}{GRN}*{WHT}.......
   .............{GRN}*{WHT}";:FORI=FTO43
   9:PRINT".";
11 NEXT:PRINT"{GRN}*{WHT}...............
   .....{GRN}*";
12 C=C+1+9*(C=9):PRINT"{HOME}{22 DOWN}
   {7 RIGHT}#"C;
15 A=FNRN(21):B=FNRN(21)
20 X=FNRN(21):Y=FNRN(21):IFSQR((A-X)*(A-
   X)+(B-Y)*(B-Y))<9THEN20
25 T=7680+B*22+A:IFPEEK(T)=42THENP=FNRN(
   30)+30-C*3:Z=Z+2:POKE36875,150
26 IFPEEK(T)=46THENF=F+1:POKE36875,228
27 IFF<>480THEN30
28 Z=Z+5+F:PRINT"{CLR}{8 DOWN} BOARD CLE
   ARED IN "LEFT$(TI$,2):POKE36875,0
29 PRINT"{RIGHT}HOURS, "MID$(TI$,3,2)" M
   INUTES.":GOSUB500:GOTO10
30 POKET,27:POKET+30720,7:L=7680+Y*22+X:
   M=PEEK(L):MX=PEEK(L+30720)
40 POKEL,29:POKEL+30720,2+(P>0)
50 B=B+FNJV(0):A=A+FNJH(0):POKE36875,0
70 PRINT"{HOME}{22 DOWN}{12 RIGHT}PTS"Z+
   F"{LEFT} ";
80 POKET,28:IFA>21THENA=0
85 IFA<0THENA=21
90 IFB>21THENB=0
95 IFB<0THENB=21
110 X=X+SGN(A-X)*ABS(P=0)+FNRN(3)-2:Y=Y+
    SGN(B-Y)*ABS(P=0)+FNRN(3)-2:IFX>21TH
    ENX=0
113 IFX<0THENX=21
115 IFY>21THENY=0
116 IFY<0THENY=21
120 POKEL,M:POKEL+30720,MX
```

```
123 IFP=ØTHEN13Ø
125 P=P-1:IFP<9THENPRINT"{HOME}{22 DOWN}
    ALERT!";:POKE36876,188-P
127 IFP=ØTHENPRINT"{HOME}{22 DOWN}
    {6 SPACES}";:POKE36876,Ø
13Ø POKET,32:POKET+3Ø72Ø,Ø:IFX<>AORY<>BT
    HEN25
135 POKE36877,22Ø:FORI=1TO3:FORII=25TO3Ø
    :POKE36865,II:NEXT
136 FORII=3ØTO25STEP-1:POKE36865,II:NEXT
    :NEXT:POKE36877,Ø
14Ø IFPTHENZ=Z+1Ø:GOTO2Ø
145 Z=Z-6Ø-C*15:IFZ+F>ØTHEN15
146 PRINT"{HOME}{22 DOWN}GAME OVER-PLAY
    AGAIN?";:POKE198,Ø
147 GETG$:IF(PEEK(37151)AND32)=ØORG$="Y"
    THENRUN
148 IFG$<>"N"THEN147
15Ø FORI=ZTO12ØØ:NEXT:POKE37154,255:POKE
    36879,27
155 POKE56,PEEK(56)+2:POKE36869,24Ø:CLR:
    PRINT"{CLR}{BLU}";:POKE8Ø8,112:END
16Ø DATA 56,56,18,124,144,18,158,24Ø,56,
    56,144,124,18,144,242,3Ø
17Ø DATA 56,84,254,214,17Ø,124,4Ø,238
21Ø FORJ=I*8TOI*8+7:READK:POKEJ+CS,K:NEX
    T:RETURN
25Ø X=PEEK(56)-2:POKE52,X:POKE56,X:POKE5
    1,PEEK(55):CLR:PRINT"{CLR}ONE MOMENT
    ..."
255 CS=256*PEEK(52)+PEEK(51)
26Ø FORI=CSTOCS+511:POKEI,PEEK(I+32768-C
    S):NEXT:FORI=27TO29:GOSUB21Ø:NEXT
27Ø POKE36869,255:RUN
5ØØ FORI=1TO125:POKE36876,INT(RND(1)*128
    )+128:FORII=ØTO9:NEXT:NEXT:POKE36876
    ,Ø:RETURN
```

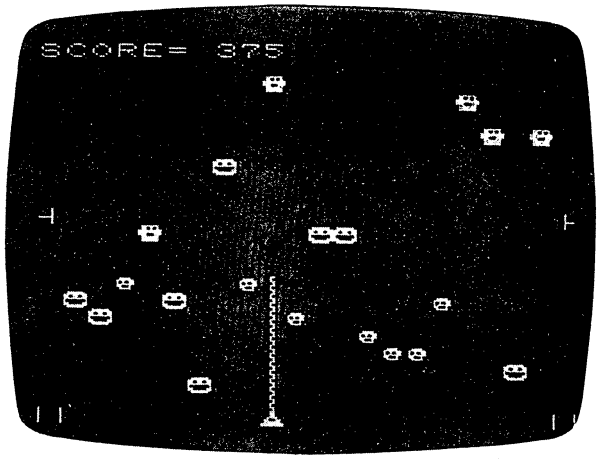
Balloons

Aaron Bobick

This game demonstrates how redefining characters can add to the graphic display of a game.

"Balloons" is a game for children and adults. The object of the game is to break each balloon before it hits the ground.

The balloons with the big smiles on their faces are worth 50 points each; the other two sizes are worth 35 and 25 points. Your pin launcher is located along the bottom of the screen. You can move your pin to the right or left by pressing the "A" and "D" keys. When you think you are in a position to pop a balloon, press F1. The game ends if a balloon hits the ground. But don't play it *too* safe. You get points only for balloons you pop in the bottom half of the screen.



In "Balloons" you try to pop all the balloons before they hit the bottom of the screen.

Program 3-3. Balloons

```
100 POKE 52,28:POKE 56,28:CLR
110 INPUT"{CLR}{DOWN}{9 RIGHT}2{HOME}SKI
    LL1,2,3{DOWN}{10 LEFT}1.EASY.";RR:PO
    KE36877,0
```

```
120 IFRR=1THENLL=499:PP=0
130 IFRR=2THENLL=999:PP=2
140 IFRR=3THENLL=1699:PP=3
150 PRINT"{CLR}{4 RIGHT}GET";LL+1;"POINT
    S"
160 PRINT"{2 DOWN}{7 RIGHT}TO WIN
170 PRINT"{5 SPACES}{2 DOWN}{RIGHT}{RVS}
    BALLOONS
190 FORX=7423TO7432:POKEX,0:NEXT
200 PRINT"{2 DOWN}{RIGHT}USE 'A' AND 'D'
     TO GO{DOWN}{RIGHT}RIGHT AND LEFT.
210 PRINT"{DOWN}{3 RIGHT}USE {RVS}F7
    {OFF} TO FIRE
220 DIMT(10):T(0)=50:T(2)=35:T(4)=25
230 DIMA(22),P(22),S(22):FORI=1TO21:A(I)
    =7702+I:P(I)=INT(RND(1)*3) *2
240 S(I)=T(P(I)):NEXT
250 C=13
260 CS=7168
270 REMORI=CSTOCS+511:POKEI,PEEK(I+32768
    -CS):NEXT
280 FORI=0TO8*C-1:READJ:POKECS+I,J:NEXT
290 DATA126,255,219,255,129,195,255,126
300 DATA24,24,24,60,102,126,255,255
310 DATA126,106,255,255,231,126,126,126
320 DATA16,24,8,24,16,24,8,24
330 DATA0,60,126,86,126,100,60,0
340 DATA24,16,24,8,24,16,24,8
350 DATA146,36,146,73,36,146,73,146
360 DATA136,132,80,10,164,2,20,65
370 DATA0,66,8,0,0,145,4,32
380 DATA0,4,0,1,68,16,4,145
390 DATA0,0,128,16,128,16,145,34,0,0,0,1
    28,193,243,255,255,60,126,231,254,25
    2,255,254,60
400 PRINT"{RVS}{DOWN}{6 RIGHT}HIT SPACE"
410 IFPEEK(203)<>32THEN410
420 POKE36869,255:PRINT"{CLR}{RVS}{WHT}
    {10 DOWN}{W}{20 RIGHT}
    {Q}"
430 POKE36879,110
440 FORI=7392TO7402:POKEI,129:NEXT:AC=128
```

```
450 E=8174:POKE8185,28:POKE8185-21,28
460 L=0:G=PEEK(203):IFG=17THENL=-1
470 IFG=18THENL=1
480 POKE36878,0
490 IFG=63THENGOTO600
500 IFXC>LLTHEN640
510 POKEE,32:E=E+L:IFPEEK(E)=28THENE=E-L
520 IFPEEK(E)<>32THEN730
530 POKEE,1
540 REM
550 FORCC=1TOPP
560 ZC=INT(RND(1)*20+1):POKEA(ZC),32:A(Z
    C)=A(ZC)+22:IFA(ZC)>8185THEN730
570 POKEA(ZC),P(ZC)
580 POKEE,1:
590 NEXTCC:GOTO460
600 S=3:D=E-8164:POKEE,1:FORI=E-22TOA(D)
    STEP-22:S=S+2+(S=5)*2:POKEI,S:POKE36
    876,S*20+128
610 POKE36876,0:POKE36878,15:NEXT:POKEA(
    D),32:FORI=A(D)TOESTEP22
620 POKEI,32:NEXT:XC=XC+S(D)+S(D)*(A(D)<
    7900):PRINT"{HOME}{RVS}{WHT}SCORE="X
    C:A(D)=7702+D:GOTO550
630 NEXTCC
640 POKE36878,15:FORI=1TO20:POKE36876,12
    8+I*6:FORJ=A(I)TO8163STEP22:POKEJ,P(
    I):GOSUB670
650 POKEJ,32:NEXT:GOSUB680:POKEJ,11:NEXT
    :PRINT"{HOME}{RVS}YOU WIN!!!!!!!!!!"
660 GOTO690
670 FORX=1TO15:NEXT:RETURN
680 POKE36878,15:FORX=128TO255STEP5:POKE
    36876,X:NEXT:POKE36878,0:RETURN
690 FORX=1TO500:NEXT:POKEE,1
700 FORX=1TO7:POKEE,1:POKE36878,15:FORO=
    1TO100:NEXTO:POKE36876,200
710 POKEE,32:FORR=1TO100:NEXTR:POKEE,32:
    POKE36878,0:NEXTX
720 GOTO820
730 POKE36878,15:REM LSE
740 POKE36874,15:FORKT=1TO4:FORQ=255TO15
```

```
    STEP-3:POKE36876,Q:NEXT:POKE36876,0:
    NEXT
750 POKEE-22,6:POKEE-1,6:POKEE+1,6:POKE3
    6878,5
760 POKE36877,129
770 FORX=1TO300:NEXT
780 POKE36878,9:POKEE-2,7:POKEE+2,7:POKE
    E-23,7:POKEE-21,7:FORX=1TO300:NEXT:P
    OKEE,230
790 POKE36878,12
800 POKEE-24,8:POKEE-45,9:POKEE-44,9:FOR
    X=1TO200:NEXT:POKE36878,14
810 POKEE-43,9:POKEE-20,10:POKE36878,15:
    FORX=1TO1999:NEXT:POKE36878,0
820 POKE198,0:POKE36869,240:PRINT"{CLR}
    {7 DOWN}{4 RIGHT}PLAY AGAIN
    {2 SPACES}{BB}{3 LEFT}";
830 INPUTA$:IF LEFT$(A$,1)="Y"THEN CLR:GO
    TO 110
840 POKE36879,27:PRINT"{CLR}"
850 PRINT"{BLU}{HOME}{11 DOWN}{2 RIGHT}T
    HANKS FOR PLAYING":END
```

— ERRATA SHEET —
COMPUTE!'s First Book of VIC Games

Richthofen's Revenge

Corrections: page 60

For disk users — switch steps 6 and 7 of instruction 3a.
For cassette users — replace steps on pages 60-61 with the following:

Follow steps 1-7 of instruction 3b as printed in the book. Then follow the instructions listed here:

8. LOAD & RUN LANDSCAPE DATA
9. LOAD & RUN CHARACTERS
10. Type: POKE55,0:POKE56,17:NEW
11. LOAD DATA CLOADER
12. Remove your program tape.
13. Insert & rewind your DATA TAPE
14. RUN DATA CLOADER
15. Type NEW. Enter 100 POKE657,128:SYS 5541
16. Type POKE55,0:POKE56,32:POKE43,1:POKE44,16: POKE 45,0:POKE46,30:SAVE"REVENGE!"
17. Reset the computer.

Richthofen's Revenge

Marc Sugiyama
Todd Koumrian
Chris Metcalf

"Richthofen's Revenge" is an all machine language game which will fit into the unexpanded VIC-20 and can be typed into the computer through BASIC. It's best to read through all the directions just before typing in this program.

At last, an all machine language game that you can type into your VIC-20 home computer. The program requires a joystick, but no memory expanders. You do need either a cassette or disk drive. If you are using a cassette drive, have two blank cassettes handy. Disk drive owners, you will need at least 82 blocks free on your game disk.

The Programs

"Richthofen's Revenge" is divided into several loader programs. These are "MEMDATA $1100" through "MEMDATA $1900," "DATA LOADER/CLOADER," "LANDSCAPE DATA," and "CHARACTERS." The MEMDATA programs create data files on disk or tape depending on line 40 of each program.

If you are using a cassette, change line 40 in each of the MEMDATA programs from OPEN1,8,2,"@0:name,S,W" to OPEN1,1,1,"name". Use the remark in line 40 of MEMDATA $1100 as a guide to changing the other MEMDATA programs. Be sure to use the proper name in each program.

If you study the MEMDATA programs, you may notice that there are 17 numbers in each line of DATA. The first 16 are the data which represent Richthofen's Revenge. The last number is a checksum, in this case the sum of the previous 16 numbers. By including this feature, the computer can determine which DATA lines contain mistyped numbers.

DATA LOADER/CLOADER reads the data disk or cassette and stores the information in RAM. Use DATA LOADER if

you have a disk drive, and DATA CLOADER if you have a cassette.

CHARACTERS contains the data necessary to create the user-defined character set. The remarks in the DATA will tell you what the characters are being used for in this program. LANDSCAPE DATA stores the data which represents the landscape. Both of these programs store their data directly into RAM. No checksum is provided for these programs, so take care that all of the numbers are typed correctly.

Typing the Programs

Now that you know what all of the programs are for, it is time to enter them into the computer. The proper sequence for creating a playable copy of Richthofen's Revenge is as follows:

1. Remove all RAM expansion before beginning.
2. Enter and SAVE all of the loader programs in the following order (this is very important if you are using a cassette):
   - Program 3-4. MEMDATA $1100
   - Program 3-5. MEMDATA $1300
   - Program 3-6. MEMDATA $1500
   - Program 3-7. MEMDATA $1700
   - Program 3-8. MEMDATA $1900
   - Program 3-9. LANDSCAPE DATA
   - Program 3-10. CHARACTERS
   - Program 3-11. DATA LOADER (Disk Users Only)
   - Program 3-12. DATA CLOADER (Cassette Users Only)

3a. If you are using a *disk:*

1. LOAD and RUN all MEMDATAs.
2. Reset the computer.
3. LOAD and RUN LANDSCAPE DATA.
4. LOAD and RUN CHARACTERS.
5. Enter POKE55,0:POKE56,17:NEW .
6. Enter NEW. Enter 100 POKE657,128: SYS5541.
7. LOAD and RUN DATA LOADER.
8. Enter POKE55,0: POKE56,32: POKE43,1: POKE44,16: POKE45,0: POKE46,30: SAVE"0:REVENGE!",8.
9. Reset the computer.

3b. If you are using a *cassette:*

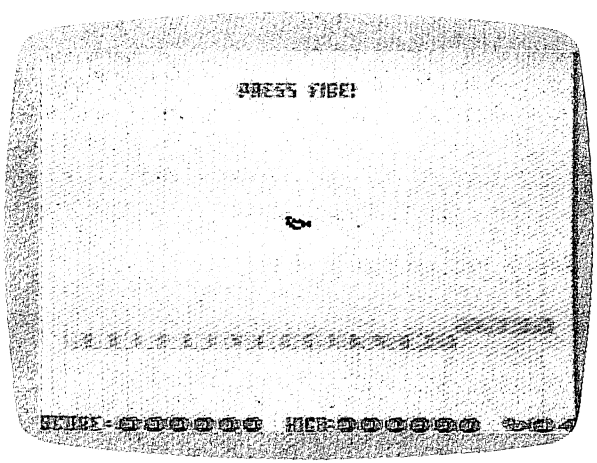
1. Have your second blank cassette handy.
2. LOAD MEMDATA $1100, but *do not* rewind the tape.

3. Insert your blank tape (which is now the DATA TAPE).
4. RUN MEMDATA $1100.
5. Remove the DATA TAPE and reinsert the program tape.
6. LOAD and RUN the other MEMDATA programs, following steps 2 to 5.
7. Reset the computer.
8. Follow steps 3 to 6 from 3a.
9. LOAD DATA CLOADER. Remove program tape.
10. Insert DATA TAPE and rewind. RUN DATA CLOADER.
11. Enter POKE55,0: POKE56,32: POKE43,1: POKE44,17: POKE45,0: POKE46,30: SAVE"REVENGE!".
12. Reset the computer.

4. To play Richthofen's Revenge, LOAD and RUN "REVENGE!".

Playing Richthofen's Revenge

If all goes well, when you LOAD and RUN Richthofen's Revenge, you should hear the first few measures of "Over There" followed by the message PRESS FIRE! Whenever you see this message, the computer is waiting for you to press the fire button of your joystick. Nothing will happen until you press the fire button. Once you press it, you will see a little



"Revenge, " a machine language game for the VIC.

black airplane in the center of the screen. This is you. Around you are white and red airplanes, and blue balloons. These are the enemy. Your mission is to shoot them down while avoiding collision with them.

But wait, where are you? The time is circa World War I. You are somewhere in Europe flying a sophisticated airplane with "hovering" abilities (you do not have constant horizontal motion). As you fly to the left you will see a small mountain, and then you will stop moving. Beyond the mountain is the ocean. If you were to fly out over the ocean, you would get lost and perish. Flying to the right also leads to a dead end — it is enemy territory, where you simply wouldn't survive.

As you play, you will notice that the enemy does not shoot at you. The reason is simple; the airplanes flying out to the ocean are saving their ammunition for their attack on an island nation somewhere out to the left. As they return home to their land, they have no ammunition left to fire at you. Balloons, for surveillance, have no weapons and are at your mercy.

You gain 75 points for every enemy airplane you destroy, and 50 points for every balloon. The enemy comes in waves. See the table on page 74 for the number of enemy aircraft in each level. For every wave you eliminate, you gain another airplane (the number of airplanes you have in reserve is shown in the lower right-hand corner of the screen). There is an upper limit of 64 enemy craft.

When a wave is eliminated, the computer plays a short victory song, and it goes on to the next level. Finally, when all of your airplanes are destroyed, the computer will play taps and then return to the power-up screen. The computer maintains the high score until it is turned off or the program resets.

So far, our high score is 11,450.

Program 3-4. MEMDATA $1100

```
5 REM MEMDATA $1100
10 LN=100
20 FORI=0TO511STEP16:C=0:FORJ=ITOI+15:RE
   ADD:C=C+D:NEXT:READC1
30 IFC1<>CTHENPRINT"ERROR IN LINE:"LN:GO
   TO35
```

```
31 PRINT"{UP}"LN
35 LN=LN+10:NEXT
39 PRINT"{DOWN}CREATING DATA FILE"
40 OPEN1,8,2,"@0:DATA $1100,S,W":REM FOR
    CASS OPEN1,1,1,"DATA $1100"
41 RESTORE:FORI=0TO511STEP16:FORJ=ITOI+1
   5:READD:PRINT#1,CHR$(D);:NEXT
42 READD:NEXT:CLOSE1:END
100 DATA 16,207,16,175,48,195,16,0,16,20
    7,16,175,48,195,16,0,1346
110 DATA 16,207,16,175,16,195,16,0,16,20
    7,16,175,16,195,16,0,1282
120 DATA 16,207,16,175,48,195,48,0,0,36,
    215,12,219,12,215,12,1426
130 DATA 209,12,207,12,209,24,215,48,231
    ,24,228,48,225,48,219,36,1995
140 DATA 215,0,48,195,48,209,32,195,16,2
    09,48,219,32,195,16,209,1886
150 DATA 48,219,32,195,16,209,48,219,32,
    195,16,209,48,219,32,209,1946
160 DATA 16,219,48,225,32,219,16,209,48,
    195,48,195,48,209,48,0,1775
170 DATA 0,169,30,133,2,169,0,192,0,240,
    11,24,105,22,144,3,1244
180 DATA 230,2,24,136,208,246,133,1,164,
    0,96,72,56,165,140,101,1774
190 DATA 143,101,144,133,139,138,72,162,
    4,181,139,149,140,202,16,249,2112
200 DATA 165,139,41,15,133,139,104,170,1
    04,96,169,0,141,19,145,141,1721
210 DATA 34,145,173,32,145,41,128,133,0,
    173,17,145,41,60,5,0,1272
220 DATA 160,128,140,19,145,160,255,140,
    34,145,96,165,1,133,57,165,1943
230 DATA 2,24,105,120,133,58,96,162,0,18
    9,0,30,201,8,208,7,1343
240 DATA 169,0,157,0,30,240,11,201,6,240
    ,4,201,7,208,3,254,1731
250 DATA 0,30,189,0,31,201,8,208,7,169,0
    ,157,0,31,240,11,1282
260 DATA 201,6,240,4,201,7,208,3,254,0,3
    1,232,208,203,96,188,2082
```

```
270 DATA 128,26,169,117,192,3,208,2,169,
    80,248,162,2,24,117,61,1708
280 DATA 149,61,169,0,202,16,247,216,160
    ,3,162,0,181,61,32,134,1793
290 DATA 18,232,224,3,208,246,96,160,12,
    162,0,181,64,32,134,18,1790
300 DATA 232,224,3,208,246,96,165,14,160
    ,20,32,134,18,96,160,0,1808
310 DATA 169,15,141,14,144,88,169,0,133,
    162,177,1,240,30,170,200,1853
320 DATA 177,1,200,141,11,144,228,162,20
    8,252,132,57,32,170,17,164,2096
330 DATA 57,41,32,208,225,32,170,17,41,3
    2,240,249,169,0,141,11,1665
340 DATA 144,141,14,144,120,96,72,74,74,
    74,74,24,105,48,153,228,1585
350 DATA 31,200,104,41,15,24,105,48,153,
    228,31,200,96,32,110,19,1437
360 DATA 169,100,133,251,32,253,18,165,251,
    201,127,208,247,96,169,30,2450
370 DATA 133,2,169,0,133,1,224,240,240,6
    ,32,46,19,76,195,18,1534
380 DATA 32,78,19,165,1,24,105,22,133,1,
    144,234,224,240,240,45,1707
390 DATA 162,1,164,251,185,0,27,133,0,16
    9,31,133,2,169,206,133,1766
400 DATA 1,32,46,19,228,0,208,6,169,1,16
    0,1,145,1,165,1,1183
410 DATA 56,233,22,133,1,232,224,11,208,
    231,198,251,96,162,1,165,2224
```

Program 3-5. MEMDATA $1300

```
5 REM MEMDATA $1300
10 LN=100
20 FORI=0TO511STEP16:C=0:FORJ=ITOI+15:RE
   ADD:C=C+D:NEXT:READC1
30 IFC1<>CTHENPRINT"ERROR IN LINE:"LN:GO
   TO35
31 PRINT"{UP}"LN
35 LN=LN+10:NEXT
39 PRINT"{DOWN}CREATING DATA FILE"
```

```
40 OPEN1,8,2,"@0:DATA $1300,S,W"
41 RESTORE:FORI=0TO511STEP16:FORJ=ITOI+1
   5:READD:PRINT#1,CHR$(D);:NEXT
42 READD:NEXT:CLOSE1:END
100 DATA 251,24,105,21,168,185,0,27,133,
    0,169,31,133,2,169,206,1624
110 DATA 133,1,32,78,19,228,0,208,6,169,
    1,160,20,145,1,165,1366
120 DATA 1,56,233,22,133,1,232,224,11,20
    8,231,230,251,96,160,19,2108
130 DATA 32,203,17,177,1,72,177,57,200,1
    45,57,104,145,1,136,136,1660
140 DATA 192,0,208,239,200,169,0,145,1,1
    69,4,145,57,96,160,2,1787
150 DATA 32,203,17,177,1,72,177,57,136,1
    45,57,104,145,1,200,200,1724
160 DATA 192,21,208,239,136,169,0,145,1,
    169,4,145,57,96,160,0,1742
170 DATA 152,153,0,30,153,0,31,200,208,2
    47,169,4,153,0,150,153,1803
180 DATA 0,151,200,208,247,160,22,169,6,
    153,228,151,136,16,250,96,2193
190 DATA 162,0,32,139,17,165,139,240,5,1
    89,128,26,208,10,160,30,1650
200 DATA 72,104,136,208,251,76,105,20,18
    9,0,26,133,253,189,64,26,1852
210 DATA 133,254,189,128,26,201,3,208,20
    ,32,139,17,164,139,192,8,1853
220 DATA 176,22,32,139,17,164,139,192,8,
    176,11,144,4,201,1,208,1634
230 DATA 5,230,253,76,216,19,198,253,32,
    139,17,165,139,201,8,176,2127
240 DATA 28,32,139,17,165,139,201,8,176,
    11,165,254,201,1,240,13,1790
250 DATA 198,254,76,253,19,165,254,201,2
    0,176,2,230,254,189,0,26,2317
260 DATA 56,229,251,240,38,201,21,176,34
    ,133,0,188,64,26,32,113,1802
270 DATA 17,189,192,26,208,10,177,1,201,
    1,240,15,169,0,240,2,1688
280 DATA 169,1,145,1,32,203,17,169,4,145
    ,57,165,253,56,229,251,1897
```

```
290 DATA 240,40,201,21,176,36,133,0,164,
    254,32,113,17,177,1,201,1806
300 DATA 1,240,2,169,0,157,192,26,189,12
    8,26,72,24,105,20,145,1496
310 DATA 1,32,203,17,104,145,57,76,95,20
    ,169,0,157,192,26,165,1459
320 DATA 253,157,0,26,165,254,157,64,26,
    232,224,64,240,3,76,146,2087
330 DATA 19,96,165,251,24,101,20,133,0,1
    62,0,189,128,26,240,42,1596
340 DATA 189,64,26,197,252,208,35,189,0,
    26,197,0,208,28,138,72,1829
350 DATA 32,15,18,104,170,164,252,32,113
    ,17,169,6,164,20,145,1,1422
360 DATA 32,203,17,169,0,145,57,157,128,
    26,232,224,64,208,204,96,1962
370 DATA 162,63,169,0,157,128,26,202,16,
    250,165,15,201,41,208,4,1807
380 DATA 169,40,133,15,168,185,124,21,13
    3,17,162,0,32,4,21,169,1393
390 DATA 1,157,128,26,232,228,17,208,243
    ,185,84,21,24,101,17,133,1805
400 DATA 17,32,4,21,169,2,157,128,26,232
    ,228,17,208,243,185,44,1713
410 DATA 21,24,101,17,133,17,32,4,21,169
    ,3,157,128,26,232,228,1313
```

Program 3-6. MEMDATA $1500

```
5 REM MEMDATA $1500
10 LN=100
20 FORI=0TO511STEP16:C=0:FORJ=ITOI+15:RE
   ADD:C=C+D:NEXT:READC1
30 IFC1<>CTHENPRINT"ERROR IN LINE:"LN:GO
   TO35
31 PRINT"{UP}"LN
35 LN=LN+10:NEXT
39 PRINT"{DOWN}CREATING DATA FILE"
40 OPEN1,8,2,"@0:DATA $1500,S,W"
41 RESTORE:FORI=0TO511STEP16:FORJ=ITOI+1
   5:READD:PRINT#1,CHR$(D);:NEXT
42 READD:NEXT:CLOSE1:END
```

```
100 DATA 17,208,243,96,32,139,17,165,139
    ,10,10,10,10,133,0,32,1261
110 DATA 139,17,165,139,5,0,165,0,157,0,
    26,32,139,17,165,139,1305
120 DATA 240,249,41,7,157,64,26,169,0,15
    7,192,26,96,4,8,10,1446
130 DATA 12,12,12,8,12,12,12,8,12,12,8,1
    2,12,8,16,16,184
140 DATA 20,18,14,20,20,16,14,20,20,14,2
    0,20,16,20,16,16,284
150 DATA 24,22,30,17,17,3,4,6,10,6,2,8,8
    ,10,6,16,189
160 DATA 10,6,14,12,18,18,12,8,10,8,14,1
    2,14,16,14,16,202
170 DATA 8,24,18,22,20,20,22,18,20,26,17
    ,30,17,3,4,4,273
180 DATA 2,6,10,8,8,6,10,4,10,14,10,12,6
    ,10,12,16,144
190 DATA 10,8,16,12,14,16,20,16,24,14,18
    ,14,20,20,22,26,270
200 DATA 20,16,17,17,30,120,169,141,141,
    15,144,169,255,141,5,144,1544
210 DATA 32,157,18,169,0,160,5,153,61,0,
    136,16,250,160,9,32,1358
220 DATA 113,17,32,203,17,160,6,162,24,1
    38,232,145,1,169,6,145,1570
230 DATA 57,200,192,16,208,243,169,1,133
    ,15,32,108,25,169,0,133,1701
240 DATA 1,169,17,133,2,32,78,18,169,4,1
    33,14,169,0,133,61,1133
250 DATA 133,62,133,63,32,176,20,32,157,
    18,32,108,25,169,7,141,1308
260 DATA 14,144,160,10,32,113,17,32,203,
    17,169,2,160,10,145,1,1229
270 DATA 169,0,145,57,169,240,133,10,160
    ,33,185,242,24,153,228,31,1979
280 DATA 136,16,247,32,40,18,32,55,18,32
    ,70,18,169,0,133,19,1035
290 DATA 133,13,169,10,133,252,160,2,32,
    113,17,32,203,17,162,34,1482
300 DATA 160,8,138,232,145,1,169,6,145,5
    7,200,192,13,208,243,32,1949
```

```
310 DATA 170,17,41,32,208,249,160,2,32,1
    13,17,32,203,17,160,8,1461
320 DATA 169,0,145,1,169,4,145,57,200,19
    2,13,208,243,169,0,133,1848
330 DATA 16,165,13,41,64,240,12,169,191,
    37,13,133,13,32,215,17,1371
340 DATA 76,137,22,169,64,5,13,133,13,32
    ,8,25,32,33,25,162,949
350 DATA 255,165,252,133,9,32,170,17,74,
    74,74,176,8,164,252,192,2047
360 DATA 1,240,2,198,9,74,176,8,164,252,
    192,20,240,2,230,9,1817
370 DATA 74,176,6,164,251,240,2,162,15,7
    4,8,74,74,176,8,164,1668
380 DATA 251,192,233,240,2,162,240,164,2
    52,196,9,208,4,224,255,240,2872
390 DATA 65,32,113,17,160,10,165,19,145,
    1,32,203,17,160,10,169,1318
400 DATA 4,145,57,164,9,132,252,224,255,
    240,5,134,10,32,174,18,1855
410 DATA 164,252,32,113,17,160,10,177,1,
    201,20,176,43,133,19,169,1687
```

Program 3-7. MEMDATA $1700

```
5 REM MEMDATA $1700
10 LN=100
20 FORI=0TO511STEP16:C=0:FORJ=ITOI+15:RE
   ADD:C=C+D:NEXT:READC1
30 IFC1<>CTHENPRINT"ERROR IN LINE:"LN:GO
   TO35
31 PRINT"{UP}"LN
35 LN=LN+10:NEXT
39 PRINT"{DOWN}CREATING DATA FILE"
40 OPEN1,8,2,"@0:DATA $1700,S,W"
41 RESTORE:FORI=0TO511STEP16:FORJ=ITOI+1
   5:READD:PRINT#1,CHR$(D);:NEXT
42 READD:NEXT:CLOSE1:END
100 DATA 4,166,10,224,15,240,2,169,2,145
    ,1,32,203,17,169,0,1399
110 DATA 145,57,32,144,19,32,33,25,164,2
    52,32,113,17,160,10,177,1412
```

```
120 DATA 1,201,20,176,3,76,9,24,104,32,1
    08,25,169,160,141,13,1262
130 DATA 144,169,15,141,14,144,160,7,165
    ,252,24,121,127,25,153,68,1729
140 DATA 3,169,10,24,121,119,25,153,60,3
    ,136,16,235,164,252,32,1522
150 DATA 113,17,169,6,160,10,145,1,32,20
    3,17,169,0,145,57,162,1406
160 DATA 15,134,16,169,42,141,15,144,32,
    215,17,162,7,188,68,3,1368
170 DATA 32,113,17,188,60,3,169,0,145,1,
    189,60,3,240,45,201,1466
180 DATA 21,240,41,24,125,119,25,157,60,
    3,189,68,3,240,29,201,1545
190 DATA 21,240,25,24,125,127,25,157,68,
    3,168,32,113,17,188,60,1393
200 DATA 3,169,63,145,1,32,203,17,169,0,
    145,57,202,16,190,32,1444
210 DATA 135,25,169,141,141,15,144,32,13
    5,25,32,135,25,166,16,142,1478
220 DATA 14,144,202,16,156,248,165,14,56
    ,233,1,133,14,216,48,6,1666
230 DATA 32,70,18,76,247,21,169,66,133,1
    ,169,17,133,2,32,108,1294
240 DATA 25,32,78,18,248,162,0,181,61,56
    ,245,64,48,7,208,9,1442
250 DATA 232,224,3,208,242,216,76,189,21
    ,162,2,181,61,149,64,202,2232
260 DATA 16,249,216,32,55,18,76,189,21,4
    0,144,3,76,109,22,164,1430
270 DATA 16,240,3,76,113,22,169,160,141,
    13,144,160,1,132,16,164,1570
280 DATA 252,32,113,17,32,203,17,165,10,
    201,240,240,43,160,9,132,1866
290 DATA 20,177,1,153,60,3,240,16,201,1,
    240,12,32,114,20,169,1459
300 DATA 46,5,13,133,13,76,130,24,169,10
    ,145,1,169,0,145,57,1136
310 DATA 136,16,220,198,20,76,130,24,160
    ,11,132,20,177,1,153,60,1534
320 DATA 3,240,16,201,1,240,12,32,114,20
    ,169,46,5,13,133,13,1258
```

```
330 DATA 76,130,24,169,10,145,1,169,0,14
    5,57,200,192,22,208,218,1766
340 DATA 230,20,160,15,173,4,144,208,251
    ,136,208,248,164,252,32,113,2358
350 DATA 17,32,203,17,165,10,201,240,240
    ,22,160,9,196,20,240,36,1808
360 DATA 185,60,3,145,1,169,4,145,57,136
    ,196,20,208,242,240,20,1831
370 DATA 160,11,196,20,240,14,185,60,3,1
    45,1,169,4,145,57,200,1610
380 DATA 196,20,208,242,169,0,141,13,144
    ,162,63,189,128,26,208,31,1940
390 DATA 202,16,248,248,169,1,24,101,14,
    133,14,216,230,15,169,41,1841
400 DATA 133,1,169,17,133,2,32,108,25,32
    ,78,18,76,244,21,76,1165
410 DATA 113,22,11,12,13,48,48,48,48,48,
    48,0,14,15,48,48,584
```

Program 3-8. MEMDATA $1900

```
5 REM MEMDATA $1900
10 LN=100
20 FORI=0TO159STEP16:C=0:FORJ=ITOI+15:RE
   ADD:C=C+D:NEXT:READC1
30 IFC1<>CTHENPRINT"ERROR IN LINE:"LN:GO
   TO35
31 PRINT"{UP}"LN
35 LN=LN+10:NEXT
39 PRINT"{DOWN}CREATING DATA FILE"
40 OPEN1,8,2,"@0:DATA $1900,S,W"
41 RESTORE:FORI=0TO159STEP16:FORJ=ITOI+1
   5:READD:PRINT#1,CHR$(D);:NEXT
42 READD:NEXT:CLOSE1:END
100 DATA 48,48,48,48,0,2,48,48,160,0,132
    ,0,32,113,17,169,913
110 DATA 0,160,0,145,1,160,21,145,1,164,
    0,200,192,22,208,234,1653
120 DATA 96,165,13,41,32,208,36,165,13,4
    1,128,240,15,169,127,37,1526
130 DATA 13,133,13,169,150,141,11,144,14
    1,12,144,96,169,128,5,13,1482
```

```
140 DATA 133,13,169,0,141,11,144,141,12,
    144,96,198,13,169,150,141,1675
150 DATA 13,144,165,13,41,15,141,14,144,
    208,16,169,223,37,13,133,1489
160 DATA 13,169,0,141,13,144,169,7,141,1
    4,144,96,169,0,160,3,1383
170 DATA 153,10,144,136,16,250,96,0,255,
    1,0,1,255,255,1,1,1574
180 DATA 1,1,255,255,255,0,0,160,0,72,10
    4,72,104,136,208,249,1872
190 DATA 96,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,96
```

Program 3-9. LANDSCAPE DATA

```
100 POKE55,0:POKE56,27:CLR:D=6912:READA
101 IFA>9THENA=9
120 A=A+1:READB:FORI=1TOB:POKED,A:D=D+1:
    NEXT:READA:IFA>-1THEN101
130 DATA1,1,2,9,3,2,4,1,5,1,6,1,7,1,8,1,
    9,1,10,4,9,1,8,1,7,1,6,1
140 DATA5,11,4,1,3,7,4,9,5,1,6,1,7,4,8,2
    ,9,5,10,6,9,2,8,2,7,8
150 DATA6,3,5,6,6,4,5,6,6,3,7,1,8,3,7,4,
    6,5,5,5,4,19,5,6,6,20
160 DATA7,1,8,1,9,1,10,6,9,1,8,10,7,6,6,
    2,5,4,4,2,3,3,2,5,1,8,2,20,1,16,-1
```

Program 3-10. CHARACTERS

```
10 REM CHARACTERS2
15 FORI=7168TO7679:POKEI,0:NEXT
20 AD=7168
25 READA:IFA=-1THEN35
30 POKEAD,A:AD=AD+1:GOTO25
35 AD=7168+48*8
40 READA:IFA=-1THEN50
45 POKEAD,A:AD=AD+1:GOTO40
50 AD=7168+24*8
55 READA:IFA=-1THENFORI=0TO7:READA:POKE7
   672+I,255-A:NEXT:END
60 POKEAD,A:AD=AD+1:GOTO55
```

```
65 DATAØ,Ø,Ø,Ø,Ø,Ø,Ø,Ø:REM SPACE
7Ø DATAØ,254,254,254,254,254,254,254:REM
    BLOCK
75 DATAØ,192,168,189,199,1Ø1,56,Ø:REM 'U
   S' RIGHT
8Ø DATAØ,Ø,Ø,Ø,Ø,Ø,Ø,Ø:REM UNUSED
85 DATAØ,3,21,189,227,166,28,Ø:REM 'US'L
   EFT
9Ø DATAØ,Ø,Ø,Ø,Ø,Ø,Ø,Ø:REM UNUSED
95 DATAØ,Ø,24,6Ø,6Ø,24,Ø,Ø:REM EXP
1ØØ DATAØ,24,Ø,66,66,Ø,24,Ø
1Ø5 DATA6Ø,Ø,129,129,129,129,Ø,6Ø
11Ø DATAØ,Ø,Ø,Ø,Ø,Ø,Ø,Ø
115 DATAØ,Ø,Ø,9Ø,Ø,Ø,Ø,Ø :REM SHOT
12Ø DATAØ,238,17Ø,136,232,4Ø,17Ø,238:REM
     "SCORE:"
125 DATAØ,238,17Ø,17Ø,17Ø,172,17Ø,234
13Ø DATAØ,224,128,132,192,132,128,224
135 DATAØ,171,17Ø,17Ø,234,17Ø,17Ø,171:RE
    M "HIGH:"
14Ø DATAØ,168,17Ø,4Ø,56,17Ø,168,168
145 DATAØ,Ø,Ø,Ø,Ø,Ø,Ø,Ø
15Ø DATAØ,Ø,Ø,Ø,Ø,Ø,Ø,Ø
155 DATAØ,Ø,Ø,Ø,Ø,Ø,Ø,Ø
16Ø DATAØ,Ø,Ø,Ø,Ø,Ø,Ø,Ø
165 DATAØ,Ø,Ø,Ø,Ø,Ø,Ø,Ø
17Ø DATA192,16Ø,19Ø,131,186,126,2Ø,Ø
175 DATA3,5,125,193,93,126,4Ø,Ø
18Ø DATA6Ø,66,129,129,66,6Ø,36,24
185 DATA-1
19Ø DATAØ,6Ø,126,1Ø2,1Ø2,1Ø2,126,6Ø:REM
    START OF NUMBERS
195 DATAØ,28,6Ø,124,28,28,28,28
2ØØ DATAØ,6Ø,126,11Ø,14,6Ø,112,126
2Ø5 DATAØ,6Ø,126,7Ø,28,7Ø,126,6Ø
21Ø DATAØ,14,3Ø,54,126,126,6,6
215 DATAØ,126,126,96,124,126,14,124
22Ø DATAØ,28,62,96,124,98,126,6Ø
225 DATAØ,126,126,6,12,24,48,48
23Ø DATAØ,6Ø,126,1Ø2,6Ø,1Ø2,126,6Ø
235 DATAØ,6Ø,126,1Ø2,62,28,56,112
24Ø DATA-1
```

```
245 DATA235,170,170,170,202,170,170,171
    :REM RICHTHOFEN'S REVENGE
250 DATA171,169,41,41,57,41,169,169
255 DATA171,42,42,42,58,42,42,43
260 DATA187,162,162,162,179,162,162,163
265 DATA169,41,42,56,56,56,40,168
270 DATA112,80,64,64,112,16,80,112
275 DATA238,168,168,168,204,168,168,174
280 DATA174,168,168,168,172,168,168,78
285 DATA174,170,168,232,234,234,170,174
290 DATA236,140,140,140,204,128,140,236
295 DATA0,238,170,170,170,236,138,138:RE
    M PRESS FIRE
300 DATA0,238,136,136,206,130,130,238
305 DATA0,224,128,128,224,32,32,224
310 DATA0,235,138,138,202,139,138,138
315 DATA0,186,162,162,178,32,162,186
320 DATA-1
325 DATA40,84,66,146,64,170,84,0:REM EXP
    LOSION
```

Program 3-11. DATA LOADER (Disk Users Only)

```
20 A=4352:FORI=1TO4:READA$:OPEN1,8,2,"0:
   DATA $"+A$+"00,S,R":FORJ=0TO511
30 GET#1,B$:POKEA,ASC(B$+CHR$(0)):A=A+1:
   NEXT:CLOSE1:NEXT
40 DATA11,13,15,17
50 OPEN1,8,2,"0:DATA $1900,S,R":FORJ=0TO
   159:GET#1,B$:POKEA,ASC(B$+CHR$(0))
60 A=A+1:NEXT:CLOSE1
```

Program 3-12. DATA CLOADER (Cassette Users Only)

```
20 A=4352:FORI=1TO4:READA$:OPEN1,1,0,"DA
   TA $"+A$+"00":FORJ=0TO511
30 GET#1,B$:POKEA,ASC(B$+CHR$(0)):A=A+1:
   NEXT:CLOSE1:NEXT
40 DATA11,13,15,17
50 OPEN1,1,0,"DATA $1900":FORJ=0TO159:GE
   T#1,B$:POKEA,ASC(B$+CHR$(0))
60 A=A+1:NEXT:CLOSE1
```

Number of Enemy Aircraft at Each Skill Level of "Richthofen's Revenge"

Level	Enemy Aircraft				Accum. Score
	Right	Left	Balloon	Total	
1	4	4	4	12	650
2	4	4	8	16	1650
3	4	6	10	20	2900
4	2	10	12	24	4400
5	6	6	12	24	5900
6	10	2	12	24	7400
7	8	8	8	24	9000
8	8	8	12	28	10800
9	6	10	12	28	12600
10	10	6	12	28	14400
11	4	16	8	28	16300
12	10	10	12	32	18400
13	14	6	12	32	20500
14	10	14	8	32	22700
15	12	12	12	36	25100
16	6	18	12	36	27500
17	10	18	8	36	30000
18	12	12	16	40	32600
19	16	8	16	40	35200
20	10	10	20	40	37700
21	8	18	18	44	39800
22	16	14	14	44	42750
23	12	12	20	44	45550
24	14	14	20	48	48650
25	16	16	16	48	51850
26	20	14	14	48	55100
27	16	16	20	52	58500
28	24	8	20	52	61900
29	14	24	14	52	65450
30	18	18	20	56	69150
31	14	22	20	56	72850
32	20	20	16	56	76650
33	20	20	20	60	80650
34	22	22	16	60	84750
35	26	18	16	60	88850
36	20	20	24	64	93050
37	16	26	22	64	97300
38	17	17	30	64	101350
39	17	30	17	64	105725
40	30	17	17	64	110100

Chameleon

Clark and Kathryn H. Kidd

You need to capture flies, but where are you? Press the joystick button to find out. But don't become visible too often or you'll lose all your energy and will have to wait for another day to be seen again.

Life in the Desert

You are a desert chameleon. Your life consists of one thing: survival. The desert is cruel, and there isn't much food for chameleons. They eat flies all day long to stay alive.

Desert life is hard for other animals. Birds need to eat, too. They eat chameleons. To survive, you must eat all the flies you can. At the same time, you must avoid the birds that would get you. You must also stay away from thorny cactuses, whose needles are dangerous to chameleons.

As a chameleon, you have an advantage over the other animals in the desert. You can see the flies buzzing over the desert sands, but you are invisible to birds, since a chameleon's colors blend in with the desert.

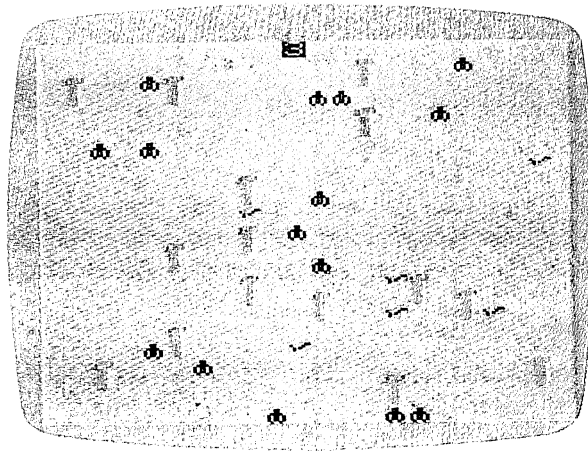
How to Play

As the day begins, the desert stretches before you. Cactuses grow here and there as far as the eye can see. Birds occasionally appear in the sky, looking for chameleons.

As a chameleon, you are invisible. To find out where you are, press the fire button on the joystick. You will remain visible as long as the fire button is pressed, but you cannot move while you are visible.

Becoming visible takes quite a bit of energy. You have the strength to become visible only a set number of times each day, so don't do it too often. If you exceed the number of times you can become visible in a day, you have to play invisibly until a new day begins and a new screen appears.

Use the joystick to move around the desert. You must eat all the flies you see, avoiding the thorny cactuses and the predatory birds. You receive points for each fly you catch, and lose points for each cactus you hit. If you encounter a bird,



You will have to avoid the birds and cactuses, which becomes difficult when you are invisible.

you disappear, and points are deducted from your score.

A game consists of three chameleons. If one chameleon is captured by a bird or if the chameleon eats all the flies in the desert, a new day begins and another screen will appear. Successive screens contain more birds and more flies, and the number of times the chameleon can become visible is reduced.

There are thousands of possible desert scenes in "Chameleon," so there is no limit to the score you can achieve. There will be flies for chameleons to eat as long as there are chameleons to eat them.

Program 3-13: Chameleon

```
50 POKE36879,29
60 PRINT"{CLR}{DOWN}{4 SPACES}* CHAMELEO
   N{2 SPACES}*"
70 GOSUB6000
80 T=PEEK(56)-2:POKE52,T:POKE56,T
100 M=7680:O=30720
120 READX:IFX<0THEN160
130 X=(X*8)+7168:FORY=0TO7
140 READZ:POKEX+Y,Z:NEXTY
150 GOTO120
```

```
160 FORX=0TO79
170 POKE(7248+X),255-PEEK(33152+X):NEXTX
200 GETX$:IFX$=""THEN200
210 PRINT"{CLR}{3 DOWN}{2 RIGHT}ENTER PL
    AY OPTION:":PRINT"{3 DOWN}{2 SPACES}
    {RVS}1{OFF} KEYBOARD{2 DOWN}"
220 PRINTSPC(5)"W{DOWN}"
230 PRINT"{3 SPACES}A ± S{2 SPACES}V=VIS
    IBLE{DOWN}"
240 PRINTSPC(5)"Z"
250 PRINT"{2 DOWN}{2 SPACES}{RVS}2{OFF}
    JOYSTICK/FIRE"
260 GETX$:IFX$=""THEN260
270 O%=VAL(X$):IFO%<1ORO%>2THEN260
300 GOSUB4000
305 C=3:L=0:S=0
310 L=L+1:GOSUB1000
400 ONO%GOTO410,430
410 IFPEEK(197)<>27THEN500
420 GOTO440
430 IF(PEEK(37137)AND32)THEN500
440 V=V-1:IFV<1THENX=3:Y=240:Z=10:GOSUB8
    00:GOTO500
450 POKEM+O+A,4:FORZ=1TO20:NEXTZ:GOTO400
500 POKEM+O+A,1:J=INT(A/22):K=A-(J*22)
510 ONO%GOTO520,570
520 X=PEEK(197):IFX=33THENJ=J+1
530 IFX=17THENK=K-1
540 IFX=9THENJ=J-1
550 IFX=41THENK=K+1
560 GOTO620
570 POKE37154,127:X=PEEK(37152):POKE3715
    4,255:Y=PEEK(37137)
580 IF(XAND128)=0THENK=K+1
590 IF(YAND16)=0THENK=K-1
600 IF(YAND4)=0THENJ=J-1
610 IF(YAND8)=0THENJ=J+1
615 IFJ<0THENJ=0
617 IFJ>22THENJ=22
620 IFK<0THENK=0
625 IFK>21THENK=21
630 B=(J*22)+K:IFA=BTHEN700
```

```
640 X=PEEK(M+B):ONXGOTO3100,3200,3300,31
    00,3100
650 IFX<>32THEN700
660 POKEM+A,32:POKEM+O+A,1:POKEM+B,4:POK
    EM+O+B,1:A=B
700 H=H+1:IFI>HTHEN400
710 X=INT(RND(1)*506):Y=PEEK(M+X):IFY<>3
    2THEN400
720 POKEM+X,3:POKEM+O+X,6:H=0:GOTO400
800 POKE36878,15:POKE36873+X,Y:FORW=1TOZ
    :NEXTW
810 POKE36878,0:POKE36873+X,0:RETURN
900 POKE36869,240:POKE198,0:PRINT"{CLR}
    {DOWN}{6 SPACES}{RVS}GAME OVER!{OFF}
    "
910 IFS<0THENS=0
920 PRINT"{2 DOWN}SCORE =";S
930 PRINT"{2 DOWN}REPLAY OPTION:":PRINT"
    {2 DOWN} {RVS}1{OFF} REPLAY/SAME LEV
    EL":PRINT"{DOWN} {RVS}2{OFF} REPLAY/
    NEW LEVEL"
940 PRINT"{DOWN} {RVS}3{OFF} END GAME
    {2 DOWN}":PRINT"ENTER OPTION"
950 GETX$:IFX$=""THEN950
960 X=VAL(X$):IFX<1ORX>3THEN960
970 ONXGOTO305,300
980 POKE52,T+2:POKE56,T+2:END
1000 POKE36869,255:PRINT"{CLR}"
1010 POKEM+10,C+10:POKEM+O+10,0
1020 F=(L*5)+(D*5):IFF>100THENF=100
1030 FORZ=1TOF:GOSUB2000:POKEM+X,2:POKEM
     +O+X,0:NEXTZ
1040 FORZ=1TO15
1045 GOSUB2000:IF(X-22)<0ORPEEK(M+X-22)<
     >32THEN1045
1047 POKEM+X,5:POKEM+X-22,1:POKEM+O+X,5:
     POKEM+O+X-22,5:NEXTZ
1050 GOSUB2000:A=X:POKEM+X,4:POKEM+O+X,1
1060 I=D*3
1070 FORZ=1TOI:GOSUB2000:POKEM+X,3:POKEM
     +O+X,6:NEXTZ
1080 H=0:V=150-(L*10)-(D*10):IFV<10THENV
     =10
```

```
1090 I=20-(L*2)-(D*2):IFI<3THENI=3
1100 RETURN
2000 X=INT(RND(1)*506)
2010 Y=PEEK(M+X)
2030 IFY<>32THEN2000
2040 RETURN
3100 POKEM+O+A,4:X=1:Y=200:Z=300:GOSUB80
     0:POKEM+O+A,1:S=S-5:GOTO700
3200 X=4:Y=220:Z=10:GOSUB800:S=S+1:F=F-1
     :IFF=0THEN310
3210 POKEM+A,32:POKEM+O+A,1:POKEM+B,4:PO
     KEM+O+B,1:A=B:GOTO700
3300 POKEM+O+A,4:X=1:Y=150:Z=600:GOSUB80
     0:C=C-1:IFC=0THEN900
3310 POKEM+O+A,1:POKEM+10,C+10:GOTO310
4000 PRINT"{CLR}{3 DOWN}{2 RIGHT}ENTER S
     KILL LEVEL:":PRINT"{3 DOWN}
     {2 SPACES}{RVS}1{OFF} BEGINNER
     {3 DOWN}":PRINT"{4 SPACES}TO
     {3 DOWN}"
4010 PRINT"{2 SPACES}{RVS}9{OFF} ADVANCE
     D"
4020 GETX$:IFX$=""THEN4020
4030 D=VAL(X$):IFD<1ORD>9THEN4020
4040 RETURN
6000 PRINT"{3 DOWN}EAT ALL THE FLIES BUT
      AVOID THE BIRDS AND{3 SPACES}THE C
     ACTUS."
6010 PRINT"{DOWN}+1 FOR EACH FLY
     {5 SPACES}{RIGHT} -5 FOR EACH CACTU
     S{5 SPACES}3 CHAMELEONS PER GAME";
6020 PRINT"{DOWN}THE FIRE/V KEY MAKES
     {2 SPACES}THE CHAMELEON VISIBLE.";
6030 PRINT"{3 DOWN}{3 SPACES}(PRESS ANY
     KEY)":RETURN
10000 DATA32,0,0,0,0,0,0,0,0
10010 DATA1,0,26,90,94,120,24,24,24
10020 DATA2,0,24,24,60,90,90,90,36
10030 DATA3,0,3,68,40,32,0,0,0
10040 DATA4,24,26,60,88,26,60,88,0
10050 DATA5,24,24,24,24,24,24,24,0,-1
```

Air Defense

T.L. Wahl

"Air Defense" is a challenging game for the Unexpanded VIC. For those with an 8K expander, a version has been upgraded to include a city skyline.

Your mission is to defend your city. A war has started, and the leaders of the nation have given you the responsibility of preventing the destruction of your city.

The enemy is dropping bombs. You can never tell just where the bombs will fall. You must line up the cross hairs of your gunsight and fire when the cross hairs and the bomb are aligned. You get only one shot.

If you decide to accept this mission, you must remember to press the S on your control panel (the keyboard) to move up, press X to move down, <cursor down> to move left, and <cursor right> to move right. Your firing button is the space bar.

You get only one shot at each bomb, and timing is critical. After 20 bombs have appeared, the attack (and game) is over, at which time you will be told how well you have done defending your city.

Increasing Difficulty

One of the unique features of this game is the increasing difficulty factor: as the player improves his skill, the cross hairs are gradually moved toward the top of the screen, and quicker reflexes and improved technique are required to destroy the falling bombs. As a reward for increasing skill, the player earns higher point value for successive hits. In addition, the player receives a higher score the sooner the falling bomb is destroyed.

Typing in the 8K Expander Version

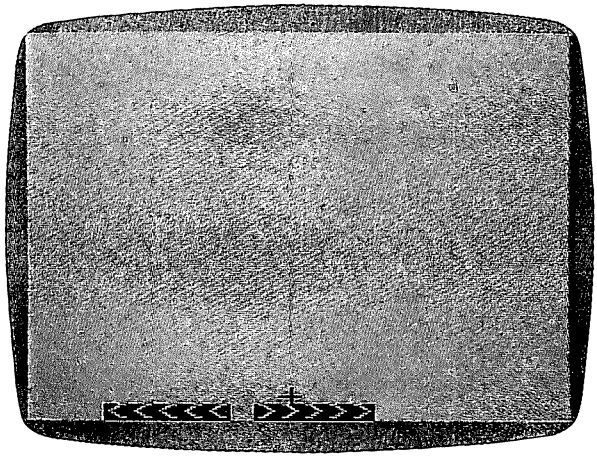
For those with an 8K expander, there is included a much enhanced version of "Air Defense" written by Richard Ruef. The 8K version includes the skyline of the city you are defending. It is important to follow these instructions before running the

program. First type in the unexpanded version (Program 3-14) with the additions and changes listed in Program 3-15. Once you have the program typed in, SAVE it. Do *not* try to RUN it first or you will lose your program. Once the program is SAVEd to disk or tape, type these lines in immediate mode.

POKE 648,30:SYS58648 <RETURN>

POKE 642,32:SYS58232 <RETURN>

Now you should reLOAD your program and RUN. You will have to use these two lines each time you wish to LOAD and RUN the 8K version of Air Defense.



"Air Defense" for the unexpanded VIC. The 8K expander version includes a skyline.

Program 3-14: Air Defense for the Unexpanded VIC

```
100 X=RND(0)
110 A=8152:B=38872:P=0:M=0:T=0:Q=0
120 PRINT"{CLR}{7 DOWN}{5 SPACES}AIR DEF
    ENSE"
130 PRINT"{2 DOWN}{5 SPACES}DO YOU NEED"
140 PRINT"{DOWN}{4 SPACES}INSTRUCTIONS?"
150 PRINT"{DOWN}{3 SPACES}TYPE 'Y' OR 'N
    '"
160 FOR H=1TO1000:GETD$
```

```
170 IF D$="N" THEN 380
180 IF D$="Y" THEN 220
190 NEXT
200 PRINT"{CLR}{DOWN}YOU DID NOT PRESS '
    Y' OR 'N'."
210 FOR K=1TO5000:NEXT:GOTO120
220 PRINT"{CLR}{2 SPACES}YOU MUST STOP T
    HE"
230 PRINT"{2 SPACES}FALLING BOMB BY"
240 PRINT"{3 SPACES}EXPLODING IT IN"
250 PRINT"{5 SPACES}MID-AIR."
260 PRINT"{DOWN} MOVE THE CROSSHAIR"
270 PRINT"{DOWN}*{RVS}LEFT{OFF}:CURSOR U
    /D KEY"
280 PRINT"{DOWN}*{RVS}RIGHT{OFF}:CURSOR
    L/R KEY"
290 PRINT"{DOWN}*{RVS}UP{OFF}:WITH THE '
    S' KEY"
300 PRINT"{DOWN}*{RVS}DOWN{OFF}:WITH THE
     'X' KEY"
310 PRINT"WHEN THE BOMB AND THE"
320 PRINT"CROSSHAIR ARE LINED UP, FIRE B
    Y PRESSING THE SPACE";
330 PRINT" BAR."
340 PRINT"{DOWN}PRESS ANY KEY TO START"
350 GET D$:IF D$="" THEN 350
360 PRINT"{CLR}{10 DOWN}{6 SPACES}GOOD L
    UCK!"
370 FOR I=1TO2500 :NEXT
380 IFT=20 THEN 860
390 PRINT"{CLR}":D=INT(RND(1)*10)
400 T=T+1
410 E=D+7685
420 F=D+38405
430 PRINTP*Q*10
440 FOR I=1 TO 200:NEXTI
450 POKE A,91:POKE B,0
460 GET A$
470 IFA$="S"THENA=A-22:B=B-22
480 IF A$="X"THEN B=B+22:A=A+22
490 IF A$="{RIGHT}"THEN A=A+1:B=B+1
500 IF A$="{DOWN}"THEN A=A-1:B=B-1
```

```
510 IF A<7680 THEN A=A+22:B=B+22
520 IF A>8163 THEN A=A-22:B=B-22
530 POKE E,42:POKE F,0
540 FOR I=1 TO 50:NEXT
550 IF E>8163 THEN GOTO 760
560 IF A=ETHEN 580
570 E=E+22:F=F+22:PRINT"{CLR}":GOTO450
580 GET B$
590 IFB$=" "THEN 620
600 GOTO 570
610 REM BOMB IS DESTROYED
620 X=100:FORI=1TO10:POKEE,X
630 POKE F,0
640 POKEE+21,X
650 POKEF+21,0
660 POKEE+24,X
670 POKEF+24,0
680 X=X+1
690 NEXT
700 NO=210:S1=-3:DU=60:GOSUB 960
710 P=P+1
720 Q=Q+22-INT((A-7680)/22)
730 A=A-22:B=B-22
740 GOTO380
750 REM BOMB GETS YOU!
760 POKE E,32:FOR I=1TO5
770 POKEE-I,188
780 POKEF-I,0
790 POKEE+I,190
800 POKEF+I,0
810 FOR S=1 TO 50:NEXT
820 NEXT
830 M=M+1
840 NO=135:S1=-2:DU=100:GOSUB 960
850 GOTO380
860 PRINT"{CLR}{DOWN}{6 SPACES}GAME OVER
    "
870 PRINT"{3 DOWN}DESTROYED"P
880 PRINT"{2 DOWN}MISSED"M
890 PRINT"{2 DOWN}TOTAL POINTS"P*Q*10
900 FOR I=1 TO 30:GET D$:NEXT I
910 PRINT"{4 DOWN}PRESS {RVS}P{OFF} TO P
    LAY AGAIN"
```

```
920 GET D$:IF D$="" THEN 920
930 IF D$="P" THEN 110
940 END
950 REM EXPLOSIONS
960 POKE 36877,NO
970 FOR I=15 TO 1 STEP S1
980 POKE 36878,I
990 FOR DELAY=1TODU:NEXTDELAY:NEXTI
1000 POKE 36877,0:POKE 36878,0
1010 RETURN
```

Program 3-15: Air Defense for a VIC with an 8K Expander

Additions and modifications to Program 3-14. Be sure to read the directions for using this version before you try to RUN the game.

```
20 GOTO100
40 L=30720:POKE650,128
50 FORW=8032TO8142STEP22:POKEW,162:POKEW
   +L,0:NEXT
55 FORW=8055TO8143STEP22:POKEW,162:POKEW
   +L,0:NEXT
56 FORW=8122TO8144STEP22:POKEW,160:POKEW
   +L,0:NEXT
57 FORW=8164TO8178:POKEW,232:POKEW+L,5:N
   EXT
58 FORW=8080TO8146STEP22:POKEW,220:POKEW
   +L,0:NEXT
59 FORW=8038TO8148STEP22:POKEW,220:POKEW
   +L,0:NEXT
60 FORW=8083TO8150STEP22:POKEW,220:POKEW
   +L,0:NEXT
61 FORW=8031TO8163STEP22:POKEW,136:POKEW
   +L,2:NEXT
62 FORW=8030TO8114STEP21:POKEW,78:POKEW+
   L,2:NEXT
63 FORW=8052TO8118STEP22:POKEW,66:POKEW+
   L,2:NEXT
64 FORW=8073TO8117STEP22:POKEW,66:POKEW+
   L,2:NEXT
65 FORW=8094TO8116STEP22:POKEW,66:POKEW+
```

```
   L,2:NEXT
66 FORW=8115TO8115STEP22:POKEW,66:POKEW+
   L,2:NEXT
67 FORW=8136TO8114STEP22:POKEW,66:POKEW+
   L,2:NEXT
68 FORW=8135TO8141:POKEW,152:POKEW+L,2:N
   EXT
69 FORW=8179TO8185:POKEW,102:POKEW+L,5:N
   EXT
70 POKE8160,98:POKE8160+L,0
71 FORW=8068TO8156STEP22:POKEW,162:POKEW
   +L,0:NEXT:POKE8046,223:POKE8046+L,0
72 POKE8045,233:POKE8045+L,0:FORW=8067TO
   8155STEP22:POKEW,220:POKEW+L,0:NEXT
73 POKE8023,103:POKE8023+L,0
75 POKE8145,111:POKE8145+L,0:POKE8147,24
   2:POKE8147+L,0:POKE8148,242:POKE8148+
   L,0
76 POKE8150,114:POKE8150+L,0:FORW=8107TO
   8151STEP22:POKEW,162:POKEW+L,0:NEXT
77 FORW=8108TO8152STEP22:POKEW,162:POKEW
   +L,0:NEXT:POKE8086,160:POKE8086+L,0
78 FORW=8109TO8153STEP22:POKEW,162:POKEW
   +L,0:NEXT:POKE8154,121:POKE8154+L,0
79 POKE8085,233:POKE8085+L,0:POKE8087,22
   3:POKE8087+L,0
80 POKE7984,81:POKE7984+L,7
81 POKE7988,100:POKE7988+L,6:POKE7989,11
   1:POKE7989+L,6
82 POKE7990,121:POKE7990+L,6:POKE7991,11
   1:POKE7991+L,6:POKE7992,100:POKE7992+
   L,6
83 POKE7944,98:POKE7944+L,6:POKE7945,121
   :POKE7945+L,6:POKE7946,111:POKE7946+L
   ,6
84 POKE7947,100:POKE7947+L,6:RETURN
425 GOSUB40
440 FOR I=1 TO 200:NEXT
445 G=A:H=B:Z=PEEK(A):J=PEEK(B)
530 POKEE,81:POKEF,0:IFE-22>=7680THENPOK
    EE-22,32
560 IFA=EANDPEEK(197)=32THEN620
```

```
565 IFA=GTHENE=E+22:F=F+22:GOTO46Ø
566 POKEG,Z:POKEH,J
57Ø E=E+22:F=F+22:PRINT"{HOME}":GOTO445
6ØØ REM DELETE THIS LINE
76Ø POKEE,81:POKEE+L,2:FORI=1TO7
765 POKE36879,127
77Ø POKEE-I,12Ø
78Ø POKEF-I,2
79Ø POKEE+I,12Ø
8ØØ POKEF+I,2
8Ø1 POKEE-(22*I),1Ø2:POKEE-(22*I)+L,2
81Ø FORS=1TO1Ø:NEXTS,I
811 Y=2:K=15:GOSUB821:Y=7:GOSUB821:Y=1:K
    =47:GOSUB821:Y=7:K=15:GOSUB821
812 Y=1:K=47:GOSUB821:Y=Ø:K=15:GOSUB821
813 FORI=1TO7:POKEE-(22*I),32:NEXT
814 FORI=1TO1ØØØ:NEXT:GOTO83Ø
82Ø REM DELETE THIS LINE
821 FORI=1TO3:POKE(E-132)+I,16Ø:POKE(E-1
    32)+I+L,Y
822 POKE(E-132)-I,16Ø:POKE(E-132)-I+L,Y:
    NEXT
824 FORI=1TO3:POKE(E-154)+I,16Ø:POKE(E-1
    54)+I+L,Y
825 POKE(E-154)-I,16Ø:POKE(E-154)-I+L,Y:
    NEXT
826 POKE36879,K:RETURN
1Ø1Ø POKE36879,27:RETURN
```

Part 4

Brain Testers

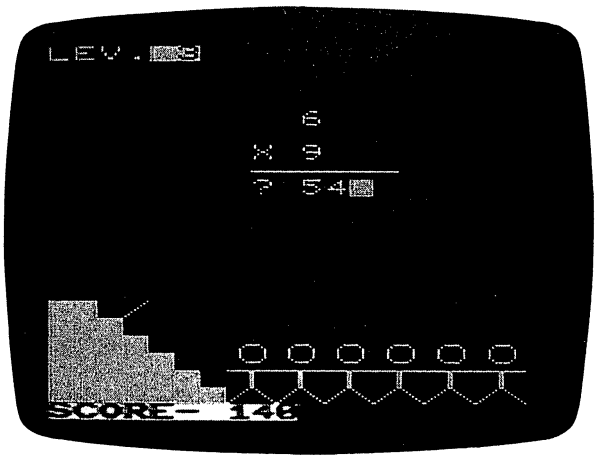
MathMan

Andy Hayes

Try this game to prove that math practice can be entertaining. "MathMan" uses animation to help teach children their multiplication facts.

Here's a program which proves that computer-aided math practice need not be boring. In the guise of a game, "MathMan" teaches multiplication facts by presenting random problems. The player (or student) types in the answer and presses RETURN. If he is correct, his friends gathered below cheer, but if the player fails to guess correctly, one of his friends will run away in shame. If all six friends flee, the game is over.

A good player can advance to the next level by successfully completing ten problems. The problems get successively more difficult, so this single program will provide a challenge for almost any elementary school child.



Multiplication practice is actually fun with "MathMan."

Program 4-1. MathMan

```
Ø A=6
1 LV=1
2 POKE198,Ø
1Ø PRINT"{CLR}{WHT}"
2Ø POKE36879,11Ø
3Ø CS$="{HOME}{21 DOWN}"
2ØØ PRINTLEFT$(CS$,16)"{RVS}{2 SPACES}
    {OFF} N"
22Ø PRINTLEFT$(CS$,17)"{RVS}{3 SPACES}
    {OFF}"
23Ø PRINTLEFT$(CS$,18)"{RVS}{4 SPACES}
    {OFF}"
24Ø PRINTLEFT$(CS$,19)"{RVS}{5 SPACES}
    {OFF}"
25Ø PRINTLEFT$(CS$,2Ø)"{RVS}{6 SPACES}
    {OFF}"
26Ø PRINTLEFT$(CS$,21)"{RVS}{7 SPACES}
    {OFF}"
27Ø IFA=6THENPRINT"{HOME}{17 DOWN}
    {7 RIGHT}UIUIUIUIUIUI"
271 IFA=6THENPRINT"{7 RIGHT}JKJKJKJKJKJK
    "
272 IFA=6THENPRINT"{7 RIGHT}POPOPOPOPOPO
    "
273 IFA=6THENPRINT"{7 RIGHT}NMNMNMNMNMNM
    "
274 IFA=5THENPRINT"{HOME}{17 DOWN}
    {7 RIGHT}{2 SPACES}UIUIUIUIUI"
275 IFA=5THENPRINT"{7 RIGHT}{2 SPACES}JK
    JKJKJKJK"
276 IFA=5THENPRINT"{7 RIGHT}{2 SPACES}PO
    POPOPOPO"
277 IFA=5THENPRINT"{7 RIGHT}{2 SPACES}NM
    NMNMNMNM"
278 IFA=4THENPRINT"{HOME}{17 DOWN}
    {7 RIGHT}{4 SPACES}UIUIUIUI"
279 IFA=4THENPRINT"{7 RIGHT}{4 SPACES}JK
    JKJKJK"
28Ø IFA=4THENPRINT"{7 RIGHT}{4 SPACES}PO
    POPOPO"
```

```
281 IFA=4THENPRINT"{7 RIGHT}{4 SPACES}NM
    NMNMNM"
282 IFA=3THENPRINT"{HOME}{17 DOWN}
    {7 RIGHT}{6 SPACES}UIUIUI"
283 IFA=3THENPRINT"{7 RIGHT}{6 SPACES}JK
    JKJK"
284 IFA=3THENPRINT"{7 RIGHT}{6 SPACES}PO
    POPO"
285 IFA=3THENPRINT"{7 RIGHT}{6 SPACES}NM
    NMNM"
286 IFA=2THENPRINT"{HOME}{17 DOWN}
    {7 RIGHT}{8 SPACES}UIUI"
287 IFA=2THENPRINT"{7 RIGHT}{8 SPACES}JK
    JK"
288 IFA=2THENPRINT"{7 RIGHT}{8 SPACES}PO
    PO"
289 IFA=2THENPRINT"{7 RIGHT}{8 SPACES}NM
    NM"
290 IFA=1THENPRINT"{HOME}{17 DOWN}
    {7 RIGHT}{10 SPACES}UI"
291 IFA=1THENPRINT"{7 RIGHT}{10 SPACES}J
    K"
292 IFA=1THENPRINT"{7 RIGHT}{10 SPACES}P
    O"
293 IFA=1THENPRINT"{7 RIGHT}{10 SPACES}N
    M"
294 IFA=0THENPRINT"{HOME}{17 DOWN}
    {7 RIGHT}{12 SPACES}"
295 IFA=0THENPRINT"{7 RIGHT}{12 SPACES}"
296 IFA=0THENPRINT"{7 RIGHT}{12 SPACES}"
297 IFA=0THENPRINT"{7 RIGHT}{12 SPACES}"
    :GOTO3000
299 PRINT"{HOME}LEV.{RVS}"LV
322 IFO=10THENLV=LV+1:GOTO2000
350 LETS=LV*2
355 O=O+1
360 B=INT(RND(1)*S)+1
370 C=INT(RND(1)*9)+1
375 PRINTLEFT$(CS$,23)"{RVS}{PUR}SCORE-"
    SC"{WHT}"
380 PRINTLEFT$(CS$,5)"{9 RIGHT}"B"{LEFT}
    {2 SPACES}"
```

```
390 IFB<10THENPRINTLEFT$(CS$,7)"
    {8 RIGHT}X"C"{LEFT}{2 SPACES}":GOTO4
    00
393 IFB<100THENPRINTLEFT$(CS$,7)"
    {8 RIGHT}X "C"{LEFT}{2 SPACES}":GOTO
    400
396 IFB<1000THENPRINTLEFT$(CS$,7)"
    {8 RIGHT}X{2 SPACES}"C"{LEFT}
    {2 SPACES}":GOTO400
400 PRINTLEFT$(CS$,8)"{8 RIGHT}
    ******"
410 PRINT"{HOME}{8 DOWN}{7 RIGHT}
    {9 SPACES}"
415 INPUT"{HOME}{8 DOWN}{8 RIGHT}";AS
430 IFAS=B*CTHEN700
440 IFAS<>B*CTHEN1000
700 SC=SC+5*LV
711 PRINTLEFT$(CS$,16)"{7 RIGHT}{RVS}THA
    NK YOU!!!{OFF}"
715 X=X+1
720 POKE36878,15
730 E=INT(RND(1)*30)+210
740 POKE36875,E
742 FORT=1TO100:NEXT:POKE36878,0
744 IFX=10THENX=0:GOTO760
750 GOTO715
760 FORT=1TO500:NEXT
770 PRINTLEFT$(CS$,23)"{12 RIGHT}
    {8 SPACES}"
772 PRINTLEFT$(CS$,10)"{20 SPACES}"
775 PRINTLEFT$(CS$,16)"{7 RIGHT}
    {14 SPACES}"
776 IFO=10THEN790
780 F=0:GOTO355
790 LV=LV+1:GOTO2000
1000 Q=7992
1005 POKEQ,32:Q=Q-21:POKEQ,78
1010 IFQ=7866THEN1030
1020 GOTO1005
1030 POKEQ,160:POKEQ+1,160:POKEQ-1,160:P
     OKEQ+22,160:POKEQ-22,160
1040 POKE36877,220
1041 Z=15
```

```
1042 Z=Z-1
1044 POKE36875,0:POKE36878,Z
1046 FORM=1TO100:NEXT
1048 IFZ=0THENZ=15:GOTO1060
1050 GOTO1042
1060 POKE36877,0:POKE36878,0:POKEQ+22,32
     :POKEQ-22,64:POKEQ,32:POKEQ-1,32:PO
     KEQ+1,32
1070 Q=7992
1075 POKEQ,32:Q=Q-21:POKEQ,78
1080 IFQ=7866THEN1200
1085 GOTO1075
1199 END
1200 Y=Y+1
1205 PRINT"{HOME}{8 DOWN}{9 RIGHT}{RVS}"
     B*C"{LEFT}{RVS} "
1210 FORT=1TO150:NEXT
1220 PRINT"{HOME}{8 DOWN}{9 RIGHT}{6 SPACES}"
1230 FORT=1TO150:NEXT
1235 IFY=8THENY=0:GOTO1300
1240 GOTO1200
1300 PRINTLEFT$(CS$,5)"{17 SPACES}"
1310 PRINTLEFT$(CS$,7)"{21 SPACES}"
1340 PRINTLEFT$(CS$,8)"{20 SPACES}"
1345 PRINTLEFT$(CS$,9)"{20 SPACES}"
1400 A=A-1
1430 IFO=10THEN2000
1500 POKEW,32:GOTO10
2000 FORT=1TO2000:NEXT
2001 PRINT"{CLR}{HOME}{6 DOWN} YOU MADE
     IT THROUGH"
2005 PRINT"{5 SPACES}LEVEL"LV-1
2010 PRINT"{2 DOWN}{2 SPACES}YOU NOW ADV
     ANCE TO"
2015 PRINT"{5 SPACES}LEVEL"LV:O=1:FORT=1
     TO4000:NEXT:GOTO10
3000 FORT=1TO2000:NEXT
3010 PRINT"{CLR}{HOME}{4 DOWN}SORRY BUT
     YOU LOST ALL";
3020 PRINT"{2 DOWN}{7 SPACES}YOUR MEN"
3030 PRINT"{4 DOWN}{4 SPACES}{RVS}YOUR S
     CORE WAS{OFF}"
3040 PRINT"{2 DOWN}{7 SPACES}"SC
```

Copy Cat

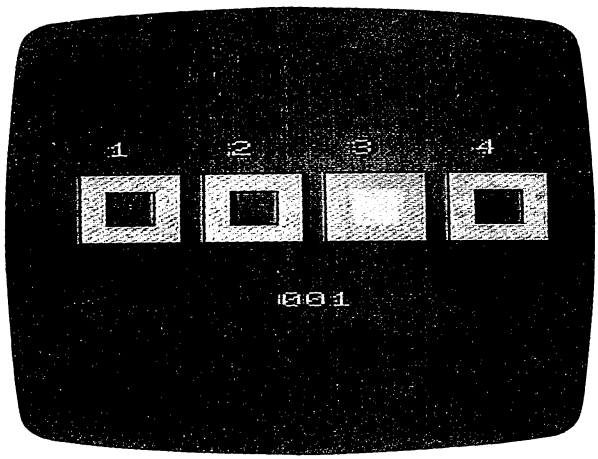
Mark and Dan Powell

How good is your memory? This memory game is easy at first and becomes progressively more difficult.

"Copy Cat" is an entertaining, musical, and colorful "match me" game. The object of this game is to duplicate the random pattern presented by the computer. Each time you correctly copy the pattern, you acquire a point.

In line 5 there is a REM in front of the POKEs. These POKEs disable the STOP key. Do not take off the REM until you've finished typing in the program (or don't put it in at all).

Note also that line 2020 reads IF PEEK (653) > 3THEN END. What this does is test for the VIC's CTRL key. To test for the SHIFT key, IF PEEK(653) =1THEN END, and to test for the Commodore key, it should read IF PEEK(653) =2THEN END. For combinations of these keys, just add them together (the value for CTRL is 4).



It takes only one mistake to end the game with "Copy Cat."

Program 4-2. Copy Cat

```
5 REM:POKE809,242:POKE808,199
10 POKE 36879,27
20 PRINT"{CLR}{4 DOWN}{BLK}{6 SPACES}COPY
    CAT":PRINT"{2 DOWN} PRESS 1-4 TO COP
   Y THE"
30 PRINT"{6 SPACES}COMPUTER":PRINT"
   {2 DOWN} YOU CAN ONLY MISS":PRINT" TH
   REE TIMES A GAME"
32 PRINT"{3 DOWN}{BLU}PRESS 'SPACE' TO ST
   ART"
35 GETA$:IFA$<>" "THEN35
40 POKE36879,8:C=38400:SC=256*PEEK(648):I
   FSC=4096THENC=37888
41 DIML%(100)
50 PRINT"{CLR}{6 DOWN}{WHT}{3 SPACES}1
   {4 SPACES}2{4 SPACES}3{4 SPACES}4
   {3 SPACES}{DOWN}{2 SPACES}{RVS}
   {4 SPACES}{OFF} {RVS}{4 SPACES}{OFF}
   {RVS}{4 SPACES}{OFF} {RVS}{4 SPACES}
   {OFF}"
55 FORT=1TO2
60 PRINT"{WHT}{2 SPACES}{RVS} {OFF}
   {2 SPACES}{RVS} {OFF} {RVS} {OFF}
   {2 SPACES}{RVS} {OFF} {RVS} {OFF}
   {2 SPACES}{RVS} {OFF} {RVS} {OFF}
   {2 SPACES}{2 RVS} {OFF}"
70 NEXTT
80 PRINT"{2 SPACES}{RVS}{4 SPACES}{OFF}
   {RVS}{4 SPACES}{OFF} {RVS}{4 SPACES}
   {OFF} {RVS}{4 SPACES}{OFF}"
82 PRINT"{HOME}{15 DOWN}"SPC(10)"000"
85 FORLA=0TO3
87 LC(LA)=INT(RND(1)*4)+2:IFLC(LA)=3THENL
   C(LA)=6
90 FORLB=1TO4:CN=LC(LA):IFLC(LA)=LC(LB)AN
   DLB<>LATHEN87
95 NEXT:POKEC+201+5*LA,CN:POKEC+202+5*LA,
   CN:POKEC+223+5*LA,CN:POKEC+224+5*LA,C
   N:NEXT
99 FORT=1TO300:NEXT
```

```
100 LF=LF+1:IFLF=100THEN2000
110 L%(LF)=INT(RND(1)*4)
120 FORLL=1TOLF:S=L%(LL):Q=160:GOSUB1000
130 FORT=1TO300:NEXT:Q=32:GOSUB1000:POKE3
    6878,0:FORT=1TO200:NEXT:NEXT
135 FORLG=1TOLF:TA=TI
140 GETA$:A=VAL(A$)-1:IFTI-TA>200THENS=L%
    (LG):GOTO160
150 S=A:IFA=-1ORA>3THEN140
152 LF$=STR$(LF)
160 Q=160:GOSUB1000:FORT=1TO200:NEXT:Q=32
    :GOSUB1000:POKE36878,0
162 IFA=L%(LG)THENFORT=1TO50:NEXT:NEXT
165 IFLG=LF+1THENPRINT"{HOME}{15 DOWN}
    {WHT}"TAB(14-LEN(LF$))RIGHT$(LF$,LEN
    (LF$)-1):GOTO99
170 PRINT"{HOME}{2 DOWN}"TAB(9)"{YEL}MISS
    ":POKE36878,15:POKE36875,128:R=R+1:F
    ORT=1TO400:GETA$:NEXT
175 IFR=3THENFORT=1TO100:NEXT:GOTO2000
180 FORT=1TO600:NEXT:PRINT"{HOME}{2 DOWN}
    {13 SPACES}":POKE36878,0:FORT=1TO500
    :NEXT
190 GOTO120
1000 POKESC+201+5*S,Q:POKESC+202+5*S,Q:PO
     KESC+223+5*S,Q:POKESC+224+5*S,Q
1010 POKE36878,15:POKE36875,7*S+217:RETUR
     N
2000 PRINT"{HOME}{16 DOWN}{WHT}{5 SPACES}
     *GAME OVER*":PRINT"{DOWN}{4 SPACES}
     TO PLAY AGAIN":POKE36878,0
2005 PRINT"{5 SPACES}PRESS SPACE":PRINT"
     {2 DOWN} PRESS 'CTRL' TO STOP"
2010 GETA$:IFA$=" "THENRUN40
2020 IFPEEK(653)>3THENEND
2030 GOTO2010
```

Outpost

Tim Parker

This strategy game will run on the VIC with any memory configuration.

The object of "Outpost" is to survive. You are placed in an immovable outpost, armed with torpedoes, main and secondary energy armaments, and a targeting computer. Your opponents come in three sizes, labelled SML (small), MDM (medium) and HVY (heavy). Their objective is to overrun you, or destroy you by knocking out your armaments, computer, or energy supply.

When RUN, the screen gives you quite a lot of information. Your status is displayed to the right center, where values for ENGY (energy), COMP (computer), MAIN (main armament), SECN (secondary armaments), TORP (torpedoes), and VP (victory points) are displayed. Energy is rated from 0 to 99. If the energy drops to zero, you lose the game. Computer efficiency and both main and secondary armaments are rated as a percentage of capability. Ninety-nine is maximum.

If the computer falls to zero, you have lost all defensive capabilities, and lose the game. If either or both armaments fall to zero, they cannot be fired until recharged by a supply ship. Five torpedoes are supplied at the beginning of the game. A maximum of nine can be stored in the outpost at any time. Victory points is your score. For each light enemy ship destroyed, one victory point is awarded; similarly, two for medium, and three for heavy ships.

The top of the screen shows the enemy. Up to four are active at a time. Each enemy ship is referenced by a number on the "radar screen" at center left. The index above gives the DIST (distance), PROB (hit probability), and ENGY (energy) of the enemy. The hit probability is a function of both enemy distance and your computer efficiency. If enemy energy falls to zero the enemy is destroyed, and victory points are awarded.

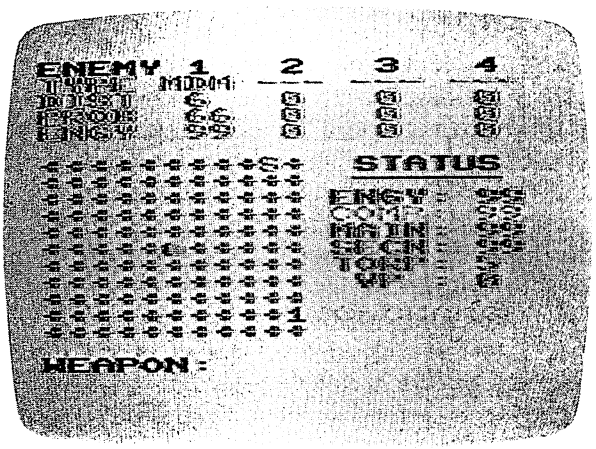
Playing Outpost

When playing, the computer will give you a "Weapon" prompt. This requires an input of T(orpedo), M(ain), or

S(econdary) for the different weapons. C can be entered to recharge your batteries, increasing the energy of the outpost to a maximum of 99. If a weapon is being fired, the prompt "TARGET NO" appears, requiring a value of one to four, depending on the enemy number.

After your turn, the computer will move some of the enemy ships, and some will fire at you. They have a hit probability that is a function of their energy. Damage to energy, computer, or armaments may result.

Occasionally, a supply ship wanders onto the screen. This is shown by a white "S." If it reaches you successfully, it recharges energy, main and secondary armaments to full power, and adds up to five torpedoes. Since a maximum of nine can be held at one time, any extras are lost. Note that the supply ship does not recharge your computer. The supply ships can be destroyed if an enemy lands on top of them.



"Outpost"

As might be expected, a hit on an enemy ship will decrease its energy. The amount of damage done is proportional to the type of enemy ship; the heavy ships are harder to destroy than mediums and lights. The type of weapon fired also affects damage. On an efficiency scale, torpedoes, main and secondary armaments are approximately 9:6:4 in damage ratios. A few trial games quickly give a feel for this.

High scores are not always easy to get. If a score of twenty is achieved, you are very good. Forty is excellent. Sixty is almost impossible, unless you're extremely lucky.

Strategy

The light ships are the most easily destroyed, but they do the least damage to you. If a heavy ship appears, try to get it fast. If an enemy gets within two moves of you, hit it hard. If it lands on you, you are destroyed. Also, protect your supply ships. They are needed and they are easily destroyed by the enemy.

As the computer efficiency rating drops, the hit probability also drops. With low computer values, you'll find that you have to wait for the enemy to get close before making shots. Torpedoes shouldn't be wasted, especially on low probability shots. If you get a few enemy ships on the screen at once, pick them off one at a time if possible to try to avoid concentrated fire. If you have four heavy ships bearing down on you, it's wisest to panic. If no enemy ships are on the screen, charge your batteries.

The Program

The program is divided into several blocks:

lines	**10-210**	Control section
	1000-1990	Screen display
	2000-2030	Refuel routine
	3000-3400	Enemy movement
	4000-4210	Enemy fire
	5000-5400	Enemy ship and supply ship appearances
	6000-6580	Weapons and firing routine
	9500-9630	Destroyed routine

When RUN, the program loops through each section, beginning with your fire routine, enemy fire and movement, new ships, and the screen routine. This is controlled by line 200.

The odds of a ship appearing are given in lines 5005 and 5010. The four ships are listed as subscripts of ET(x), with a value of one for light, two for medium, and three for heavy ships. ET(5) is the supply ship, and has a value of five if one is on the screen, and zero otherwise. EH(x) is the ship's hit probability, given by line 40, and ED(x) is the distance, given in line 30. To change the difficulty level of the game, change

the values of the number following "FNA(x)" in lines 5005 and 5010. If a ship is scheduled to appear, line 5110 chooses the type. Light ships are the most probable.

The "radar screen" is generated by section 1500-1720 using an individual coordinate system. EX(x) and EY(x) define each enemy ship location. This is a slow method for generating the display, but it saves the most memory.

Program 4-3. Outpost

```
10 PRINT"{CLR}":POKE36879,76
20 DEFFNA(X)=INT(RND(1)*X+1)
30 DEFFNB(Z)=INT(SQR((EX(G)-6)↑2+(EY
   (G)-6)↑2))
40 DEFFNC(Z)=INT(1/(ED(G))*100+(C/2))
50 C=99:G=1:GOSUB5110:GOSUB2000
200 GOSUB5000:GOSUB1000:GOSUB6000:GOSUB3
    000:GOSUB4000
210 GOTO200
1000 PRINT"{HOME}{DOWN}{BLK}ENEMY 1
     {3 SPACES}2{3 SPACES}3{3 SPACES}4"
1020 PRINT"{WHT}TYPE ";
1030 FORG=1TO4
1040 IFET(G)=0THENPRINT"--- ";
1050 IFET(G)=1THENPRINT"LGT ";
1060 IFET(G)=2THENPRINT"MDM ";
1070 IFET(G)=3THENPRINT"HVY ";
1080 NEXT
1100 PRINT:PRINT"DIST":PRINT"PROB
     {16 SPACES}":PRINT"ENGY{16 SPACES}"
1120 FORG=1TO4
1122 X=1+G*4
1124 PRINT"{4 UP}"
1126 PRINTSPC(X)ED(G)
1128 PRINTSPC(X)EH(G)
1130 PRINTSPC(X)EE(G)
1140 NEXTG
1300 PRINT:PRINTSPC(12);" {BLU}STATUS"
1305 PRINTSPC(12)" DDDD
     DD"
1310 PRINTSPC(11)" {CYN}ENGY:{3 SPACES}
     {3 LEFT}";E
```

```
1320 PRINTSPC(11)"{YEL} COMP:{3 SPACES}
     {3 LEFT}";C
1330 PRINTSPC(11)"{CYN} MAIN:{3 SPACES}
     {3 LEFT}";M
1340 PRINTSPC(11)" SECN:{3 SPACES}
     {3 LEFT}";S
1350 PRINTSPC(11)" TORP:";T
1360 PRINTSPC(11)"{2 SPACES}VP :";VP
1400 PRINT:PRINTSPC(11);" {RED}C=CHARGE"
     :PRINT:PRINT
1500 PRINT"{HOME}{6 DOWN}"
1510 A=0
1520 FORY=1TO11
1530 FORX=1TO11
1540 FORG=1TO5
1550 IFY<>EY(G)THEN1620
1560 IFX<>EX(G)THEN1620
1570 A=1:IFG=1THENPRINT"{BLK}1";
1580 IFG=2THENPRINT"{BLK}2";
1590 IFG=3THENPRINT"{BLK}3";
1600 IFG=4THENPRINT"{BLK}4";
1610 IFG=5THENPRINT"{WHT}S";
1620 NEXTG
1630 IFX=6ANDY=6THENPRINT"{GRN}Q";:A
     =1
1640 IFA=1THENA=0:GOTO1660
1650 PRINT"{GRN}+";
1660 NEXTX
1670 PRINT
1680 NEXTY
1690 PRINT"{HOME}{6 DOWN}"
1700 FORA=1TO11
1710 PRINTSPC(11)" "
1720 NEXT
1990 PRINT:RETURN
2000 ET(5)=0:EX(5)=0:EY(5)=0
2010 E=99:M=99:S=99
2020 T=T+5:IFT>9THENT=9
2030 RETURN
3000 FORG=1TO5:IFET(G)>0THEN3100
3010 NEXTG:RETURN
3100 IFG<5ANDFNA(9)>5THEN3010
```

```
3200 IF EX(G)>6THENEX(G)=EX(G)-1
3210 IF EX(G)<6THENEX(G)=EX(G)+1
3220 IFEY(G)<6THENEY(G)=EY(G)+1
3230 IFEY(G)>6THENEY(G)=EY(G)-1
3240 IFET(5)=5ANDEY(5)=6ANDEX(5)=6THENGO
     SUB2000
3250 IFEY(G)=6ANDEX(G)=6THEN9500
3265 IFG<5ANDEX(G)=EX(5)ANDEY(G)=EY(5)TH
     ENET(5)=0:EX(5)=0:EY(5)=0
3300 ED(G)=FNB(1)
3330 EH(G)=FNC(0):IFEH(G)>99THENEH(G)=99
3400 GOTO3010
4000 PRINT"{UP}ENEMY FIRING & MOVING"
4010 FORG=1TO4:IFET(G)<>0THEN4100
4020 NEXTG:RETURN
4100 IFFNA(99)>(EE(G)+FNA(30))OREE(G)<10
     THEN4020
4110 E=E-FNA(5)*ET(G)
4150 EE(G)=EE(G)-FNA(10)
4160 IFFNA(10)=1THENC=C-FNA(25):IFC<1THE
     N9500
4170 IFFNA(10)=1THENM=M-FNA(25):IFM<0THE
     NM=0
4180 IFFNA(10)=1THENS=S-FNA(25):IFS<0THE
     NS=0
4200 IFE<0THEN9500
4210 GOTO4020
5000 G=FNA(5)
5005 IFG=5ANDET(5)=0ANDFNA(4)>1THENET(5)
     =5:GOTO5160
5010 IFG=5ORET(G)<>0ORFNA(9)>4THEN5400
5110 A=4-INT(LOG(FNA(50)+2))
5120 ET(G)=A:EE(G)=99
5160 EX(G)=FNA(11)
5170 EY(G)=FNA(11)
5180 A=FNA(4):IFA=1THENEY(G)=1
5190 IFA=2THENEY(G)=11
5200 IFA=3THENEX(G)=11
5210 IFA=4THENEX(G)=1
5300 ED(G)=FNB(1)
5320 EH(G)=FNC(0):IFEH(G)>99THENEH(G)=99
5400 RETURN
```

```
6000 PRINT"{BLK}WEAPON:{14 SPACES}"
6010 GETA$:IFA$=""THEN6010
6020 IFA$="M"ANDM>0THENA=6:M=M-FNA(5):IF
     M<0THENM=0
6025 IFA$="C"THENE=E+FNA(20):IFE>99THENE
     =99
6030 IFA$="C"THENRETURN
6035 IFA$="S"ANDS>0THENA=4:S=S-FNA(5):IF
     S<0THENS=0
6040 IFA$="T"ANDT>0THENA=9:T=T-1
6060 IFA<3THENPRINT"{UP}BAD INPUT! WEAPO
     N:":GOTO6010
6100 PRINT"{UP}TARGET NO:{10 SPACES}"
6120 GET B$:IFB$=""THEN6120
6125 B=VAL(B$)
6130 IFET(B)=0THENPRINT"{UP}BAD DATA! TA
     RGET:":GOTO6120
6200 IFFNA(99)>EH(B)THENPRINT"{UP}MISSED
     !{7 SPACES}":FORZ=1TO1000:NEXT:RETU
     RN
6210 EE(B)=INT(EE(B)-((A*FNA(15))/ET(B)))
6215 PRINT"{UP}{WHT}TARGET HIT{6 SPACES}
     ":FORZ=1TO1000:NEXT
6220 IFEE(B)<1THEN6500
6230 E=E-FNA(5)
6300 RETURN
6500 VP=VP+ET(B)
6505 EX(B)=0:EY(B)=0
6510 ET(B)=0:EH(B)=0:ED(B)=0:EE(B)=0
6570 PRINT"{UP}{BLU}{RVS}{2 SPACES}TARGE
     T DESTROYED!{2 SPACES}"
6575 FORA=1TO1000:NEXT
6580 RETURN
9500 POKE36879,110
9510 PRINT"{CLR}{WHT}{3 DOWN}{4 SPACES}D
     ESTROYED!!!!!"
9550 PRINT"{4 DOWN}{5 SPACES}SCORE= ";VP
     :PRINT:PRINT
9560 IFVP>HSTHENHS=VP
9580 PRINT"{2 DOWN} {GRN} **************
     ***"
```

```
9590 PRINT" {GRN}{2 SPACES}HIGH SCORE= "
     ;HS
9600 PRINT" {GRN} ****************"
9605 PRINT"{4 DOWN}{2 SPACES}{WHT}
     {2 SPACES}ANOTHER GAME?"
9610 GETA$:IFA$=""THEN9610
9620 IFA$="Y"THENRUN
9630 STOP
```

Cryptic Numbers

C.G. McGaffin

This game of reasoning, logic, and luck will prove challenging to all age levels.

"Cryptic Numbers" is a game of logic and a little bit of luck for the Unexpanded VIC. Unlike the rapid-fire intergalactic space-war games, which demand flawless eye-hand coordination, Cryptic Numbers is a mentally challenging yet relaxed-pace game.

The object of the game is to determine the value of a hidden four-digit secret code number in as few tries as possible. The secret code number consists of the digits 0 through 9 in any order and combination including multiple repetitions, such as 2322.

The program randomly selects a new secret code number at the beginning of each game. Optional screen displays describe the play of the game and give an explanation of the clues which are given by the computer following each four-digit guess. The program compares each guess to the secret code number, and if the guess is incorrect it displays clues to aid the player in picking his next number.

Play continues until either the secret number is determined or ten incorrect guesses have been made. A maximum of ten guesses is allowed to preserve both the sanity of the player and a reasonable screen display. Upon completion of the game, the computer uncovers the X'd out secret code number and maintains the screen display to allow the player to study the input numbers and clues in comparison with the actual secret code number.

The Clues

Clues are given using the symbols ● (a black disk), o (a white disk), or a blank space. The computer will display one black disk for each digit in the player's guess which is identi-

cal to and in the same position as a digit in the secret code number. A white disk is displayed for each digit in the player's number which is the same as a digit in the secret code number but not in the correct position. A vacant space — that is, no symbol — results from each digit in the player's number which is not found in the secret code number.

For example, the number 4096, when compared to a secret code number of 1079, would result in the following clues: one black disk, one white disk, and two blank spaces. The black disk results from the 0 in the number being equal to and located in the same position as the 0 in the secret code number. The white disk results from the 9 in the number being equal to the 9 in the secret code, but not located in the correct position. The two blank spaces result from the digits 4 and 6 which are not found in the secret code number.

Black disks (if any) are displayed initially in each clue followed by white disks (if any) and then the blank spaces. The order of the symbols and spaces in the clues does not imply which of the digits in the player's number are correct or incorrect.

Entering Your Guess

Each input number must contain four digits and is terminated with a RETURN. Corrections may be made to the current input number by using the delete key (DEL) and retyping. The player should use all of the preceding guesses and their corresponding clues to arrive at subsequent guesses.

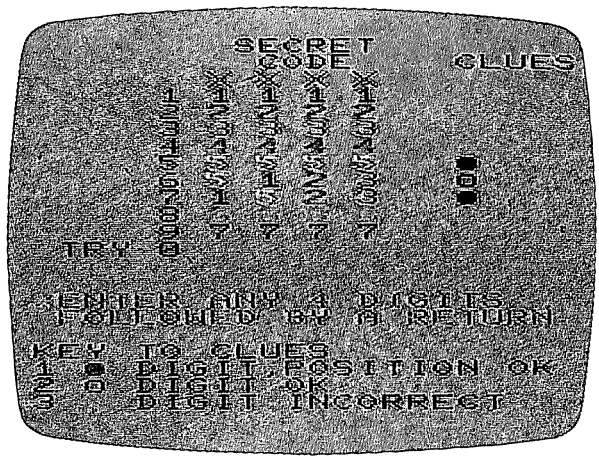
A Sample

The following screen display illustrates the play of the game:

	SECRET CODE	CLUES
	X X X X	
1	2 4 6 8	o
2	1 3 5 7	●o
3	9 2 3 7	
4	1 5 4 0	●oo
5	1 6 0 5	●●o
TRY 6		

Can you deduce what the hidden code is from the available information? One of the digits 2, 4, 6, or 8 is correct

based on the clue resulting from the first guess. The second guess resulted in clues which state that one digit is correct and in the right code number position and one other digit is correct but not correctly positioned. The third guess is very helpful since the digits 2, 3, 7, and 9 have been eliminated from further play. What would your guess be for TRY 6? The correct answer is given at the end of this article.



Only four more chances remain to figure out the secret code.

Crunching

The initial version of Cryptic Numbers was "crunched" (packed into memory) after it was operating to see how much the program's memory requirements could be reduced. The effort was well worth it. Since I now had more memory available, I made other revisions, adding some musical output, explanatory screen displays, and the ability to correct any of the digits being input in the current guess.

The program will run on a 5K or 8K system with no modification. Memory addresses found in lines 16, 51, 105, 106, and 108 must be changed to run the program on larger systems.

The Solution: For those of you who did not figure out the solution to the sample here, it is — 1085.

Program 4-4: Cryptic Numbers

```
1 DIMC(9),S(9),CG(3),AC(3),AB(3)
2 PRINT"{CLR}{3 DOWN}{3 SPACES}CRYPTIC-NU
  MBERS":PRINT"{2 DOWN}"
3 SC=36879:POKESC,25:PRINT"BREAK THE SECR
  ET FOUR"
4 PRINT"DIGIT CODE SET BY VIC.":PRINT:PRI
  NT"TYPE C FOR MORE ON HOWTO PLAY.":PRI
  NT
6 PRINT"TYPE RETURN TO START{2 SPACES}PLA
  Y.":PRINT:PRINT"{DOWN}{2 SPACES}*** GO
  OD LUCK! ***"
8 V=36878:S1=36876:GOSUB72:RESTORE
9 GETA$:IFA$=""THEN9
10 IFA$<>CHR$(13)ANDA$<>CHR$(67)THEN9
11 IFA$=CHR$(67)THENGOSUB82
12 PRINT"{CLR}{8 SPACES}SECRET{17 SPACES}
   CODE{4 SPACES}CLUES":PRINT" TRY 1"
13 FORJ=1TO14:PRINT:NEXT:PRINT"KEY TO CLU
   ES"
14 PRINT"1 Q DIGIT,POSITION OK 2 W DIGIT
   OK{10 SPACES}3{3 SPACES}DIGIT INCORRE
   CT":GOSUB106
15 FORJ=0TO3:AC(J)=INT(RND(1)*10):AB(J)=A
   C(J):NEXT
16 NG=0:NL=49:CS=7724:CC=38444:GS=7746:GC
   =38466:L=0
17 FORJ=0TO9:READC(J),S(J):NEXT
18 FORM=7TO13STEP2
19 POKECS+M,86:POKECC+M,2:NEXT
20 L=0:P1=7:P2=15:FORM=P1TOP2STEP2
21 GOSUB108:GETA$:IFA$=""THEN21
22 N=ASC(A$):IFN=20ORN=13THEN100
23 N=ASC(A$)-48:IFN<0ORN>9THEN21
24 IFM=15THEN21
25 CG(L)=N:POKEGS+M,S(N):POKEGC+M,C(N):L=
   L+1:NEXT
26 B=0:W=0:FORJ=0TO3:CG(J)=CG(J)+1:AC(J)=
   AC(J)+1:NEXT:FORJ=0TO3:IFCG(J)<>AC(J)
   THEN28
27 CG(J)=-CG(J):AC(J)=-AC(J):B=B+1
```

```
28 NEXT
29 FORJ=ØTO3:IFCG(J)<ØTHEN33
3Ø FORL=ØTO3:IFCG(J)<>AC(L)THEN32
31 CG(J)=-CG(J):AC(L)=-AC(L):W=W+1:GOTO33
32 NEXTL
33 NEXTJ
34 FORJ=ØTO3:AC(J)=AB(J):NEXT
36 GS=GS+17:GC=GC+17:IFB=ØTHEN38
37 FORJ=1TOB:POKEGS+J-1,81:POKEGC+J-1,Ø:N
   EXT
38 IFW=ØTHEN4Ø
39 FORJ=1TOW:POKEGS+B+J-1,87:POKEGC+B+J-1
   ,Ø:NEXT
4Ø GS=GS+5:GC=GC+5:IFB=4THEN51
41 FORJ=19TO21:POKEGS-J,32:POKEGC-J,1:NEX
   T:IFNG=9THEN45
42 POKEGS+1,2Ø:POKEGC+1,6:POKEGS+2,18:POK
   EGC+2,6:POKEGS+3,25:POKEGC+3,6
43 NL=NL+1:POKEGS+5,NL:POKEGC+5,6
44 IFNL=57THENNL=47
45 NG=NG+1:POKEV,15
48 FORJ=1TO3Ø:POKES1,183:NEXT
49 POKES1,Ø:POKEV,Ø
5Ø IFNG<1ØTHEN2Ø
51 MS=7724:MC=38444:L=Ø
52 FORM=7TO13STEP2:N=AC(L)+2
53 POKEMS+M,S(N-2):POKEMC+M,C(N-2):L=L+1:
   NEXT
54 IFNG<1ØTHENGOSUB72
56 M$="NICE JOB!{2 SPACES}TRY AGAIN!
   {SHIFT-SPACE}"
57 IFNG=1ØTHENM$="{3 SPACES}SORRY TRY AGA
   IN{4 SPACES}"
58 MS=MS+264:MC=MC+264
59 GOSUB7Ø
6Ø M$="{2 SPACES}TYPE AN E TO EXIT
   {3 SPACES}"
61 MS=MS+22:MC=MC+22
62 GOSUB7Ø
63 M$="OR C TO CONTINUE PLAY "
64 MS=MS+22:MC=MC+22
65 GOSUB7Ø
```

```
66 GETA$:IFA$=""THEN66
67 IFA$=CHR$(67)THEN2
68 IFA$<>CHR$(69)THEN66
69 GOTO79
70 FORJ=0TO21:C$=MID$(M$,J+1,1):N=ASC(C$)
   -64:IFN<0THENN=N+64
71 POKEMS+J,N:POKEMC+J,2:NEXT:RETURN
72 POKEV,15:FORL=250TO230STEP-2:POKES1,L
73 FORM=1TO110:NEXTM
74 NEXTL
75 FORL=232TO250STEP2:POKES1,L
76 FORM=1TO85:NEXTM
77 NEXTL:POKEV,0:POKES1,0:RETURN
78 DATA4,48,0,49,3,50,2,51,3,52,4,53,5,54
   ,6,55,7,56,2,57
79 POKESC,27
80 PRINT"{CLR}"
81 END
82 PRINT"{CLR}*EACH GUESS IS MADE UPOF FO
   UR DIGITS.{7 SPACES}":PRINT"*THE DIGI
   TS 0 TO 9 MAY";
83 PRINT"BE USED IN ANY COMBIN-ATION AND
   SEQUENCE.{3 SPACES}":PRINT"*DIGITS MA
   Y BE CORREC-";
84 PRINT"TED BY DELETING (DEL{2 SPACES}KE
   Y) AND RETYPING.":PRINT:PRINT"*A RETU
   RN MUST FOLLOW"
85 PRINT"EACH FOUR DIGIT GUESS.":PRINT"*V
   IC COMPARES THE CODEWITH EACH GUESS A
   ND"
86 PRINT"GIVES CLUES TO HELP{3 SPACES}YOU
    BREAK THE CODE."
87 PRINT:PRINT:PRINT"**TYPE C TO CONTINUE
   **"
90 GETA$:IFA$=""THEN90
91 IFA$<>CHR$(67)THEN90
92 PRINT"{CLR}*VIC PRINTS UP TO FOURSYMBO
   LIC CLUES AFTER{2 SPACES}EACH GUESS D
   EPENDING"
93 PRINT"ON THE NUMBER OF COR- RECT DIGIT
   S IN EACH{3 SPACES}GUESS.":PRINT
94 PRINT"Q IS PRINTED FOR EACH CORRECT DI
```

```
    GIT WHICH ISIN EXACTLY THE SAME"
95 PRINT"LOCATION IN THE CODE.":PRINT:PRI
    NT"W MEANS A DIGIT IS OK,BUT IN THE W
    RONG PLACE"
96 PRINT"(A VACANT CLUE MEANS ADIGIT IS I
    NCORRECT.)":PRINT"{3 DOWN}*HIT RETURN
     TO BEGIN*"
97 GETA$:IFA$=""THEN97
98 IFA$<>CHR$(13)THEN97
99 RETURN
100 IFN=13THEN104
101 IFM=7THEN21
102 M=M-2:L=L-1:IFL<0THENL=0
103 POKEGS+M,32:POKEGC+M,25
104 IFM<=13THEN21
105 POKE38730,1:GOTO26
106 MS=8010:MC=38730:M$="*ENTER ANY 4 DIG
     ITS{3 SPACES}":GOSUB70
107 MS=MS+22:MC=MC+22:M$=" FOLLOWED BY A
     RETURN ":GOSUB70:RETURN
108 MC=38730:POKEMC,1:FORI=1TO100:NEXT:PO
     KEMC,2:FORI=1TO100:NEXT:RETURN
```

Word Hunt

Eric Jansing and Bob Meyers, Jr.

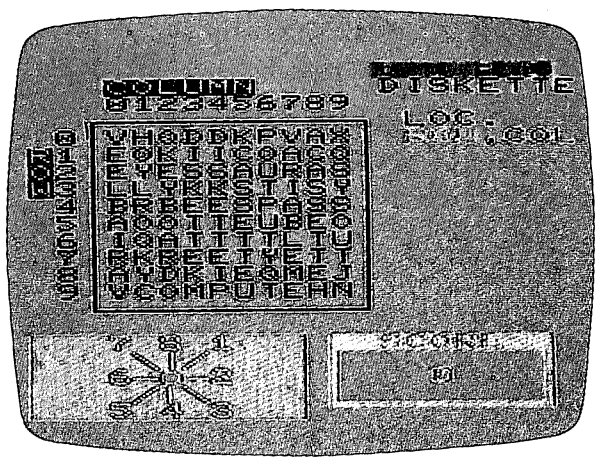
This VIC-20 translation of a popular Commodore PET program written by Robert W. Baker shows how screen compatibility problems can be overcome. It requires an 8K memory expansion.

"Word Hunt" is a great game that appeared in the March 1982 issue of *COMPUTE!* Magazine. The game was written perfectly except for one thing — it couldn't be RUN on a VIC without some problems. I put the game on the PET and liked it so much that I decided to translate it for the VIC.

After days of problems, I was ready to give up completely until I met Bob Meyers. He too had a VIC. I told him about the game and he agreed that it would be a good project. Bob's solution was to use a VIC 8K expander.

Screen Compatibility

The expander gave us enough memory, but it didn't solve the screen compatibility problem. The original program was written for a 40-column screen; VIC's screen is 22 columns wide.



Hunting for the word "flypaper." The next entry would be "6" to indicate the word is spelled backwards.

In the end, we took out a few words and abbreviated some others to make the program compatible with the VIC's screen.

When you RUN Word Hunt, you get a catchy title with lots of color. The computer then asks you the skill level, 1 being easy and 5 being hard. Then the computer asks you to input ten words. When the computer finishes the puzzle, you're asked to press any key and the game begins.

The computer writes the puzzle on the screen and displays the word you must hunt for. Then you are asked the starting location of the word (ROW,COLUMN), and the computer will ask you the direction. The direction box is located at the bottom left of the screen. To answer the direction, just look at the word and match its direction at the bottom.

If you are correct, the computer will respond "yes" and give you points. The number of points you get is determined by the time you took to answer.

Program 4-5: Word Hunt

```
2 CLR
3 A=Ø
4 POKE36879,8
5 A=A+1
1Ø PRINT"{CLR}":PRINT"{7 DOWN}{4 RIGHT}
   {DOWN}{RVS}QQQQQQQQQQQQ"
11 PRINT"{RVS}{4 RIGHT}Q{11 SPACES}Q"
12 PRINT"{RVS}{4 RIGHT}Q WORD HUNT Q"
13 PRINT"{RVS}{4 RIGHT}Q{11 SPACES}Q"
14 PRINT"{RVS}{4 RIGHT}QQQQQQQQQQQQQ
   {OFF}"
2Ø FOR X=1TO2ØØ:NEXT:PRINT"{CLR}"
25 L$="{WHT}{YEL}{GRN}{PUR}{RED}{CYN}":P
   RINT MID$(L$,A,1):IF A<7 THEN 5
7Ø FOR X=1 TO VAL(RIGHT$(TI$,2)):R=RND(1
   ):NEXT
8Ø S=1Ø:W=1Ø:DIM M(S,S),W$(W),P(S,S),L(W
   ,3),F(8)
9Ø POKE36879,253:PRINT"{CLR}{DOWN}{BLU}W
   HAT SKILL LEVEL"
95 PRINT:PRINT:PRINT
1ØØ PRINT"{PUR}1(EASY) TO 5(HARD)
    {3 SPACES}3{3 LEFT}";
```

```
110 INPUTR$:X=VAL(R$):IF X<1 AND X>5 THE
    N 90
120 SL=6-X
130 PRINT"{BLK}{2 DOWN}ENTER"W"WORDS,"
140 PRINT:PRINT"MAKE EACH WORD 3 TO 8"
150 PRINT:PRINT"CHARACTERS LONG."
170 FOR X=1TOW:L(X,1)=0:L(X,2)=0:L(X,3)=
    0
180 PRINT:PRINT"{RED}WORD";X;TAB(8);"
    {2 RIGHT}?{3 LEFT}";
190 INPUT R$:Q=LEN(R$)
200 IF Q<3 THEN PRINTTAB(26);"{RVS}{BLU}
    {UP}* TOO SHORT *{OFF}":GOTO 180
210 IF Q>8 THEN PRINTTAB(26);"{RVS}{PUR}
    {UP}* TOO LONG *{OFF}":GOTO 180
220 X9=0:FOR Y=1TOQ:A=ASC(MID$("*"+R$+"*
    ",Y+1,1))
230 IF A<65 OR A>90 THEN X9=1:Y=Q
240 NEXT Y:IF X9=1 THEN PRINTTAB(26)"
    {UP}* BAD WORD *":GOTO 180
250 IF X=11 THEN W$(X)R$+"*":GOTO290
260 X9=0:FORY=1TOX:IFQ<=LEN(W$(Y))-1 THE
    N 280
270 FOR B=XTOY+1STEP-1:W$(B)=W$(B-1):NEX
    T:W$(Y)=R$+"*":X9=1:Y=X-1
280 NEXT
290 NEXT
295 POKE36879,194
300 PRINT"{BLU}{CLR}{4 DOWN}{23 SPACES}T
    HAT'S ENOUGH WORDS!{23 SPACES}"
310 PRINT"{PUR}{4 DOWN}{23 SPACES}PLEASE
     BE PATIENT...{23 SPACES}"
320 PRINT"{BLK}{2 DOWN}{22 SPACES}I'M MA
    KING THE PUZZLE!{22 SPACES}"
340 FOR X=1TOS:FORY=1TOS:M(Y,X)=42:NEXT:
    NEXT:Q=0
360 FOR X=1 TO S:FORY=1TOS:P(Y,X)=0:NEXT
370 NEXT:Q=Q+1:IF Q>W THEN 760
380 G=LEN(W$(Q))-2
400 X9=0:FORX=1TOS:FORY=1TOS:IF P(Y,X)=0
    THENX9=1:X=S:Y=S
410 NEXT:NEXT:IF X9=1 THEN 450
```

```
430 PRINT"{CLR}THIS LIST OF WORDS
    {4 SPACES}WILL NOT ALL FIT
440 PRINT"PLEASE ENTER NEW WORDS":GOTO13
    0
450 A=INT(S*RND(1)+1):B=INT(S*RND(1)+1):
    IF P(B,A)<>0 THEN 450
460 P(B,A)=1:IF M(B,A)=42 THEN 490
470 IF M(B,A)<>ASC(LEFT$(W$(Q),1))THEN40
    0
490 FOR X=1TO8:F(X)=0:NEXT
500 X9=0:FOR X=1TO8:IF F(X)=0 THEN X9=1:
    X=8
510 NEXT:IF X9=0THEN400
520 D=INT(8*RND(1)+1):IF F(D)=1 THEN 520
530 F(D)=1:ON D GOTO 550,590,580,620,610
    ,650,640,560
550 IF (A+G)>S THEN 500
560 IF (B-G)<1 THEN 500
570 GOTO 670
580 IF (B+G)>S THEN 500
590 IF (A+G)>S THEN 500
600 GOTO 670
610 IF (A-G)<1 THEN 500
620 IF (B+G)>S THEN 500
630 GOTO 670
640 IF (B-G)<1 THEN 500
650 IF (A-G)<1 THEN 500
670 X=A:Y=B:X9=0:FORN=2TOG+1:GOSUB1550:I
    F M(Y,X)=42 THEN 690
680 IF M(Y,X)<>ASC(MID$(W$(Q),N,1)) THEN
     X9=1:N=G+1
690 NEXT:X=A:Y=B:IF X9=1 THEN500
710 FOR N=1TOG+1:IF M(Y,X)=42 THEN M(Y,X
    )=ASC(MID$(W$(Q),N,1))
720 GOSUB 1550:NEXT
740 L(Q,1)=A-1:L(Q,2)=B-1:L(Q,3)=D:IF Q<
    W THEN360
760 FOR Y=1TOS:FORX=1TOS:IFM(Y,X)=42 THE
    NM(Y,X)=INT(25*RND(1)+65)
770 NEXT:NEXT:WP=0:TS=0
775 POKE36879,15
780 PRINT"{CYN}{CLR}{5 DOWN}{RVS}READY"
```

```
790 PRINT"{GRN}{5 DOWN}PRESS ANY KEY TO
    PLAY"
800 R$="":GETR$:IF R$="" THEN 800
810 POKE36879,25
820 PRINT"{BLU}{CLR}{DOWN} {DOWN}
    {2 SPACES}{RVS}COLUMN";TAB(14);"
    {CYN}{RVS}{UP}W O R D"
860 PRINT"{BLU}{4 DOWN}{RVS}R{DOWN}
    {LEFT}O{DOWN}{LEFT}W{5 UP}{2 LEFT}
    {OFF}";
861 PRINT"{BLK}{4 RIGHT}";
870 FORX=0TOS-1:PRINTRIGHT$(STR$(X),1);:
    NEXTX:PRINT:Y=1:GOSUB1650
880 FORY=1TOS:PRINT"{RIGHT}";RIGHT$(STR$
    (Y-1),1);"-";
890 FORX=1TOS:PRINTCHR$(M(Y,X));:NEXTX
900 PRINT"-":NEXTY:Y=0:GOSUB1650
910 PRINT"{RED}{RVS}{DOWN}{3 SPACES}7 8
    1{3 SPACES}"
920 PRINT"{RVS}{4 SPACES}M-N{4 SPACES}":
    PRINT"{RVS}{3 SPACES}6*Q*2
    {3 SPACES}"
921 PRINT"{RVS}{4 SPACES}NBM{4 SPACES}":
    PRINT"{RVS}{3 SPACES}5 4 3{3 SPACES}
    "
930 G=17:GOSUB1700:PRINT:PRINT"{UP}"TAB(
    12);:PRINT"{PUR}{RVS}{2 SPACES}SCORE
    {2 SPACES}":PRINTTAB(12);"{J}
    {7 SPACES}{L}"
940 PRINTTAB(12);"{J}{3 SPACES}0
    {3 SPACES}{L}"
950 PRINTTAB(12);"{J}{7 SPACES}{L}"
951 PRINTTAB(12);"{9 U}":PRINT"{HOME}"
960 G=2:GOSUB1700:PRINT"{9 SPACES}"
970 WP=WP+1:IFWP>WTHEN1450
980 Q=LEN(W$(WP))-1
1000 GOSUB1700:PRINTTAB(15-(Q/2));LEFT$(
     W$(WP),Q):TI$="000000"
1020 G=4:GOSUB1700:PRINTTAB(15);"{BLU}LO
     C."
1025 PRINTTAB(15)"{GRN}ROW{BLK},{RED}COL
     "
```

```
1030 FORG=6TO11:GOSUB1700:
1040 PRINT"{5 SPACES}":NEXTG:G=6:GOSUB1700
1050 B$="":GETB$:IFB$=""THEN1050
1060 IF ASC(B$)=13THEN1050
1070 PRINTB$;",";:IFB$="0"THENB=0:GOTO10
     90
1080 B=VAL(B$):IFB<1ORB>9THENPRINT"
     {2 LEFT}{2 SPACES}{2 LEFT}";:GOTO10
     50
1090 A$="":GETA$:IFA$=""THEN1090
1100 IF ASC(A$)=13THEN1090
1110 PRINTA$:IFA$="0"THENA=0:GOTO1140
1120 A=VAL(A$):IFA<1ORA>9THEN1030
1140 G=7:GOSUB1700:PRINT"DIR:":PRINT:PRI
     NTTAB(15);" {LEFT}";
1150 GETD$:IFD$=""THEN1150
1160 IF ASC(D$)=13THEN1150
1170 PRINT"{UP}{RIGHT}";D$:D=VAL(D$):IFD
     <1ORD>8THEN1140
1190 WT=TI:IFB<>L(WP,2)THEN1230
1200 IF A<>L(WP,1)THEN1230
1210 IF D=L(WP,3)THEN1360
1230 X=A+1:Y=B+1:G=LEN(W$(WP))-1:IFM(Y,X
     )<>ASC(LEFT$(W$(WP),1))THEN1300
1240 X9=0:FORN=2TOG:GOSUB1550:IF X<1ORX>
     10THEN1270
1250 IF Y<1 OR Y>10 THEN1270
1260 IF M(Y,X)=ASC(MID$(W$(WP),N,1))THEN
     1280
1270 X9=1:N=G
1280 NEXTN:IF X9=0THEN 1360
1300 G=6:GOSUB 1700:PRINTSPC(0);:B$=STR$
     (L(WP,2)):A$=STR$(L(WP,1))
1310 PRINTRIGHT$(B$,LEN(B$)-1);",";RIGHT
     $(A$,LEN(A$)-1)
1320 G=8:GOSUB1700:PRINT SPC(1);L(WP,3)
1330 G=10:GOSUB 1700:PRINT"↑"
1340 G=11:GOSUB1700:PRINT"J {RVS} NO
     {OFF}"
1341 G=13:GOSUB1700:PRINT"{DOWN}HIT ANY"
     :G=13:GOSUB1700:PRINT"{2 DOWN}
     {2 SPACES}KEY"
```

```
1342 QW$="":GETQW$:IFQW$=""THEN1342
1343 G=10:GOSUB1700:PRINT" "
1344 G=11:GOSUB1700:PRINT"{5 SPACES}"
1345 G=13:GOSUB1700:PRINT"{DOWN}
     {7 SPACES}":G=13:GOSUB1700:PRINT"
     {2 DOWN}{5 SPACES}"
1350 GOTO 1420
1360 IF WT<(SL*60)THENWS=100:GOTO1390
1370 IF WT<(SL*1200)THENWS=10:GOTO1390
1380 WS=5+INT((SL*1200)-WT)/60)
1390 G=10:GOSUB1700:PRINT"↑"
1400 G=11:GOSUB1700:PRINT"{RVS}Y{OFF},
     {LEFT}"WS:TS=TS+WS
1420 G=17+2:GOSUB1700:PRINT TS
1430 GOTO 960
1450 PRINT"{HOME}{15 DOWN}"
1460 FORX=1TO6:PRINT"{12 SPACES}":NEXTX
1470 FORG=-2TO14:GOSUB1700
1480 PRINT"{22 SPACES}":NEXTG
1490 FORX=1TO1500:NEXTX:PRINT"{CLR}"
1491 POKE36879,76
1492 PRINT"{HOME}{8 DOWN}{YEL}DO YOU WIS
     H TO PLAY{3 SPACES}{DOWN}ANOTHER GA
     ME? IF YOU{2 SPACES}{DOWN}DO ENTER
     Y FOR YES."
1493 PRINT"{DOWN}IF YOU DON'T ENTER N
     {2 SPACES}{DOWN}FOR NO.
1500 R$="":GETR$:IFR$=""THEN1500
1505 IFR$="N"THEN1520
1510 IF R$="Y"THEN90
1515 IFR$<>"N"ANDR$<>"Y"THEN1500
1520 PRINT"{CLR}":POKE36879,42
1525 PRINT"{HOME}{7 DOWN}{CYN}THANK YOU
     FOR PLAYING {DOWN}{YEL}WORD HUNT
     {CYN}. HOPE YOU{3 SPACES}{DOWN}HAD FUN.
1530 PRINT"{2 DOWN}{6 RIGHT}{GRN}SEE YOU
      LATER!!!"
1535 FORX=1TO5000:NEXTX:PRINT"{CLR}":POK
     E36879,110
1540 PRINT"{HOME}{10 DOWN}{CYN}
     {4 SPACES}END OF PROGRAM":FORI=1 TO
      1000:NEXT I
```

```
1541 PRINT"{CLR}":POKE 36879,27:END
1550 ON D GOTO 1560,1570,1580,1590,1600,
     1610,1620,1630
1560 Y=Y-1
1570 X=X+1:RETURN
1580 X=X+1
1590 Y=Y+1:RETURN
1600 Y=Y+1
1610 X=X-1:RETURN
1620 X=X-1
1630 Y=Y-1:RETURN
1650 PRINT"{2 RIGHT}";:IFY=1THENPRINT"
     [A]";:GOTO1670
1660 PRINT"[Z]";
1670 FORX=0TOS-1:PRINT"*";:NEXTX:IFY=1T
     HENPRINT"[S]":RETURN
1680 PRINT"[X]":RETURN
1700 PRINT"{HOME}"TAB(14);:FORX9=1TOG:PR
     INT"{BLK}{DOWN}";:NEXTX9:RETURN
```

Lost Fox

Warren Pugh

"Lost Fox" is a fun game even young children will enjoy. It, like other programs in this book, uses a technique called chaining *to fit into an Unexpanded VIC. Be sure to SAVE Program 4-7 immediately following Program 4-6 on the same tape.*

Silky is the youngest fox in the city zoo. Silky is a born performer, and crowds of people every day watch the fox scamper around in the fox run. However, like a lot of young creatures, Silky doesn't much like getting the shots to prevent disease. Yesterday Silky saw the vet getting ready for the injections and decided it was time to leave the zoo.

So Silky burrowed out of the fox run — and completely out of the city zoo. Now the young fox is loose somewhere in the city. Silky doesn't know much about staying away from cars and buses, and there isn't much to eat. You are the zookeeper, and it's your job to find Silky before anything bad happens.

Fortunately, all the zoo animals have special collars that send out radio signals every now and then. You can pick up those signals. They tell you what direction you have to go in in order to find Silky. The possible directions are:

N

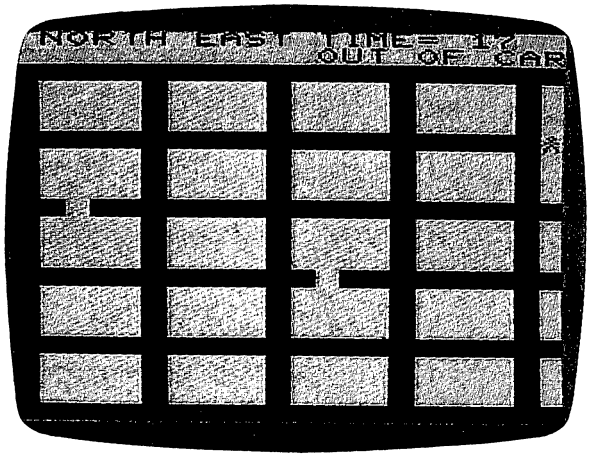
NW **NE**

W **you** **E**

SW **SE**

S

When you find Silky, the tired fox will be glad to see you. Silky didn't know how tough life could be outside the safe walls of the zoo. Getting back to plenty of good food, a nice warm place to sleep, and plenty of friendly people to perform for — that'll be worth getting a few shots from the vet, after all.



The zookeeper is getting close to the fox in "Lost Fox."

How to Play

Levels of difficulty. At the beginning of the game, before you begin your search for the lost fox, you will be asked what level of difficulty you want. 1 is the easiest level, and 8 is the hardest.

Where is Silky? When the game begins, Silky is hidden randomly somewhere in the city. At the easier levels of play, Silky stays in the same place. At the harder levels, the fox moves around.

What you see. The TV screen shows you an aerial view of the city, looking down on the streets and the city blocks. You will start from a position in the center of the city. You have at most 30 moves in which to find Silky.

How to move. You can move by pressing *N* for *north*, *S* for *south*, *E* for *east*, and *W* for *west*.

Radio signals. Every three moves (or every four or five, if you chose higher difficulty levels) you will get a reading from Silky's radio transmitter, telling you approximately what direction the fox is from where you are. But don't count on getting a signal every time — sometimes you might have a building or a bridge in the way, blocking out the radio reception.

Roadblocks. Also, beware of places where the road is blocked by construction or traffic jams or large trucks unload-

ing — it's easy to get boxed in there and waste a lot of your turns getting out again. There are more roadblocks at higher difficulty levels.

Getting closer. When you get within three spaces of Silky, you'll have to start searching on foot. On the screen, you'll know when this happens because the car will be replaced by a human figure. Once you're out of the car, you don't have to stay on the road anymore — you can make your way right through the city blocks.

Found! When you reach Silky, the fox will come out of hiding and a fanfare will sound. In the meantime, the game will reset itself so you can play again, if you want.

Even harder levels. Besides levels 1 to 8, there is a level 9 in which you can program your own obstacles.

Typing in the Program

So that "Lost Fox" will run on an Unexpanded VIC, it is loaded in two parts. Type in Program 4-6, SAVE it, and then type in Program 4-7 and SAVE it on the same tape following Program 4-6. So that Program 4-7 will fit, a question mark should be used for all PRINT statements and all extra spaces should be removed. So if a program line says for instance,

```
IFN=55THEN100
```

don't change it to read

```
IF N=55 THEN 100
```

or the program won't fit!

Program 4-6: Lost Fox, Part 1

```
1  V=36878:SS=36874:POKEV+1,57
2  PRINT"{CLR}{9 DOWN}{RIGHT}{WHT}** L O
     S T F O X **{BLK}"
3  PRINT"{8 DOWN}{RIGHT}":GOSUB 1000:FOR
     T=1 TO 1000:NEXT
4  POKEV+1,25:PRINT"{CLR}{DOWN}{RIGHT}
   {RED}THE FOX IS HIDDEN!{3 SPACES}EVER
   Y 3,4, OR 5 MOVES,YOU WILL BE ALERTED
   "
5  PRINT"AND WILL RECEIVE A{4 SPACES}REA
   DING. MOVE YOURSELFBY USING THE KEYS:"
```

```
6  18 PRINT"{DOWN}{6 SPACES}{RVS}N{OFF}- NO
      RTH{DOWN}{8 LEFT}{RVS}S{OFF}- SOUTH
      {DOWN}{8 LEFT}{RVS}E{OFF}- EAST{DOWN}
      {7 LEFT}{RVS}W{OFF}- WEST{DOWN}"
7  20 PRINT" YOU CANNOT LEAVE THE ROADWAY U
      NTIL YOU GET CLOSE ENOUGH. ONCE YOU";
8  22 PRINT"GET THE 'OUT OF CAR'{2 SPACES}M
      ESSAGE, THEN YOU MAY MOVE ANYWHERE!"
9  24 PRINT"{DOWN} BEWARE OF ROADBLOCKS!";
10 26 PRINT"{DOWN} HIT {RVS}RETURN{OFF} TO
      GO ON.";
11 28 GETA$:IFA$=""THEN28 //
12 100 GOTO1000 22
13 200 PRINT"{CLR}":POKE36879,253
14 205 PRINT"{BLK}{8 DOWN}{3 RIGHT}{RED}PLE
      ASE WAIT FOR {DOWN}{6 SPACES}PROGRAM
       TO LOAD."
15 207 PRINT"{BLU}{4 DOWN}"
16 210 POKE198,1:POKE631,131:END
17 1000 POKEV,15:READNS,DS:IFNS=-1THENPOKEV
      ,0:RETURN
18 1010 POKESS+1,NS:POKESS+2,NS:FORT=1TODS*
      100:NEXT:POKESS+1,0:POKESS+2,0
19 1020 FORT=1TO20:NEXT:GOTO1000 17
20 1030 DATA223,1,230,1,234,1,223,1,230,1,2
      34,1,223,1,230,1,234,1,0,4
21 1040 DATA231,2,230,1,0,2,230,2,227,1,0,2
      ,227,2,223,4,-1,0
22 1100 POKE51,0:POKE52,28:POKE55,0:POKE56,
      28:CB=7168
23 1110 READA:IFA=-1THEN200 13
24 1120 FORN=0TO7:READB:POKECB+A*8+N,B:NEXT
25 1130 GOTO1110 23
26 1140 DATA0,255,255,255,255,255,255,255,2
      55
27 1150 DATA30,68,56,56,16,57,58,60,40
28 1160 DATA35,84,124,84,16,56,126,127,36
29 1170 DATA36,16,40,16,124,16,40,68,68
30 1180 DATA32,0,0,0,0,0,0,0,0
31 1220 DATA-1
```

Program 4-7: Lost Fox, Part 2

```
5 CLR:V=36878:SS=36874
30 POKE36869,240
31 PRINT"{CLR}":POKEV+1,234
32 PRINT"{2 DOWN}{2 RIGHT}SELECT DEGREE
   OF":PRINT"{2 RIGHT}DIFFICULTY. (1-8)"
34 IFPEEK(197)=39THEN30
36 GETB$:IFVAL(B$)=0THEN36
37 B=VAL(B$)
38 ONBGOTO41,42,43,44,45,46,47,48
39 IFB=9THEN60
41 T1=3:T2=0:T3=0:T4=30:GOTO95
42 T1=4:T2=5:T3=0:T4=30:GOTO95
43 T1=3:T2=0:T3=1:T4=30:GOTO95
44 T1=4:T2=5:T3=1:T4=30:GOTO95
45 T1=5:T2=10:T3=0:T4=30:GOTO95
46 T1=5:T2=10:T3=1:T4=30:GOTO95
47 T1=4:T2=5:T3=2:T4=30:GOTO95
48 T1=5:T2=10:T3=2:T4=30:GOTO95
60 PRINT"{RED}{3 DOWN}":INPUT"# MOVES FO
   R READING";T1:INPUT"{DOWN}# OF ROADBL
   OCKS";T2
62 INPUT"{DOWN}# SPACES FOX MOVES";T3:IN
   PUT"{DOWN}# MOVES ALLOWED";T4
95 CC=0:TT=0:TC=0:RC=0:H=35
100 PRINT"{CLR}":POKE36879,222:POKE36869
   ,255
105 OF=30720:FORX=7680TO7680+43:POKEX,0:
   POKEX+OF,1:NEXT
110 S1=7724
115 FORX=1TO6
120 FORY=S1TOS1+21
125 POKEY,0:POKEY+OF,0:NEXTY
130 S1=S1+88
135 NEXTX
140 S1=7724
145 FORX=1TO5
150 FORY=S1TOS1+22*21STEP22
155 POKEY,0:POKEY+OF,0:NEXTY
160 S1=S1+5
165 NEXTX
```

```
170 RH=10:CH=10:LH=7724+(22*(RH)+CH):POK
    ELH,H:POKELH+OF,2:NRH=10:NCH=10
175 GOSUB1100:IFRC<T2THENGOTO175
190 RF=INT(RND(1)*20):CF=INT(RND(1)*21)
192 LF=7724+(22*(RF)+CF):IF LF=LHTHEN190
197 GOSUB500
198 PRINT"{HOME}{12 RIGHT}{RVS}{BLK}TIME
    ="TT"{BLU}"
200 GETA$
210 IFA$="N"THEN NR=RH-1
212 IFA$="S"THEN NR=RH+1
215 IFA$="E"THEN NC=CH+1
217 IFA$="W"THEN NC=CH-1
218 IFPEEK(197)=39THEN30
220 IFNC<0THENCH=0:NC=0:GOTO200
222 IFNC>21THENCH=21:NC=21:GOTO200
225 IFNR<0THENR=0:NR=0:GOTO200
227 IFNR>20THENR=20:NR=20:GOTO200
230 NL=7724+(22*(NR)+NC)
232 IFNL=LFTHEN300
233 IFA$=""THEN235
234 IFCC>0THEN300
235 IFPEEK(NL)<>0THENNR=RH:NC=CH:GOTO200
300 POKELH,SC:POKELH+OF,SL
305 SC=PEEK(NL):SL=PEEK(NL+OF)
310 POKENL,H:POKENL+OF,2
311 TT=TT+1:TC=TC+1
312 PRINT"{HOME}{12 RIGHT}{RVS}{BLK}TIME
    ="TT"{BLU}"
313 POKEV,15:POKESS+2,230:FORT=1TO50:NEX
    T:POKESS+2,0:POKEV,0
315 A$=""
320 CH=NC:RH=NR:LH=NL
350 IFLH=LFTHEN900
355 IFTC=T1-1THENGOSUB1200
360 IFTC=T1THENGOSUB500
370 GOSUB519
375 IFTT=T4THENPOKEV,15:POKESS,200:FORT=
    1TO1500:NEXT:POKEV,0:POKESS,0
380 IFTT=T4THENPRINT"{HOME}{DOWN} {RVS}
    {PUR}YOU LOSE!!{BLU}":POKELF,30:POKE
    LF+OF,2:FORT=1TO8000:NEXT:GOTO95
```

```
70  400 GOTO200
71  500 PRINT"{HOME}{RVS}{PUR}**READING**
        {BLU}":POKEV,15:POKESS+2,220:FORT=1T
        O1500:NEXT:POKESS+2,0:POKEV,0
72  501 PRINT"{HOME}{11 SPACES}":GOSUB1100
73  502 RK=INT(RND(1)*10):IFRK=5THEN518
74  503 IFRH>RFTHENPRINT"{HOME}{RVS} NORTH
        {5 RIGHT}"
75  505 IFRH<RFTHENPRINT"{HOME}{RVS} SOUTH
        {5 RIGHT}"
76  507 IFRH=RFTHENPRINT"{HOME} {WHT}{RVS}
        {5 RIGHT}{OFF}{BLU}{5 SPACES}"
77  510 IFCH>CFTHENPRINT"{HOME}{RVS}
        {7 RIGHT}WEST"
78  515 IFCH<CFTHENPRINT"{HOME}{RVS}
        {7 RIGHT}EAST"
79  517 IFCH=CFTHENPRINT"{HOME}{7 RIGHT}
        {WHT}{RVS}{4 RIGHT}{OFF}{BLU}"
80  518 TC=0
81  519 IFCC>1THEN550
82  520 IFABS(CH-CF)<3ANDABS(RH-RF)<3THENCC=
        1
83  525 IFCC=1THEN530
84  527 GOTO550
85  530 POKEV,15:FORX=1TO5:POKESS,250:FORT=1
        TO100:NEXTT:POKESS,0:FORT=1TO25:NEXT
        T:NEXTX
86  535 IFCC=1THENPRINT"{HOME}{DOWN}
        {12 RIGHT}{BLK}{RVS}OUT OF CAR{BLU}"
        :CC=CC+1:H=36:POKELH,H
87  550 RETURN
88  900 PRINT"{HOME}{DOWN}{RED}{RVS}
        {12 RIGHT}FOX FOUND!{BLU}":CC=0
89  902 POKELF,30:POKELF+OF,6
90  905 GOSUB1000
91  907 POKELF,36
92  910 FORT=1TO5000:NEXT:GOTO95
93  1000 POKEV,15:READNS,DS:IFNS=-1THENPOKEV
        ,0:RESTORE:RETURN
94  1010 POKESS+1,NS:POKESS+2,NS:FORT=1TODS*
        100:NEXT:POKESS+1,0:POKESS+2,0
95  1020 FORT=1TO20:NEXT:GOTO1000
```

```
96 1030 DATA223,1,230,1,234,1,223,1,230,1,2
        34,1,223,1,230,1,234,1,0,4
97 1040 DATA231,2,230,1,0,2,230,2,227,1,0,2
        ,227,2,223,4,-1,0
98 1100 RB=INT(RND(1)*20*21)+7724:IFPEEK(RB
        )=0THENPOKERB,32:RC=RC+1
99 1110 RETURN
   1200 MR=((INT(RND(1)*3))-1)*T3:MC=((INT(
        RND(1)*3))-1)*T3
101 1210 RF=RF+MR:CF=CF+MC
102 1212 IFRF<0THENRF=0
103 1214 IFRF>20THENRF=20
104 1216 IFCF<0THENCF=0
105 1218 IFCF>21THENCF=21
106 1220 LF=7724+(22*(RF)+CF):IF LF=LHTHEN12
        00
107 1240 RETURN
```

Pharaoh's Treasure

Clark and Kathryn H. Kidd

"Pharaoh's Treasure" offers the user of the Unexpanded VIC an adventure game that you would expect only on computers with more memory. The adventure takes practice and it helps to create a map of the tomb.

The Legend of the Treasure

You are in the middle of mysterious Egypt, standing at the entrance to the pyramid tomb of King Ramus IV. Deep within the pyramid is a golden sarcophagus. If you can find it, you will be rich beyond your wildest dreams.

Although many others have sought the treasure, no explorer has ever returned from the tomb. Rumors claim the pyramid is a maze of passages, some of which run in circles or lead to dead ends.

Hieroglyphics at the pyramid entrance warn of five deadly perils awaiting those who would violate the sanctity of the tomb. No clue is given in the warning about the five perils, but you need to arm yourself against them if you can.

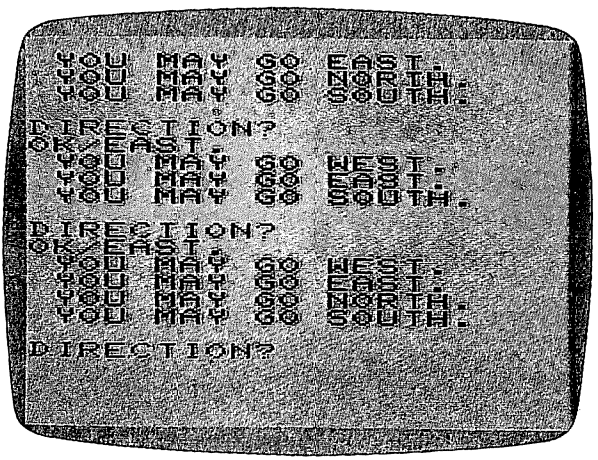
Previous explorers have left various tools strewn along the passageways of the tomb. If you find one of these tools, it may help you bypass a peril and reach the treasure. If you run into a danger for which you have no tool, you must turn around and go back.

Local tradition has told you one of the five perils of the tomb is the Cat of the Golden Sarcophagus, a magical statue that was placed on the sarcophagus to guard Pharaoh Ramus IV. A previous explorer moved the cat from its original location, so it may not be near the sarcophagus. The Cat of the Golden Sarcophagus is a fearful peril, but you may get around it with the proper tool.

Although you may be saved from the Cat of the Golden Sarcophagus, nothing can save you from the Pharaoh's Curse.

If the planets are in the proper positions and the desert sands blow just right, the curse of the Pharaoh may fall upon you. If you receive the Pharaoh's Curse, four things will happen:

- You will be covered with a cloud of orange smoke and banished to a random location within the tomb.
- If you have not yet found the treasure, the golden sarcophagus will be transported to a new location.
- All tools you have not yet found will be moved to new locations.
- All perils you have not yet encountered will be moved to new locations.



This adventure becomes easier when you create a map to help you locate the treasure.

How to Play

The game begins at a secret tomb entrance. Remember where you enter the pyramid, for you will have to leave by the same opening. To move, press F1 for north, F3 for south, F5 for east, and F7 for west. You may want to make a map of the tomb so you will avoid backtracking as well as be able to find your way out again.

You're looking for the Golden Sarcophagus; as soon as you find the treasure, all you need to do to win is leave the tomb. Any tools you find along the way will count as a

bonus. Any perils you encounter without the proper tool to counteract them will subtract from your score. If you find several tools on your way to the treasure, you may want to stay in the tomb after reaching the Golden Sarcophagus to find the perils for those tools and add to your score.

Besides choosing a direction to move, you have two other things you can do. Press C to cause the Pharaoh's Curse to fall upon you. This is a last resort, to redistribute all the objects in the tomb if a peril for which you don't have a tool guards the only entrance to the treasure.

If you think you've looked everywhere and just can't find the sarcophagus, press O to end the game. The treasure *is* there, however, even if you haven't found it. But it's no disgrace to lose; Egyptologists have been searching for the Golden Sarcophagus for centuries without success. Maybe you'll succeed on your next trip inside the pyramid.

Scoring

Although the object of the treasure hunt is to escape the tomb with the Golden Sarcophagus, scores are provided so players may compare their skills:

500 points for entering the tomb.
500 points for finding the treasure.
500 points for escaping the tomb with the treasure.
100 points for finding a tool.
50 points for finding a peril and having the correct tool.
-100 points for getting the Pharaoh's Curse.
-25 points for finding a peril without the correct tool.
-5 points for each minute spent in the tomb.
-1 point for each turn taken.

Any player with a score of 2000 or better should be given the respect due an expert Egyptologist.

Program 4-8: Pharaoh's Treasure

```
5 DIMCP%(3,3)
10 CT%=828:Z$=CHR$(13):PRINT"{CLR}{RED}
   {DOWN}WELCOME TO.....{2 DOWN}"
15 PRINT"{3 DOWN}"SPC(8)"{BLK}T H E"Z$"
   {3 DOWN}"SPC(2)"P H A R A O H ' S"Z$
16 PRINT"{2 DOWN}"SPC(3)"T R E A S U R E
   "
```

```
17 LV=36878:V1=36874:V2=36875:V3=36876
20 FORX=0TO189:READN%:POKECT%+X,N%:NEXT
25 FORX=0TO5:READX$,Y$:PT$(X)=X$:TT$(X)=
   Y$:TT%(X)=0:NEXT:TU%=0
60 GOSUB600:L%=1:SC%=500:MS=TI/60:QQ%=IN
   T(RND(1)*4)
62 FORX=0TO3:READX$:CP$(X)=X$:NEXT
65 GOSUB400
66 FORX=0TO3:FORY=0TO3:READX%:CP%(X,Y)=X
   %:NEXT:NEXT
70 X%=((L%-1)*5)+CT%:N%=PEEK(X%):S%=PEEK
   (X%+1):E%=PEEK(X%+2)
75 W%=PEEK(X%+3):P%=PEEK(X%+4):T%=0
80 IFP%>5THENT%=P%-5:P%=0
85 IFP%=0THEN125
86 IFTT%(P%-1)=2THEN125
90 PRINT"{CLR}{3 DOWN}{RED}LOOK! UP AHEA
   D! --{2 DOWN}"Z$"A "PT$(P%-1)". {2 DOWN}"
95 IFTT%(P%-1)=0THEN105
100 PRINT"BUT YOU HAVE A{2 DOWN}"Z$TT$(P
    %-1)"{2 DOWN}"Z$"AND MAY CONTINUE.
    {BLK}":Z=0:SC%=SC%+50
103 TT%(P%-1)=2:GOTO120
105 PRINT"YOU DON'T HAVE{2 DOWN}"Z$"THE
    PROPER TOOL,{2 DOWN}"Z$"SO YOU MUST
    GO BACK.{BLK}"
110 L%=PL%:Z=1:SC%=SC%-25
120 GOSUB400
123 IFZ=1THENGOTO70
125 IFT%<>7ORTT%(5)=0THEN150
130 PRINT"{CLR}{GRN}{DOWN}CONGRATULATION
    S!!!!!{2 DOWN}"Z$" YOU ESCAPED
    {2 DOWN}"Z$"{2 SPACES}WITH THE TREAS
    URE!!"
140 PRINTSPC(2)"{2 DOWN}{RVS}{RED}Y O U
    {4 SPACES}W O N{OFF}{BLU}{2 DOWN}":S
    C%=SC%+500:GOTO380
150 IFT%=0ORT%=7THEN200
155 IFTT%(T%-1)>0THEN200
160 TT%(T%-1)=1:PRINT"{CLR}{3 DOWN}LOOK!
     ON THE GROUND --{DOWN}"Z$"A "TT$(T%
    -1)"!{2 DOWN}"
```

```
170 PRINT"WE'LL SAVE IT.":SC%=SC%+100:IF
    T%=6THENSC%=SC%+400
180 GOSUB400
200 GOTO500
201 IFN%>0THENX=0:GOSUB700
202 IFS%>0THENX=1:GOSUB700
204 IFE%>0THENX=2:GOSUB700
206 IFW%>0THENX=3:GOSUB700
207 TU%=TU%+1
210 PRINT"{DOWN}{BLU}DIRECTION?{BLK}"
215 GETX$
220 IFX$>CHR$(132)ANDX$<CHR$(137)THEN280
255 IFX$="Q"THEN370
257 IFX$="C"THEN505
260 IFX$<>""THEN210
270 GOTO215
280 X=ASC(X$)-133:Y=0
285 IFX=CP%(QQ%,Y)THEN290
287 Y=Y+1:GOTO285
290 ONY+1GOTO300,305,310,315
300 X%=N%:GOTO350
305 X%=S%:GOTO350
310 X%=E%:GOTO350
315 X%=W%
350 IFX%=0THENPRINT"DEAD END.":GOTO200
360 PRINT"OK/"CP$(X)".":PL%=L%:L%=X%:GOT
    O70
370 PRINT"{CLR}{2 DOWN}{BLK}SO SORRY THA
    T YOU"Z$"{2 DOWN}COULDN'T FIND YOUR"
    Z$"{2 DOWN}WAY OUT.{2 DOWN}"
380 X%=INT(((TI/60)-MS)/60)*5:SC%=SC%-X%
    -TU%
385 PRINT"{2 DOWN}{BLU}YOU USED"TU%"TURN
    S."
390 PRINT"{2 DOWN}{BLU}YOUR SCORE IS"SC%
    "{LEFT}."
392 INPUT"{2 DOWN}PLAY AGAIN? (Y/N)";X$
394 IFX$="N"THENEND
396 IFX$<>"Y"THEN392
398 RESTORE:GOTO10
400 X%=INT(RND(1)*7+5)
402 FORX=1TOX%:POKELV,X:Y=INT(RND(1)*50+
```

```
    128):POKEV2,Y:POKEV3,128:POKEV1,Y-50
404 FORY=1TO350:NEXTY
408 NEXTX:POKELV,0
409 POKEV1,0:POKEV2,0:POKEV3,0
410 PRINT"{2 DOWN}{3 SPACES}(PRESS ANY K
    EY)"
415 GETD$:IFD$=""THEN415
420 PRINT"{CLR}":RETURN
500 X%=INT(RND(1)*500+1):IFX%>2THEN201
505 SC%=SC%-100:GOSUB600
510 X%=INT(RND(1)*38+1):Y%=PEEK(((X%-1)*
    5)+4+CT%)
515 IFY%<>0THEN510
520 L%=X%:PRINT"{CLR}{PUR}{DOWN}H O R R
    O R S !!!"Z$"{DOWN}YOU HAVE BECOME A
    "Z$"{DOWN}VICTIM OF THE AWFUL"
530 PRINT"{DOWN}''PHARAOH'S CURSE''"Z$"
    {DOWN}AND WILL BE BANISHED"Z$"{DOWN}
    TO AN UNKNOWN PART"
540 PRINT"{DOWN}OF THE TOMB!!!{BLK}":GOS
    UB400:GOTO70
600 FORX=1TO37:POKECT%+4+(X*5),0:NEXT
630 X%=INT(RND(1)*19+19):N%=(X%*5)+4+CT%
    :POKEN%,11
635 FORX=1TO10
640 X%=INT(RND(1)*37+1):N%=(X%*5)+4+CT%:
    S%=PEEK(N%)
645 IFS%<>0THEN640
650 POKEN%,X:NEXT:RETURN
700 Y%=CP%(QQ%,X):X$=CP$(Y%):PRINT" YOU
    MAY GO "X$".":RETURN
901 DATA0,10,4,2,12,0,11,1,3,0,0,0,2,0,0
    ,0,9,5,1,0,0,7,6,4,0
903 DATA0,0,0,5,0,5,14,8,0,0,0,0,0,7,0,4
    ,16,0,10,0,1,13,9,0,0
905 DATA2,18,0,12,0,0,0,11,0,0,10,0,0,0,
    0,7,0,27,15,0,0,26,14,16,0
906 DATA9,0,15,17,0,0,20,16,18,0,11,38,1
    7,19,0,0,0,18,0,0,17,21,25,0,0
907 DATA20,33,0,23,0,0,23,37,38,0,22,34,
    21,24,0,38,35,23,36,0,31,30,32,20,0
909 DATA15,0,28,29,0,0,0,0,14,0,0,0,0,26
```

```
    ,Ø,Ø,Ø,26,Ø,Ø,25,Ø,Ø,Ø,Ø
911 DATAØ,25,Ø,Ø,Ø,Ø,Ø,Ø,25,Ø,21,Ø,Ø,Ø,Ø
    ,23,Ø,Ø,Ø,Ø,24,Ø,Ø,Ø,Ø
913 DATAØ,Ø,24,Ø,Ø,Ø,Ø,Ø,22,Ø,18,24,22,Ø
    ,Ø,"DEEP PIT",LOG,"QUICKSAND POOL",R
    OPE
915 DATA"ROCK SLIDE",SHOVEL,"GUARDIAN CA
    T","SACRED AMULET"
917 DATA"LOCKED DOOR",KEY,X,"*GOLD SARCO
    PHAGUS*",NORTH,SOUTH,EAST,WEST
92Ø DATAØ,1,2,3,1,Ø,3,2,2,3,1,Ø,3,2,Ø,1
```

Part 5

Scrolling

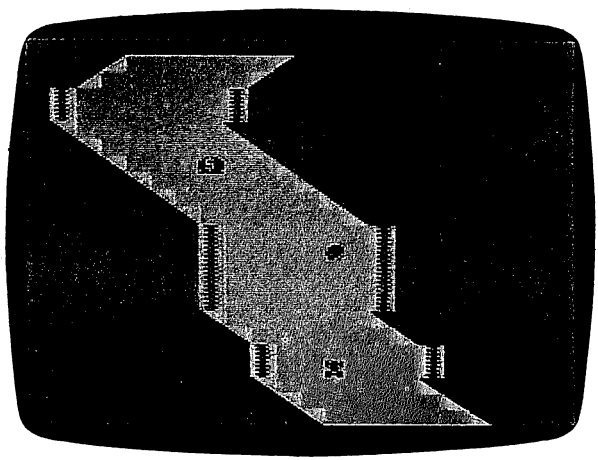
Grand Prix Foo

Mark Vittek

This game uses scrolling to create the effect of driving. It takes practice to play well.

"Grand Prix Foo" uses scrolling to create the effect of driving on a highway. Unlike most interstate highways, though, this roadway is curved and winding. Your job as driver is to negotiate the curves without running off the road. If you are daring, and you think driving on a curvy road is too easy, you always have the option of adding potholes.

For more of a challenge, you can play "Money Foo." Here you not only negotiate the road, but also attempt to gather money bags as you go along. In both games, you use a joystick to turn, accelerate, and brake.



Test your driving skill with "Grand Prix Foo."

Grand Prix Foo consists of two programs chained together. Type in and SAVE Program 5-1 to tape. Then immediately following it, type in and SAVE Program 5-2 on the same tape. Once both programs are SAVEd, rewind the tape and LOAD

Program 5-1. Program 5-1 will run Program 5-2 at the proper time. Just be sure to leave the play button pressed until Program 5-2 is running.

Program 5-1. Grand Prix Foo Instructions

```
1 POKE36879,26:PRINT"{CLR}{RVS}{RED}
  {2 SPACES}GRAND PRIX FOO{6 SPACES}"
10 PRINT"{DOWN}{BLK} TO DRIVE, YOU MUST
   {4 SPACES}USE THE JOYSTICK.":GOTO100
20 PRINT"{DOWN} PULL DOWN TO{7 SPACES}
   {2 RIGHT} ACCELERATE{12 SPACES}PUSH U
   P TO{12 SPACES}BRAKE"
30 PRINT"{DOWN} THE FARTHER DOWN THE
   {2 SPACES}ROAD YOU GO, THE MORE POINT
   S YOU EARN.{2 SPACES}"
35 PRINT"{PUR}{DOWN} WAIT WHILE I LOAD T
   HE{2 SPACES}RACE TRACK.{BLK}"
40 PRINT"{DOWN}{RIGHT}{RVS}GOOD LUCK
   {WHT}"
50 POKE198,5:POKE631,78:POKE632,69:POKE6
   33,87:POKE634,13:POKE635,131:END
100 X=28:POKE52,X:POKE56,X:POKE51,PEEK(5
    5):C=256*PEEK(52)+PEEK(51):FORI=CTOC
    +511
101 POKEI,PEEK(I+32768-C):NEXT:
102 READJ:IFJ=-1THEN200
103 POKE7168+H,255-J:H=H+1:GOTO102
200 FORT=0TO7:READS:POKE7424+T,S:NEXT:CL
    R:GOTO20
1000 DATA24,60,24,60,24,60,24,60,12,92,6
     2,126,124,124,56,,255,127,63,31,15,
     7,3,1,255,254
1001 DATA252,248,240,224,192,128,128,192
     ,224,240,248,252,254,255,1,3,7,15,3
     1,63,127,255
1002 DATA,,,,,,,,102,126,102,24,24,90
1003 DATA126,90
1005 DATA24,24,255,255,24,24,24,24,255,1
     18,227,239,227,251,98,52,24,24,126,
     24,60,126,126
1020 DATA126,-1,,,,,,,,
```

Program 5-2. Grand Prix Foo

```
1 X=-1:G=8:I=4:Z=36879:L=145:POKEZ,18:PO
  KEZ-10,240
2 GOTO500
10 A=0:POKEP1+3,127:P=PEEK(P1+1)AND128:J
   0=-(P=0):POKEP1+3,255:P=PEEK(P1):J1=-
   ((PAND8)=0)
11 J2=-((PAND16)=0):J3=-((PAND4)=0)
12 POKEZ-4,0:IFJ2THENA=A-1
13 IFJ0THENA=A+1
15 IFJ1THENA=A+22:V=V+.5
16 IFJ3THENA=A-22:V=V-.5
20 IFB<-59ORB>451THEN700
25 POKEZ-3,L+V*2
100 L1=L1+1:L=L+.8:K=K+1:Q=Q+1+V:IFL>220
    THENL=130+(RND(1)*20)
101 READS:IFS=99THENRESTORE:GOTO101
103 G=G+S:IFS=1THENC=67
104 IFS=1THENC=68:C1=66
105 IFS=-1THENC=67:C1=69
106 IFS=0THENC=64:C1=64
110 IFG<1THENG=1
111 IFG+I>18THENG=G-1
112 IFK=25THENK=0
113 IFK=K3THENPOKE8143+G+(RND(1)*I),9
114 POKE7734+B,6:B=B+A:IFK=K1THENK=0:X=R
    ND(1)*I:POKE8143+G+X,1
115 PRINTTAB(G)"{YEL}"CHR$(C)"{WHT}";:FO
    RT=0TOI:PRINT"F";:NEXT:PRINT"{YEL}"C
    HR$(C1)"{WHT}"
116 J=PEEK(7734+B):IFJ<6ORJ=32THEN700
117 JJ=10:IFJ1THENJJ=PEEK(7734+B-22):IFJ
    J<6THEN700
118 IFJ=9ORJJ=9THENM=M+1:POKE7712+B,8:PO
    KE38432+B,(RND(3)*3)+3:POKEZ-4,(RND(
    5)*100)+150
120 POKE7734+B,7:GOTO10
500 PRINT"{CLR}{RED}{5 SPACES}*WELCOME T
    O*{8 SPACES}*GRAND PRIX FOO*
    {3 SPACES}====================";
501 PRINT"{2 SPACES}TRY YOUR RACING
```

```
    {6 SPACES}SKILL ON THE TUBE
    {3 SPACES}":I=5
502 PRINT"{RVS}{BLK}{DOWN} HOW WIDE DO Y
    OU WANT{2 SPACES}THE TRACK (1 TO 5)"
    :INPUT"{2 RIGHT}";I:IFI>5ORI<1THEN50
    0
510 P1=37151:PRINT"{PUR}{DOWN}{RVS}
    {DOWN}DO YOU WANT DANGEROUS
    {3 SPACES}{OFF} POT HOLES ?"
511 GETA$:IFA$="Y"THEN520
512 IFA$="N"THENK1=-1:GOTO530
513 GOTO511
520 K1=8:PRINT"{DOWN}{RVS}{BLK} WHAT POT
    HOLE LEVEL{4 SPACES}":INPUT"
    {2 RIGHT}";K1:K1=K1*2
525 IFK1<5ANDK1>=0THEN550
530 PRINT"{2 DOWN}{RVS}{BLU} DO YOU WANT
     TO PLAY{3 SPACES}{OFF}MONEY
    {2 SPACES}FOO ?"
531 GETA$:IFA$="Y"THENK3=5:GOTO550
532 IFA$="N"THENK3=-1:GOTO550
533 GOTO531
550 PRINT"{CLR}{BLU} PRESS ANY KEY TO GO
    {2 SPACES}":
556 GETA$:IFA$=""THEN556
557 PRINT"{HOME}{RVS}{5 DOWN} COUNTDOWN"
    :FORT=9TO1STEP-.05:PRINT"{HOME}{PUR}
    {6 DOWN}{11 LEFT}{RVS}"INT(T):NEXT
600 POKEZ-10,255:FORT=0TO20:POKE7734,7:P
    RINTTAB(8)"{YEL}@";:FORY=0TOI:PRINT"
    F";:NEXT
610 PRINT"@":NEXT:POKEZ,104:POKEZ-1,10:G
    OTO10
700 POKEZ-3,0:FORD=12TO0STEP-1
701 READS:IFS=99THENRESTORE:GOTO701
702 G=G+S:POKEZ-2,130+(RND(3)*120):POKEZ
    ,8:POKE7734+B,6:B=B+22:IFB+7734=>812
    0THEND=0
703 IFG<1THENG=1
704 IFG+I>18THENG=G-1
705 PRINTTAB(G)"{RED}"CHR$(C)"{WHT}";:FO
    RT=0TOI:PRINT"F";:NEXT:PRINT"{RED}"C
    HR$(C)"{WHT}"
```

```
707 POKE7734+B-22,6:PQKE7734+B,7:POKEZ,D
    *8:NEXT:POKE7734+B,8
720 FORT=15TOØSTEP-.Ø3:POKEZ-1,T:POKEZ-2
    ,143:NEXT:POKEZ,11Ø:POKEZ-1,Ø
740 FORT=1TO1500:NEXT:POKEZ-1Ø,24Ø
750 PRINT"{CLR}{WHT}{2 DOWN} YOU WENT "I
    NT(L1/23)"{LEFT} MILES":PRINT" YOU E
    ARNED"INT(Q/1Ø)"POINTS"
751 POKEZ-2,Ø:IFK3>ØTHENPRINT"{DOWN} YOU
     GOT"M"MONEYBAGS"
755 PRINT"{2 DOWN} DO YOU WANT TO START
    {2 SPACES}OVER {RVS}(ENTER Y IF SO.)
    "
756 PRINT" OR C TO CONTINUE":POKEZ-2,Ø
760 INPUTA$:IFA$="Y"THENRUN
761 IFA$="C"THENPRINT"{CLR}":G=8:B=Ø:V=Ø
    :GOTO557
762 END
1ØØØ DATA,,,,,1,1,1,1,1,1,,,-1,-1,,-1,-1
     ,-1,-1,-1,-1,-1,-1,-1,-1,1,1,1,1,,,
1ØØ1 DATA,1,1,1,1,1,,,-1,-1,,,-1,-1,1,1,1,
     -1,-1,-1,-1,-1,-1,-1,-1,-1,,,-1,-1,,1,
     1,1,1,,,1,1,99
```

Part 6

Dexterity

Thunderbird

Dave Sanders

Fast action is what you'll get when you play "Thunderbird." Try it and see if you're skilled enough to keep the satellite from flying past the Thunderbird.

"Thunderbird" will demand your undivided attention and all of the memory the unexpanded VIC-20 has to offer. The object of Thunderbird is to score as high as possible. The high score will be kept from game to game. The scoring is as follows: 200 points for taking out a tree, 50 points for taking out a saucer, 75 points for deflecting off either wing of the Thunderbird, 25 points for deflecting off the main body of the Thunderbird, and 1000 points for breaking out the bottom of the playing field. When the satellite drops into a well, 125 points are subtracted from the score.

You score these points by keeping the satellite in the playing field. The satellite can break out the top and the bottom of the screen. When it breaks out the bottom, you score 1000 points, and a new and more difficult playing field is set up for you. If the satellite breaks out the top of the field, your game is half over. You can lose only two satellites out the top. You prevent the satellite from breaking out the top by deflecting it back into the field with the Thunderbird. The Thunderbird is moved across the top of the field with the cursor control keys.

The display on the right side of the screen tells you if you are playing the first or second satellite. When the satellite drops into a well, the Thunderbird lasers down from one to three multicolored saucers to further hinder the satellite from breaking out the bottom. You will notice that the Thunderbird deflects the satellite one way off its main body and a different way off its wings. You have to keep the Thunderbird moving across the screen in conjunction with the direction the satellite is moving, or you will not play for very long.

With a little practice, the first breakout is not too hard. The second breakout will not be out of reach either, but no

one in our neighborhood has broken out the third time. Just in case you are a whiz though, the game will continue to get harder.

Typing in Thunderbird

Thunderbird uses almost all the memory of the unexpanded VIC. So, in order to get Thunderbird to fit into memory, a technique called "crunching" is used (see below). Thus it is necessary, when typing in this program, to use keyword abbreviations. These can be found in Appendix D of *Personal Computing on the VIC-20* which comes with each VIC.

Crunching It into the VIC

You can pack more instructions — and power — into your BASIC programs by making each program as short as possible.

Crunching programs lets you squeeze the maximum possible number of instructions into your program. It also helps you reduce the size of programs which might not otherwise run in a given size.

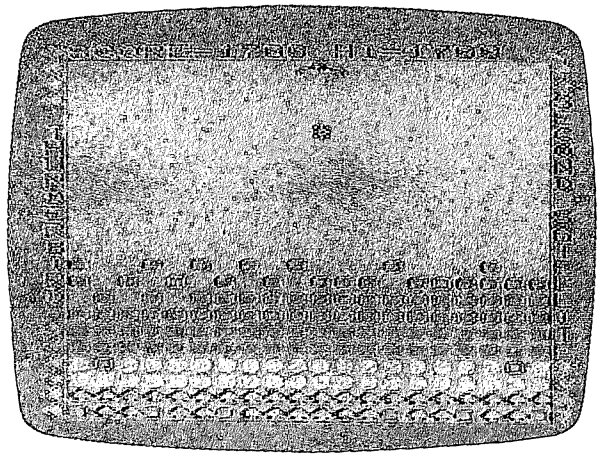
Using keyword abbreviations is helpful when you program because you can actually crowd more information on each line by using these abbreviations. In this program it is mandatory to use this technique on many of the lines when you type them in. The most frequently used abbreviation in this program is PO (P shifted-O) which is the BASIC abbreviation for the POKE command. However, if you LIST a program that has abbreviations, the VIC-20 will automatically print out the listing with the full-length keywords.

If any program line exceeds 88 characters (four lines on the screen) with the keywords unabbreviated, and you want to change it, you will have to re-enter that line with the abbreviations before saving the program.

SAVEing a program incorporates the keywords without inflating any lines because BASIC keywords are tokenized by the VIC-20. Usually, abbreviations are added after a program is written and do not have to be LISTed any more before SAVEing.

REM statements are helpful in reminding yourself — or showing other programmers — what a particular section of a program is doing. However, when the program is completed

and ready to use, you probably will not need those REM statements any more; you can save quite a bit of space by removing them. If you plan to revise or study the program structure in the future, it is a good idea to keep a copy on file with the REM statements intact.



"Thunderbird" for the VIC requires fast reflexes.

Following is a list of REM statements that would have been in my program if there had been room.

Program Lines	Description
4-17	Routine to set up playing field.
25-26	Subroutine for printing score.
50-54	Routine for making game more difficult.
55-59	Routine for displaying instructions and a short game.
65-80	Routine for firing laser and starting satellite back at a random location.
85	Routine for the graphics when satellite takes out saucers.
90-91	Routine for sounds and points on breakout.
95-98	Routine for sounds and colors on losing satellite out the top of the field.
100-103	Routine for moving Thunderbird across screen.
104-118	This section moves satellite and has all the PEEKs for the other routines in the program.

One of the easiest ways to reduce the size of your program is to eliminate all the spaces. Although programmers often include spaces in sample programs to provide clarity, you actually do not need any spaces in your program and will save memory if you eliminate them.

Instead of PRINTing several cursor commands to position a character on the screen, it is often more economical to use the TAB and SPC instructions to position words or characters on the screen. Well, that's enough on "crunching." You can find these and many other useful instructions in the *VIC-20 Programmer's Reference Guide* published by Commodore.

On line 68 a couple of saucers are lasered down by the Thunderbird. The screen code POKEd for the saucers is the same as all the other saucers, but they certainly look different. This effect is achieved by POKEing a nine into the color code location for these saucers. POKEing a color location with a number above eight will switch that location into multicolor mode. You can get some very interesting shapes and colors by using multicolor.

In lines four through seven, the (Q) is the ball graphic, and the (W) is the circle.

Program 6-1. Thunderbird

```
2 VD=36874:F=125:OX=30742:OF=30720:P1=1:
  L=1:SC=0:HI=0:K=1:M=7703:RS=1:VA=VD+2:
  C=VA+3
4 PRINT"{CLR}":POKEC,105:FORR=1TO17:PRIN
  T:NEXT:PRINT"{YEL}QQQQQQQQQQQQQQQQQQQQ
  QQQQQQ{WHT}W{YEL}QQQQQQQQQQ{WHT}W
  {YEL}QQQQQ"
5 PRINT"{WHT}{UP}QQQQQQQQQQQQQQQQQQQQQQ
  QQQQQQQQQQQQQQQQQQQQQQ"
6 PRINT"{CYN}{UP}QQQQQQQQQQQQQQQQQQQQQQ
  QQQQQQQQQQQQQQQQQQQQQQ"
7 PRINT"{PUR}{UP}QQ{WHT}W{PUR}QQQQQQQQQQ
  QQQQQQ{WHT}W{PUR}QQQQQQQQQQQQQQQQQQQQ
  QQQ"
8 FORR=8142TO8186:POKER,65:NEXT:J=87:G=8
  1:FORR=38423TO38442:POKER,7:NEXT:POKE3
  6878,15
9 FORR=38863TO38882:POKER,5:NEXT:FORR=38
```

```
  885TO3890Ø4:POKER,5:NEXT:POKE8165,J:POK
  E8168,J:POKE8171,J
1Ø POKE8178,J:POKE8181,J:POKE8184,J:FORR
  =7987TO8141STEP22:POKER+OF,1:NEXT:FOR
  R=768ØTO8164STEP22
11 READA:POKER,A:NEXT:FORR=77Ø1TO8185STE
  P22:READA:POKER,A:NEXT:FORR=7681TO77Ø
  Ø:READA:POKER,A:NEXT
12 FORR=7966TO812ØSTEP22:POKER+OF,1:NEXT
  :RESTORE:IFP1=>2THEN5Ø
13 PRINTSPC(6)"{RVS}{WHT}{22 UP}"SC:POKE
  7686,189:PRINTSPC(14)"{RVS}{WHT}{UP}"
  HI:POKE7694,189
14 X=1:Y=1:DX=1:DY=1:POKEM+1,85:POKEM+2,
  88:POKEM+3,73:IFRS=1THENRS=RS+1:GOTO5
  5
15 IFTT=5ØØTHENTT=1:X=12:L=1:SC=Ø:PRINT"
  {HOME}{7 RIGHT}{RVS}{5 SPACES}":GOTO1
  Ø4
16 IFL<>2THEN1Ø4
17 POKE7767,147:POKE7789,133:POKE7811,13
  1:POKE7833,143:POKE7855,142:POKE7877,
  132:GOTO1Ø4
25 PRINTSPC(6)"{UP}{RVS}"SC:POKE7686,189
  :IFSC>HITHENHI=SC:PRINTSPC(14)"{RVS}
  {UP}"HI:POKE7694,189
26 RETURN
5Ø FORR=7945TO7964:POKER,G:NEXT:POKE8Ø59
  ,J:POKE8Ø7Ø,J:FORR=7945TO7964:POKER+O
  F,7:NEXT
51 IFP1=>3THENPOKE8Ø12,J:POKE8Ø29,J
52 IFP1=>4THENPOKE81Ø5,J:POKE8112,J
53 IFP1=>5THENPOKE7951,J:POKE7958,J
54 GOTO13
55 POKE7754,8:POKE7755,9:POKE7756,2Ø:POK
  E7799,153:POKE784Ø,2Ø:POKE7841,15:POK
  E7843,16:POKE7844,12
56 POKE7845,1:POKE7846,25:POKEM+1,85:POK
  EM+2,88:POKEM+3,73:POKE7783,42:POKE79
  Ø3,21:POKE79Ø4,19:POKE79Ø5,5
57 POKE79Ø7,3:POKE79Ø8,21:POKE79Ø9,18:PO
  KE791Ø,19:POKE7911,15:POKE7912,18:POK
```

```
   E7914,11:POKE7915,5
58 POKE7916,25:POKE7917,19:POKE7925,6:PO
   KE7926,15:POKE7927,18:POKE7929,18:POK
   E7930,9:POKE7931,7
59 POKE7932,8:POKE7933,20:POKE7935,38:PO
   KE7937,12:POKE7938,5:POKE7939,6:POKE7
   940,20
60 GETA$:IFA$="Y"THENSC=0:L=1:GOTO4
61 IFA$<>"Y"THENPOKEVA,0:TT=TT+1:IFTT=5
   00THEN4
62 GOTO60
65 SC=SC-F:G=M+2:IFHI=SC+FTHENHI=HI-F
66 POKEG+22,77:POKEC,10:POKEG+OF+22,1:G=
   G+22
67 IFPEEK(G+22)=81ORPEEK(G+22)=65THENPOK
   EG,81:POKEG+OF,9:GOTO72
68 IFPEEK(G+22)=87THENPOKEG,81:POKEG-22,
   81:G=G-22:POKEG+OF,9:POKEG+OX,9:GOTO7
   4
69 IFG>8185THENPOKEG,81:POKEG+OF,9:GOTO7
   2
70 IFPEEK(G)=77THENPOKEG+22,78:POKEG+22+
   OF,1:G=G+22:GOTO67
71 GOTO66
72 IFPEEK(G-1)=32THENPOKEG-1,81:POKEG-1+
   OF,9
73 IFPEEK(G+1)=32THENPOKEG+1,81:POKEG+1+
   OF,9
74 POKEG-22,32:G=G-22:IFPEEK(G-22)=88THE
   N76
75 GOTO74
76 FORR=255TO128STEP-.9:POKEVA,R:NEXT:PO
   KEVA,0
77 X=INT(RND(1)*18)+1:DY=1:Y=1:DX=1:IFX=
   >11THENDX=-DX
78 IFX=<12THENDX=+DX
79 IFDX=>50THENDX=1
80 GOSUB25:FORR=1TO750:POKEC,105:GOTO105
85 POKEBD,91:POKEBD,90:DX=+DX:DY=-DY:POK
   EBD,91:GOSUB25:POKEBD,32:GOTO105
90 FORR=1TO15:FORW=250TO240STEP-1:POKEVA
   ,W:NEXT:FORW=240TO250:POKEVA,W:NEXT:P
```

```
   OKEVA,Ø:NEXT:P1=P1+1
91 FORR=1TO1ØØ:SC=SC+1Ø:POKEVA,245:GOSUB
   25:FORW=1TO1Ø:NEXT:POKEVA,Ø:NEXT:GOTO
   4
95 IFL>1THENP1=1:FORR=ØTO255:POKEC,R:POK
   EVA,INT(RND(Ø)*128+127):NEXT:POKEC,1Ø
   5:POKEVA,Ø:GOTO55
96 IFL<3THENL=L+K:POKE7767,147:POKE7789,
   133:POKE7811,131:POKE7833,143:POKE785
   5,142:POKE7877,132:X=17
97 DX=1:Y=1:DY=1:POKEC,47:FORR=1TO28:REA
   DA:POKEVA,A:POKEC,A:FORW=1TO5Ø:NEXTW,
   R
98 POKEVA,Ø:RESTORE:POKEC,1Ø5:DX=-DX:DY=
   +DY:GOTO1Ø4
1ØØ IFM<77Ø3THEN1Ø4
1Ø1 POKEM,85:POKEM+1,88:POKEM+2,73:POKEM
    +3,32:M=M-1:GOTO1Ø4
1Ø2 IFM>7718THEN1Ø4
1Ø3 POKEM+2,85:POKEM+3,88:POKEM+4,73:POK
    EM+1,32:M=M+1
1Ø4 POKEBO,32:BO=77Ø3+X+22*Y:POKEBO,42
1Ø5 X=X+DX:IFX=ØORX=19THENDX=-DX:POKEVA,
    24Ø
1Ø6 Y=Y+DY:IFY=-1THEN95
1Ø7 IFY=22THEN9Ø
1Ø8 IFDX=ØTHENDX=1
1Ø9 POKEVA,Ø:POKEVD,Ø:BD=77Ø3+X+22*Y
11Ø IFPEEK(BD)=32THEN116
111 POKEBO,32:IFPEEK(BD)=JTHEN65
112 IFPEEK(BD)=81THENPOKEVA,238:POKEVD,2
    38:SC=SC+5Ø:GOTO85
113 IFPEEK(BD)=65THENSC=SC+2ØØ:FORR=128T
    O255STEP2:POKE36875,R:NEXT:POKE36875
    ,Ø:GOTO85
114 IFPEEK(BD)=85ORPEEK(BD)=73THENPOKEVA
    ,14Ø:SC=SC+75:GOSUB25:DX=+DX:DY=-DY:
    GOTO1Ø5
115 IFPEEK(BD)=88THENPOKEVA,212:SC=SC+25
    :GOSUB25:DX=Ø:DY=-DY:GOTO1Ø5
116 IFPEEK(197)=31THEN1ØØ
117 IFPEEK(197)=23THEN1Ø2
```

```
118 GOTO1Ø4
125 DATA 218,218,218,218,16Ø,148,136,149
    ,142,132,133,146,13Ø,137,146,132,16Ø
    ,218,218,218
126 DATA218,218,218,218,218,218,218,134,
    137,146,147,148,16Ø,147,129,148,133,
    14Ø,14Ø,137
127 DATA148,133,218,218,218,218,147,131,
    143,146,133,16Ø,16Ø,16Ø,16Ø,16Ø,16Ø,
    136,137,16Ø
128 DATA16Ø,16Ø,16Ø,16Ø,16Ø,16Ø
```

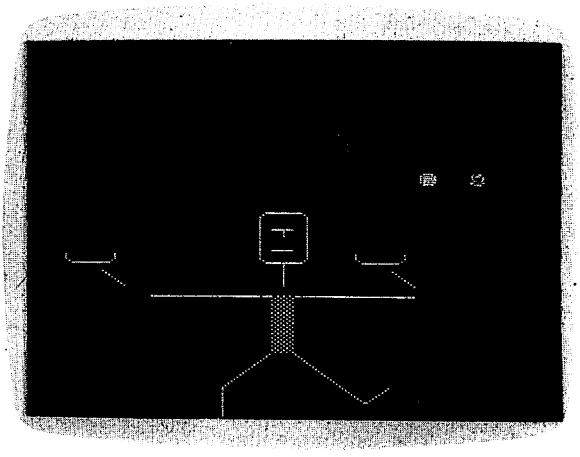
Juggler

Doug Ferguson

This game is a challenge to your dexterity. You may choose to juggle two or three balls at once.

"Juggler" is a fast-action game I wrote when the idea of programming BASIC was very new to me. It has undergone countless revisions since then, but the main loop (lines 860-970) remains what it was the first night I struggled.

The purpose of Juggler is to use the joystick to control the arms of a cartoon juggler in his attempt to keep two or three balls in the air. The juggler's hands move in tandem and can be positioned to catch the three colored balls (inside, middle, and outside). Since this game is not for the timid, it requires a joystick because only game addicts know how to work them.



A missed ball and the game is over in "Juggler."

The balls are as random as I could make them and, contrary to the opinion of novice players, cannot land at the catching stage at the same time. The most important playing

strategy is to make sure you hear the ball being caught before you move the hand toward another catch. The game can be slowed down merely by adding a few extra GOSUB 150 statements in the main loop, preferably at lines 860, 900, and 940.

The game will run on either the unexpanded or expanded VIC-20. I would rather not explain how it all works, mainly because it often follows inconsistent logic and layout. One technical footnote for those curious enough to investigate the program: instead of using zeros as values, I used a period (as in Y=.) because I read somewhere that this is faster and still zeros out the variable.

To start a game, push the joystick up or left for the three-ball or two-ball games, respectively. To repeat the same game, press the fire button, or move the stick to change games. If you want to quit, hit Q.

Program 6-2. Juggler

```
100 POKE36879,75:X=RND(-TI):Y=2:GOTO670
110 POKEV,15:POKEV-2,N:FORT=1TO10:NEXT:P
    OKEV-2,0
120 O=O+1+ABS(C)*9
130 PRINTTAB(7)"{WHT}SCORE "O"{HOME}";:R
    ETURN
140 DEFFNJ(X)=-((PEEK(37151)ANDX)=.):RET
    URN
150 IFFNJ(16)ANDU=1THENGOSUB360
160 IFFNJ(16)ANDU=2THENGOSUB420
170 POKEG+2,127:Q=PEEK(G):POKEG+2,255
180 IFQ=119ANDU=1THENGOSUB480
190 IFQ=119ANDU=.THENGOSUB420
200 RETURN
210 IFPEEK(H)<64THEN330
220 RETURN
230 IFPEEK(H+2)<64THEN340
240 RETURN
250 IFPEEK(H+4)<64THEN350
260 RETURN
270 IFPEEK(H+16)<64THEN330
280 RETURN
290 IFPEEK(H+14)<64THEN340
300 RETURN
```

```
310 IFPEEK(H+12)<64THEN350
320 RETURN
330 POKEJ%(K-A),32:D=K-1:GOSUB630:GOTO54
    0
340 POKEU%(L-B),32:D=L:GOTO540
350 POKEG%(M-C),32:D=M+1:GOSUB640:GOTO54
    0
360 POKEH+2,32:POKEH+3,32:POKEH-1,74:POK
    EH,64:POKEH+1,75
370 POKEH+4,32:POKEH+5,32:POKEH+24,32:PO
    KEH+23,77:POKEH+25,32
380 POKEH+14,32:POKEH+15,32:POKEH+11,74:
    POKEH+12,64:POKEH+13,75
390 POKEH+16,32:POKEH+17,32:POKEH+37,32:
    POKEH+35,77:POKEH+36,32
400 POKEH+187,77:POKEH+188,78:POKEH+209,
    32:POKEH+181,103:POKEH+203,122:POKEH
    +180,32:U=.
410 RETURN
420 POKEH+1,74:POKEH+2,64:POKEH+3,75:POK
    EH,32:POKEH-1,32
430 POKEH+4,32:POKEH+5,32:POKEH+25,32:PO
    KEH+23,32:POKEH+24,72
440 POKEH+13,74:POKEH+14,64:POKEH+15,75:
    POKEH+12,32:POKEH+11,32
450 POKEH+16,32:POKEH+17,32:POKEH+37,32:
    POKEH+35,32:POKEH+36,72
460 POKEH+181,103:POKEH+203,122:POKEH+18
    7,101:POKEH+209,76:POKEH+188,32:POKE
    H+180,32:U=1
470 RETURN
480 POKEH+3,74:POKEH+4,64:POKEH+5,75:POK
    EH+2,32:POKEH+1,32
490 POKEH,32:POKEH-1,32:POKEH+25,78:POKE
    H+23,32:POKEH+24,32
500 POKEH+15,74:POKEH+16,64:POKEH+17,75:
    POKEH+14,32:POKEH+13,32
510 POKEH+12,32:POKEH+11,32:POKEH+37,78:
    POKEH+35,32:POKEH+36,32
520 POKEH+181,78:POKEH+180,77:POKEH+203,
    32:POKEH+187,101:POKEH+209,76:POKEH+
    188,32:U=2
```

```
530 RETURN
540 GOSUB650:PRINTTAB(D)"{21 DOWN}{WHT}C
    RASH{HOME}";:POKEV-1,N:POKE7954-SC,1
    5:POKEH-36,34
550 FORT=15TO0STEP-1:POKEV,T:POKEV+1,PEE
    K(V+1)AND248ORT
560 FORW=1TO100:NEXT:NEXT:POKEV-1,.:POKE
    36879,75
570 PRINTTAB(7)"{YEL}{2 DOWN}GAME OVER
    {DOWN}":POKEH-14,64:GOTO590
580 PRINT"{4 SPACES}{BLK}↑{YEL} 3 BA
    LLS{DOWN}":PRINT"{4 SPACES}{BLK}←
    {YEL} 2 BALLS{DOWN}":PRINT"
    {4 SPACES}PRESS {RVS}Q{OFF} TO QUIT"
    :GOSUB140
590 GETA$:IFA$="Q"THENSYS65234
595 IFFNJ(32)ANDY<>2THENRESTORE:E=0:O=0:
    GOTO710
600 IFFNJ(16)THENCLR:Y=1:GOTO700
610 IFFNJ(4)THENCLR:GOTO700
620 GOTO590
630 FORT=38884TOT+6:POKET-CO,7:NEXT:FORT
    =38900TOT+6:POKET-CO,7:NEXT:RETURN
640 FORT=38884TOT+6:POKET-CO,6:NEXT:FORT
    =38899TOT+6:POKET-CO,6:NEXT:RETURN
650 FORT=8165+DTOT+2:POKET-SC,123:NEXT
660 RETURN
670 PRINT"{CLR}{DOWN}{6 SPACES}{YEL}VIC
    JUGGLER{WHT}"
680 PRINT"{3 DOWN}{3 RIGHT}USE JOYSTICK
    ONLY{4 DOWN}"
690 PRINT"{YEL}{4 RIGHT}CHOOSE GAME
    {DOWN}":GOTO580
700 V=36878:H=7968:G=37152:DIMJ%(18),U%(
    16),G%(13)
710 PRINT"{CLR}";:IFPEEK(36869)=192THENS
    C=3584:CO=512
720 H=7968-SC:GOSUB140
730 FORX=8015TO8025:POKEX-SC,64:NEXT:POK
    EX-6-SC,104
740 POKEH-37,93:POKEH-35,93:POKEH+7,74:P
    OKEH+9,75:POKEH+30,93:POKEH+118,102
750 POKEH-15,93:POKEH-13,93:POKEH-59,85:
```

```
    POKEH-57,73
760 POKEH-36,114:POKEH-14,82:POKEH+8,114
    :POKEH-58,64:POKEH+74,102
770 POKEH+139,78:POKEH+141,77:POKEH+160,
    78:POKEH+164,77:POKEH+96,102
780 GOSUB360
790 FORK=2TO16:READJ%(K):J%(K)=J%(K)-SC:
    POKEJ%(K)+30720+SC-CO,7:NEXT
800 FORK=2TO14:READU%(K):U%(K)=U%(K)-SC:
    NEXT
810 FORK=2TO12:READG%(K):G%(K)=G%(K)-SC:
    POKEG%(K)+30720+SC-CO,6:NEXT
820 K=INT(RND(1)*8)+2:A=1
830 L=INT(RND(1)*7)+2:B=1
840 IFY=0THENM=INT(RND(1)*6)+2:C=1
850 IFO>49+450*ABS(C)THENE=1:Y=.
860 GOSUB150
870 N=220:IFK=17THENGOSUB270:A=-1:GOSUB1
    10:K=15-E
880 IFK=1THENGOSUB210:A=1:GOSUB110:K=3+E
890 POKEJ%(K-A),32:POKEJ%(K),81:K=K+A
900 GOSUB150
910 IFL=15THENGOSUB290:B=-1:GOSUB110:L=1
    3-E
920 IFL=1THENGOSUB230:B=1:GOSUB110:L=3+E
930 POKEU%(L-B),32:POKEU%(L),81:L=L+B
940 GOSUB150
950 IFM=13THENGOSUB310:C=-1:GOSUB110:M=1
    1-E
960 IFM=1THENGOSUB250:C=1:GOSUB110:M=3+E
970 POKEG%(M-C),32:POKEG%(M),81:M=M+C:IF
    E=.THEN850
980 POKEJ%(K-3*A),32:POKEJ%(K-2*A),81
990 POKEU%(L-3*B),32:POKEU%(L-2*B),81
1010 POKEG%(M-3*C),32:POKEG%(M-2*C),81:G
     OTO860
1020 DATA 7946,7902,7858,7815,7772,7730,
     7710,7712,7714,7738,7784,7829,7874,
     7918,7962
1030 DATA7948,7904,7860,7817,7774,7754,7
     756,7758,7782,7827,7872,7916,7960
1040 DATA7950,7906,7862,7819,7798,7778,7
     802,7825,7870,7914,7958
```

Deflector

Frank J. Tyniw

This strategy game tests your mind as well as your hand/eye coordination.

Playing "Deflector" is simple. A ball bounces from side to side or from top to bottom of the screen. Pressing the left arrow key above the control key will print a slash in front of the ball's path, deflecting it 90 degrees. The F1 key will print a backslash (\). Your goal is to deflect the ball into the square targets, using as few slashes as possible to achieve the highest score.

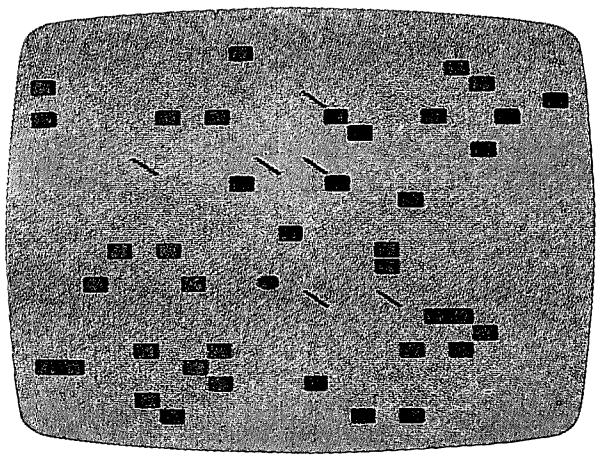
Scoring is ten points for every block hit, minus one point for every slash used and minus five points for every slash on the screen if you hit the panic button. The panic button is the British pound sign (£). If you get too many slashes on the screen or deflect yourself into a corner, hitting the panic button will remove all slashes, subtract five points per slash, and resume the game.

The subroutine at 63000 is a useful utility you may want to include in other programs. When the program starts, it asks "adjust screen? (y/n)". The screen will switch to a black border and white background, and color bars for fine tuning your set. The cursor control keys will move the entire screen up, down, left, or right to adjust for your TV.

Press D when done adjusting, and the program asks if you want instructions. Then it will ask for number of targets. The program then will select random screen locations for the targets (160-200). Lines 700-990 handle the score display and rerun lines. Lines 4300-6210 are the sound routines. This program works on the Unexpanded VIC or with the 3K cartridge.

Suggested Modification

One way to improve your programming skills is to first try to understand how a program works, and then try to modify or improve it. Why not try to make some changes in Deflector? For example, you could use programmable characters for the



"Deflector" requires quick thinking.

targets. The targets could be germs or political symbols, or instead of a ball you could use up, down, left, and right darts, arrows, anything.

Don't be afraid to experiment. But first, type in and SAVE Deflector as is. Then make the changes.

Program 6-3. Deflector

```
10 PRINT"{CLR}":TR=208:J=3:BC=36879:VO=B
   C-1:S4=BC-2:S3=BC-3:S2=BC-4:S1=BC-5
20 GOSUB63000:POKEBC,93:V=15
122 PRINT"{DOWN}INSTRUCTIONS? (Y/N)
123 GETV$:IFV$=""THEN123
125 IFV$="Y"THENGOSUB1000
130 PRINT"{CLR}"CHR$(142)
140 K=0:T=0:CL=5
142 INPUT"{DOWN}HOW MANY TARGETS";J:J=AB
    S(J)
144 IFJ>506THENPRINT"TOO MANY!":GOTO142
146 IFJ<10ORJ>200THENPRINT"{DOWN}BRAVE,
    AREN'T YOU?"
155 FORI=1TO1000:NEXT:PRINT"{CLR}":GOSUB
    7000
157 SS=7680:SR=38400
```

```
160 FORI=1TOJ
170 A=INT(506*RND(1))
180 IFPEEK(SS+A)=TRTHEN170
185 POKES2,0:POKES3,0
190 POKESS+A,TR:POKESR+A,6:GOSUB4300
200 NEXTI
205 POKES2,0:POKES3,0
210 A=INT(506*RND(1))
230 U=A+SS
240 DI=1:IFRND(1)>.5THENDI=-1
300 GETX$
310 IFX$<>""THEN600
320 NE=U+DI
330 IFABS(DI)=1THEN430
340 IFDI>0THEN380
350 IFNE<SSTHENDI=-DI:GOSUB6000:GOTO320
355 A=NE
360 IFPEEK(A)=77THENDI=-1:NE=NE-1
370 IFPEEK(A)=78THENDI=1:NE=NE+1
375 GOTO530
380 IFNE>SS+506THENDI=-DI:GOSUB6000:GOTO
    320
390 A=NE
400 IFPEEK(A)=77THENDI=1:NE=NE+1
410 IFPEEK(A)=78THENDI=-1:NE=NE-1
420 GOTO530
430 IFDI>0THEN490
440 IFNE-22*INT(NE/22)=1THENDI=-DI:GOSUB
    6200:GOTO320
450 A=NE
460 IFPEEK(A)=77THENDI=-22:NE=NE+DI
470 IFPEEK(A)=78THENDI=22:NE=NE+DI
480 GOTO530
490 IFNE-22*INT(NE/22)=2THENDI=-DI:GOSUB
    6200:GOTO320
500 A=NE
510 IFPEEK(A)=77THENDI=22:NE=NE+DI
520 IFPEEK(A)=78THENDI=-22:NE=NE+DI
530 POKEU,32
540 IFPEEK(NE)=32THENPOKENE,81:U=NE:GOTO
    300
550 IFPEEK(NE)=TRTHENK=K+1:SC=SC+10
```

```
552 IFPEEK(NE)=TRTHENGOSUB5000
555 POKENE,170:U=NE:FORI=1TO25:NEXT
560 IFK=JTHEN700
570 GOTO300
600 IFX$="←"THENA=78:GOTO630
610 IFX$="{F1}"THENA=77:GOTO630
615 IFX$="≒"THENGOSUB2000
616 IFX$="Q"THEN990
620 GOTO320
625 GOSUB4600
630 IFPEEK(U+DI)=32THENPOKEU+DI,A:SL=SL+
    1:SC=SC-1
640 GOTO300
700 REM
712 PRINT"{CLR}":POKEBC,125
715 IFSC>HSTHENHS=SC:PRINT"{RVS} NEW ";
716 PRINT"HIGH SCORE:"HS"{LEFT} "
720 PRINT"{DOWN}IT TOOK"SL"SLASHES
730 PRINT"{DOWN}TO HIT"J"TARGETS"
905 PRINT"{DOWN}YOUR SCORE";SC
910 PRINT"{2 DOWN}TRY AGAIN?(Y OR N)"
920 GETW$:IFW$=""THEN920
925 IFW$="N"THEN990
926 SL=0:SC=0
930 PRINT:PRINT"HOW MANY TARGETS";:INPUT
    J
940 J=ABS(INT(J))
960 PRINT"{CLR}":POKEBC,93:GOSUB7000:K=0
    :T=0:GOTO155
990 PRINT"{CLR}":POKEBC,27:END
1000 PRINT"{CLR}"
1010 PRINTCHR$(14);"{2 SPACES}THE OBJECT
      OF THIS
1015 PRINT"{DOWN}GAME IS TO DEFLECT THE
1020 PRINT"{DOWN}{UP}BALL INTO THE BOXES
      BY
1025 PRINT"{DOWN}{UP}USING ← AND F1 KEY
     S
1030 PRINT"{DOWN}TO PRINT DIAGONALS IN
1035 PRINT"{DOWN}ITS PATH. IF YOU GET
1040 PRINT"{DOWN}STUCK IN A LOOP USE
1045 PRINT"{DOWN}THE ≒ KEY AS A PANIC
```

```
1050 PRINT"{DOWN}BUTTON.
1085 PRINT"{3 DOWN}HIT ANY KEY...
1090 GETB$:IFB$=""THEN1090
1100 PRINT"{CLR}{DOWN}SCORING IS 10 POIN
     TS
1110 PRINT"{DOWN}PER BLOCK HIT, ONE
1120 PRINT"{DOWN}POINT SUBTRACTED FOR
1130 PRINT"{DOWN}EVERY SLASH YOU LAY,
1140 PRINT"{DOWN}AND -5 FOR EVERY SLASH
1150 PRINT"ON THE SCREEN IF YOU
1160 PRINT"{DOWN}HIT THE PANIC BUTTON.
1170 PRINT"{4 DOWN}HIT ANY KEY TO START.
     ."
1180 GETA$:IFA$=""THEN1180
1190 RETURN
2000 FORI=SSTOSS+506
2010 IFPEEK(I)<>77ANDPEEK(I)<>78THEN2030
2020 GOSUB4300:POKES2,0:POKES3,0:SC=SC-5
     :POKEI,32
2030 NEXTI
2040 RETURN
4300 SO=INT(RND(1)*100)+129
4310 POKEVO,V:POKES3,SO:POKES2,SO:FORT1=
     1TO35:NEXTT1:RETURN
5000 POKEVO,V:FORS=128TO250STEP10
5010 POKES4,S
5020 NEXTS
5030 POKEVO,0:POKES4,0:RETURN
6000 POKEVO,V:POKES3,250:FORII=1TO25:NEX
     TII:POKES3,0:POKEVO,0:RETURN
6200 POKEVO,V:POKES3,245:FORII=1TO25:NEX
     TII:POKES3,0:POKEVO,0
6210 RETURN
7000 FORI=38400TO38905:POKEI,6:NEXT:RETU
     RN
63000 REM SCREEN ADJUSTMENT
63010 POKE36879,24:PRINT"{CLR}":H=PEEK(3
      6864):V=PEEK(36865)
63020 PRINT"ADJUST SCREEN? (Y/N)"
63030 GETA$:IFA$=""THEN63030
63040 IFA$="Y"GOTO63060
63050 PRINT"{CLR}{BLK}";:RETURN
```

```
63060 PRINT"{2 DOWN}USE THE CRSR KEYS TO
63070 PRINT"{DOWN}MOVE SCREEN AND THE
63080 PRINT"{DOWN}LETTER D WHEN DONE
      {2 DOWN}
63081 PRINT"{RVS}{RED}RED{18 SPACES}"
63082 PRINT"{RVS}{CYN}CYAN{17 SPACES}"
63083 PRINT"{RVS}{PUR}PURPLE{15 SPACES}"
63084 PRINT"{RVS}{GRN}GREEN{16 SPACES}"
63085 PRINT"{RVS}{BLU}BLUE{17 SPACES}"
63086 PRINT"{RVS}{YEL}YELLOW{15 SPACES}"
63090 GETA$:IFA$=""THEN63090
63100 IFA$="D"THENPRINT"{CLR}{BLK}";:RET
      URN
63110 IFA$="{UP}"THENV=V-1:IFV<0THENV=0
63120 IFA$="{DOWN}"THENV=V+1:IFV>40THENV
      =40
63130 IFA$="{LEFT}"THENH=H-1:IFH<0THENH=
      0
63140 IFA$="{RIGHT}"THENH=H+1:IFH>17THEN
      H=17
63150 POKE36864,H:POKE36865,V:GOTO63090
```

Jumping Jack

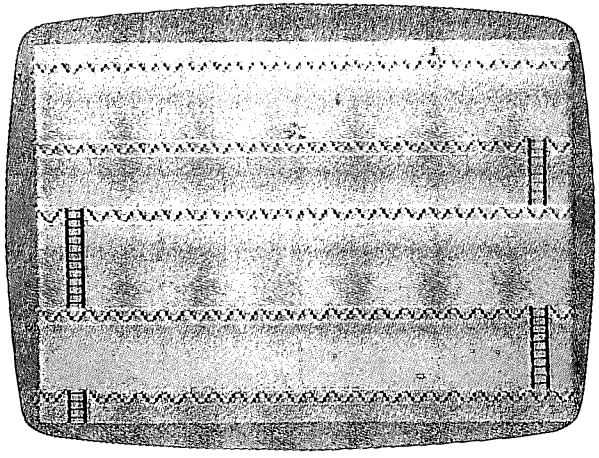
Paul Burger

"Jumping Jack," for the unexpanded VIC, is a challenging game that makes full use of your computer's color and sound capabilities.

Jack is running across platforms and climbing down ladders to get to the bottom of the screen. Sounds easy enough, right?

There's just one problem: these platforms are not very sturdy at all, and at any time they can collapse in certain places. You must be ready to press the space bar causing Jack to jump. If your timing is right, Jack will clear the hole and land safely on his feet. If not, Jack will fall into the collapsed section of the platform.

If you are not quite quick enough on the space bar, you still have a chance to clear the hole. Here's how: If the space bar is pressed immediately after Jack gets over the hole, you can make a saving jump. However, Jack must be over the hole while in the air to get points for jumping the hole, so no



Make Jack jump just at the right time.

points are scored for using a saving jump to get over a hole. This method can also be used to jump two holes in a row. Simply make a saving jump as described above for the first hole, and Jack will fly over the second hole (this scores points only for the second hole, however).

Program 6-4. Jumping Jack

```
0 M=3:T=150:D=5:X=25:P=61:POKE55,160:POK
  E56,29:S=36876:POKE36878,15:GOTO10012
1 C=27:F%=5:FORI=7680TO8185:POKEI,59:NEX
  T
2 FORI=7702TO7723:POKEI,53:NEXT:FORI=781
  2TO7833:POKEI,53:NEXT:FORI=7900TO7921:
  POKEI,53:NEXT
3 POKE36879,C:FORI=8032TO8053:POKEI,53:N
  EXT:FORI=8142TO8163:POKEI,53:NEXT
4 FORI=38400TO38884+21:POKEI,4:NEXT
5 FORI=38488TO38510+21:POKEI,F%:NEXT
6 FORI=38576TO38598+21:POKEI,F%:NEXT
7 FORI=38708TO38730+21:POKEI,F%:NEXT
8 FORI=38818TO38840+21:POKEI,F%:NEXT:GOS
  UB10020:FORI=1TO1000:NEXT
9 I=7790
10 IFI/2=INT(I/2)THENPOKEI-1,59:GOSUB110
11 IFI/2=INT(I/2)THENPOKEI,55:FORJ=1TOT:
   NEXT:GOTO14
13 POKEI-1,59:POKEI,56:FORJ=1TOT:NEXT:B=
   7812:GOSUB510
14 IFPEEK(197)=32THENGOSUB20
15 IFPEEK(I+22)=54THENPOKEI,59:GOTO30
16 IFPEEK(I+22)=60THEN500
17 I=I+1:IFI>7811THENI=7790:POKE7811,59
18 GOTO10
20 I=I-21:POKEI+21,59
21 IFPEEK(I+22)<>59ORPEEK(I+44)<>53THENS
   C=SC+X:POKEI-22,P:GOSUB112:POKEI-22,5
   9
23 POKEI,55:FORJ=1TOT:NEXT:I=I+23:IFI>78
   11THENI=7790:POKE7811,59
24 POKE7789,59:POKE7790,59
25 FORJ=1TOT:NEXT:POKEI-23,59:POKEI,55:R
   ETURN
```

```
30 I=7898
31 IFI/2=INT(I/2)THENPOKEI+1,59:GOSUB110
32 IFI/2=INT(I/2)THENPOKEI,58:FORJ=1TOT:
   NEXT:GOTO34
33 POKEI+1,59:POKEI,57:FORJ=1TOT:NEXT:B=
   7900:GOSUB510
34 IFPEEK(197)=32THENGOSUB400
35 IFPEEK(I+22)=54THENPOKEI,59:GOTO50
36 IFPEEK(I+22)=60THEN500
37 I=I-1:IFI<7878THENI=7898:POKE7878,59
38 GOTO31
40 I=I-23:POKEI+23,59
41 IFPEEK(I+22)<>59ORPEEK(I+44)<>53THENS
   C=SC+X:POKEI-22,P:GOSUB112:POKEI-22,5
   9
43 POKEI,58:FORJ=1TOT:NEXT:I=I+21:IFI<78
   78THENI=7898:POKE7878,59
44 POKE7856,59:POKE7855,59
45 FORJ=1TOT:NEXT:POKEI-21,59:POKEI,58:R
   ETURN
50 I=8010
51 IFI/2=INT(I/2)THENPOKEI-1,59:GOSUB110
52 IFI/2=INT(I/2)THENPOKEI,55:FORJ=1TOT:
   NEXT:GOTO54
53 POKEI-1,59:POKEI,56:FORJ=1TOT:NEXT:B=
   8032:GOSUB510
54 IFPEEK(197)=32THENGOSUB600
55 IFPEEK(I+22)=54THENPOKEI,59:GOTO70
56 IFPEEK(I+22)=60THEN500
57 I=I+1:IFI>8031THENI=8010:POKE8031,59
58 GOTO51
60 I=I-21:POKEI+21,59:IFPEEK(I)<>59THENS
   C=SC+300
61 IFPEEK(I+22)<>59ORPEEK(I+44)<>53THENS
   C=SC+X:POKEI-22,P:GOSUB112:POKEI-22,5
   9
63 POKEI,55:FORJ=1TOT:NEXT:I=I+23:IFI>80
   31THENI=8010:POKE8031,59
64 POKE8009,59:POKE8010,59
65 FORJ=1TOT:NEXT:POKEI-23,59:POKEI,55:R
   ETURN
70 I=8140
```

```
71 IFI/2=INT(I/2)THENPOKEI+1,59:GOSUB110
72 IFI/2=INT(I/2)THENPOKEI,58:FORJ=1TOT:
   NEXT:GOTO74
73 POKEI+1,59:POKEI,57:FORJ=1TOT:NEXT:B=
   8142:GOSUB510
74 IFPEEK(197)=32THENGOSUB80
75 IFPEEK(I+22)=54THENPOKEI,59:GOTO100
76 IFPEEK(I+22)=60THEN500
77 I=I-1:IFI<8120THENI=8140:POKE8120,59
78 GOTO71
80 I=I-23:POKEI+23,59
81 IFPEEK(I+22)<>59ORPEEK(I+44)<>53THENS
   C=SC+X:POKEI-22,P:GOSUB112:POKEI-22,5
   9
83 POKEI,58:FORJ=1TOT:NEXT:I=I+21:IFI<81
   20THENI=8140:POKE8120,59
84 POKE8098,59:POKE8097,59
85 FORJ=1TOT:NEXT:POKEI-21,59:POKEI,58:R
   ETURN
100 P=P+1:IFP=64THENP=61
101 D=D-1:T=T-50
102 X=X+50:IFX>125THENX=25:D=8:T=150:C=2
    7:F%=5
103 IFX=75THENC=232:F%=0
104 IFX=125THENC=8:F%=7
105 GOTO2
110 POKES,140:FORY=1TO10:NEXT:POKES,0:RE
    TURN
111 POKES+1,190:FORY=1TO25:NEXT:POKES+1,
    0:RETURN
112 FORO=1TO15:POKES,200+O:NEXTO:POKES,0
    :RETURN
113 FORO=20TO0STEP-1:POKES,230+O:FORY=1T
    O25:NEXTY,O:POKES,0:RETURN
500 GOSUB113:M=M-1:IFM=0THEN502
501 P=61:X=25:D=6:C=27:T=150:F%=5:POKEI,
    59:GOTO2
502 POKE36869,240:PRINTCHR$(147);SPC(225
    );"GAME OVER!":PRINT:PRINT"YOUR SCOR
    E WAS ";SC
503 PRINT:PRINT"PLAY AGAIN?"
504 K=PEEK(197):IFK=32ORK=64THEN504
```

```
505 IFK=11THENRUN
506 END
510 IFINT(RND(1)*D)+1<>1THENRETURN
511 L=INT(RND(1)*21)+1:IFL=20ORL=1THEN51
    1
512 POKEB+L,60:GOSUB111:RETURN
10000 DATA255,129,66,66,36,36,24,255
10002 DATA66,126,66,66,66,126,66,66
10003 DATA12,8,13,62,44,12,18,33
10004 DATA24,16,24,24,24,16,16,24
10005 DATA24,8,24,24,24,8,8,24
10006 DATA24,8,88,62,26,24,36,66
10007 DATA0,0,0,0,0,0,0,0
10008 DATA129,66,66,66,98,34,34,34
10009 DATA27,10,27,17,27,0,0,0
10010 DATA59,10,11,9,11,0,0,0
10011 DATA91,74,91,81,91,0,0,0
10012 RESTORE:FORI=7592TO7679:READA:POKE
      I,A:NEXT
10015 POKE36869,255
10016 GOTO1
10020 FORI=7832TO7898STEP22:POKEI,54:NEX
      T:FORI=7901TO8011STEP22:POKEI,54:N
      EXT
10021 FORI=8052TO8140STEP22:POKEI,54:NEX
      T:FORI=38552TO38618STEP22:POKEI,6:
      NEXT
10022 FORI=38621TO38731STEP22:POKEI,6:NE
      XT:FORI=38772TO38860STEP22:POKEI,6
      :NEXT
10023 POKE8143,54:POKE8165,54:POKE38863,
      6:POKE38885,6:RETURN
```

Skydiver

Alan Crossley

"Skydiver" is an arcade-style game for the Unexpanded VIC. The game will test your timing and your ability to calculate the wind's effect on the skydiver.

The sky is clear, the land is flat, but the winds are variable. You are determined to learn to skydive. Not only do you wish to learn to skydive, but you wish to become an expert. Safe jumps are always important, but you also want to be able to land exactly where you have preselected.

Playing Skydiver

It is up to you to decide which landing pad to try for. There are three pads to choose from — labeled 2X, 5X, and 10X. Each pad is more difficult to land on than the previous one and therefore scores more points. If you land successfully, you are rewarded with bonus points. If you miss, you lose one of your three skydivers.

At 5000 points you are awarded an extra skydiver. Each time you make two successful landings, the game's difficulty level increases, and the bonus value goes up 50 points.

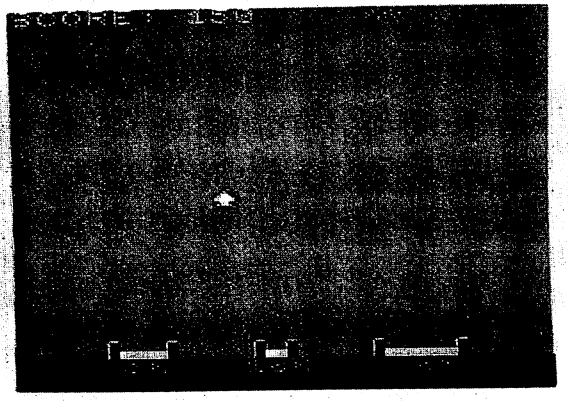
Controlling the Dive

Press the joystick button to clear the title screen. Next, pick which landing pad to try for by manipulating the joystick to indicate your selection. Don't let the countdown timer run out because then you will have to land wherever the computer decides. Once you have selected the landing pad, press the joystick button to inform the computer of your choice and to start the dive.

Take note of the wind speed and direction; you must take it into account when deciding when to jump. The airplane will emerge from the left side of the screen. If you do not dive by the time the airplane reaches the right side of the screen, it will reappear on the left.

To dive, simply press the joystick button and watch your

diver. At first the diver will be in free fall (parachute not opened), but have no fear, the chute will open in time, and the diver will float down and onto the landing pad. If you miscalculated the time to jump, the joystick can be used to make minor adjustments to the descent.



In "Skydiver," you must consider the wind if you plan to land on the pad.

Program 6-5: Skydiver

```
8 P1=37151:P2=37152:P3=37154:T7=30720:NE=
   2
10 PRINT"{CLR}":GOTO6010Ø
12 A(1)=7772:A(2)=7778:A(3)=7782
14 SC=Ø:BO=1ØØ:BA=3:WA=7:W=8:EX=Ø
16 PRINT"{HOME}SCORE: ØØØ{3 SPACES}"
20 GOSUB5ØØØ
21 PRINT"{HOME}";TAB(14);"{3 SPACES}
    {3 LEFT}";A$
22 CT=21+INT(RND(TI)*20+1):CC=Ø
30 A=Ø
100 PRINT"{HOME}{DOWN}";TAB(A);"{OFF} {RV
    S}{*}{2 LEFT}{DOWN}{OFF} {RVS}{2 I}":
    A=A+1
102 CC=CC+1:IFCC>=CT ANDA<21ANDA>1THEN120
```

```
105 IFA>=20THENPRINT"{HOME}{DOWN}";TAB(A)
    ;" {DOWN}{LEFT}{2 SPACES}":A=0
110 GOSUB 1000:IFR<>128THENFORI=1TO50:NEX
    T:GOTO100
112 IFA>20ORA<1THEN100
120 PRINT"{HOME}{DOWN}";TAB(A);" {LEFT}
    {DOWN}{2 SPACES}"
200 B=7747+A:T=INT(RND(TI)*8+1):T=T+1
210 CH=46:C=0
212 C=C+1
220 POKEB,CH:POKEB,32:B=B+22
230 POKEB+T7,4:POKEB,CH:IFC<TTHENFORY=1TO
    80:NEXT:GOTO212
300 CH=65
305 POKEB+T7,4:POKEB,CH
310 GOSUB1000:C=C+1
320 IFC>=WTHENIFR=4THENPOKEB,32:B=B-1:POK
    EB+T7,4:POKEB,CH:C=0
330 IFC>=WTHENIFR=8THENPOKEB,32:B=B+1:POK
    EB+T7,4:POKEB,CH:C=0
340 D=D+1
350 IFD>=WATHENPOKEB,32:B=B+22
355 IFD>=WATHEND=0:IFPEEK(B)<>32THEN3000
360 POKEB+T7,4:POKEB,CH
370 WI=WI+1
390 IFWI>=SANDDI=-1THENPOKEB,32:B=B+DI:PO
    KEB+T7,4:WI=0
395 IFWI>=SANDDI=1THENPOKEB,32:B=B+DI:POK
    EB+T7,4:WI=0
400 IFPEEK(B)<>32ANDPEEK(B)<>CHTHEN3000
410 POKEB,CH
500 GOTO300
1000 POKEP3,127:X=NOTPEEK(P1)AND60-((PEEK
     (P2)AND128)=0):POKEP3,255
1010 R=-((XAND4)=4)-2*((XAND8)=8)-4*((XAN
     D16)=16)-8*((XAND1)=1)-128*((XAND32
     )=32)
1020 RETURN
3000 POKEB+T7,4
3010 IFP=1THENIFB=8124ORB=8125THENPOKEB,C
     H:GOTO4000
3020 IFP=2THENIFB=8130THENPOKEB,CH:GOTO4050
```

```
3030 IFP=3THENIFB=8135ORB=8136ORB=8137THE
     NPOKEB,CH:GOTO4100
3040 POKEB,170:GOTO12000
4000 PRINT"{HOME}{8 DOWN}{5 RIGHT}5 X";BO
     ;"=";5*BO
4010 VA=5*BO:GOTO10000
4050 PRINT"{HOME}{8 DOWN}{4 RIGHT}10 X";B
     O;"=";10*BO
4060 VA=10*BO:GOTO10000
4100 PRINT"{HOME}{8 DOWN}{5 RIGHT}2 X";BO
     ;"=";2*BO
4110 VA=2*BO:GOTO10000
5000 PRINT"{HOME}{19 DOWN}"
5020 PRINT"{3 SPACES}{BLK}{L}{YEL}
     {2 I}{BLK}{J}{2 SPACES}{BLK}
     {L}{YEL}{I}{BLK}{J}{2 SPACES}
     {L}{YEL}{3 I}{BLK}{J}{BLU}"
5030 PRINT"{RVS}{4 SPACES}5X{3 SPACES}10X
     {4 SPACES}2X{4 SPACES}{OFF}";
5040 FORI=8164TO8185:POKEI+T7,6:POKEI,160
     :NEXT
5041 IFBA<=0THEN60000
5042 PRINT"{HOME}";TAB(17);"{4 SPACES}"
5043 IFBA>1THENPRINT"{HOME}";TAB(18);:FOR
     I=1TOBA-1:PRINT"A";:NEXT:PRINT
5045 SS=INT(RND(TI)*3+1):DI=INT(RND(TI)*2
     +1):IFDI=2THENDI=-1
5048 IFSS=1THENS=5:SD=15
5049 IFSS=2THENS=10:SD=10
5050 IFSS=3THENS=15:SD=5
5052 D$="RIGHT":IFDI=-1THEND$="LEFT"
5053 IFNJ>=NETHENBO=BO+50:PRINT"{HOME}
     {7 DOWN}BONUS ADVANCE TO{RED}";BO:P
     RINT"{BLU}"
5054 IFNJ>=NETHENIFWA>4THENWA=WA-1
5055 IFNJ>=NETHENNJ=0:IFW<18THENW=W+1
5056 PRINT"{HOME}{2 DOWN}";TAB(3);"WIND:"
     ;SD"TO ";D$:GOSUB50000
5057 FORK=1TO1000:NEXT:PRINT"{HOME}{DOWN}
     {20 SPACES}"
5060 FORI=7724TO8141STEP22:POKEI,96:POKEI
     +21,96:NEXT
```

```
5070 RETURN
10000 JM=JM+1:NJ=NJ+1
10100 FORP=1TOVA/10:PRINT"{HOME}{BLU}SCOR
      E:{RED}";SC:POKES3,0:SC=SC+10:NEXT
      :PRINT"{HOME}{BLU}SCORE:{RED}";SC
10110 FORID=1TO1000:NEXT:IFEX=0ANDSC>=500
      0THENBA=BA+1:EX=1
10120 PRINT"{HOME}{8 DOWN}{21 SPACES}":GO
      TO20
12000 PRINT"{HOME}{8 DOWN}{3 RIGHT}SORRY
      NO BONUS!":BA=BA-1
12010 FORID=1TO500:NEXT
12120 POKEB,32:PRINT"{HOME}{8 DOWN}
      {21 SPACES}":GOTO20
50000 REM SELECTION
50010 PRINT"{HOME}{4 DOWN}{RED}{4 SPACES}
      5X{3 SPACES}10X{2 SPACES}2X{BLU}":
      P=1:Z=P
50020 TI$="000000"
50025 POKEA(P),PEEK(A(P))+128AND255
50027 TM=15-VAL(TI$)
50028 PRINT"{HOME}{12 DOWN}";TAB(10);"
      {2 SPACES}{3 LEFT}";TM:IFTM=0THENP
      =INT(RND(TI)*3+1):GOTO50100
50030 GOSUB1000:IFR=4ORR=8THENX=PEEK(A(P)
      ):IFX>129THENPOKEA(P),X+128AND255
50035 IFR>127THEN50100
50040 GOSUB1000:IFR=4THENP=P-1:IFP<1THENP
      =3
50045 IFR=8THENP=P+1:IFP>3THENP=1
50050 FORID=1TO100:NEXT:GOTO50025
50100 A$="2X":IFP=1THENA$="5X"
50110 IFP=2THENA$="10X"
50120 PRINT"{DOWN} YOU MUST LAND ON ";A$
50140 FORID=1TO1500:NEXT
50150 PRINT"{HOME}":FORID=1TO18:PRINT"
      {21 SPACES}":NEXT:RETURN
60000 PRINT"{HOME}{7 DOWN}{6 RIGHT}GAME O
      VER":IFSC>HITHENHI=SC
60010 FORHD=1TO2000:NEXT
60020 PRINT"{HOME}{7 DOWN}{6 RIGHT}
      {9 SPACES}"
```

```
60100 POKE36879,188:PRINT"{HOME}{2 DOWN}
       {6 SPACES}{RED}SKY DIVER"
60110 PRINT:PRINT:PRINT:PRINT
60120 PRINT" {PUR}USE JOYSTICK TO PLAY"
60130 PRINT:PRINT:PRINT:PRINT
60140 PRINT"{6 SPACES}{GRN}HIGH SCORE
       {RED}":PRINT
60150 XX=LEN(STR$(HI))/2:PRINTTAB(10-XX);
       HI
60160 GOSUB1000:IFR=0THEN60160
60170 PRINT"{HOME}":FORK=1TO18:PRINT"
       {21 SPACES}":NEXT:GOTO12
```

The Hawkmen Of Dindrin

Esteban V. Aguilar, Jr.

Fly down through the dangerous skies of the planet Dindrin to collect golden stones. Retrieve enough of them and you can win the game, but beware of the floaters, skimmers, and lizards.

There's a strange planet named Dindrin where multicolored floaters and a giant sky skimmer drift through the daytime skies. On the surface of the planet, vicious land hunters come up from the ground and set polished, golden stones in the sun. It's a form of worship too obscure, too alien to describe.

Suddenly a strange looking hawk-like creature dives down and snatches a stone. With this program and your VIC, you can experience what it's like to be one of the Hawkmen of Dindrin.

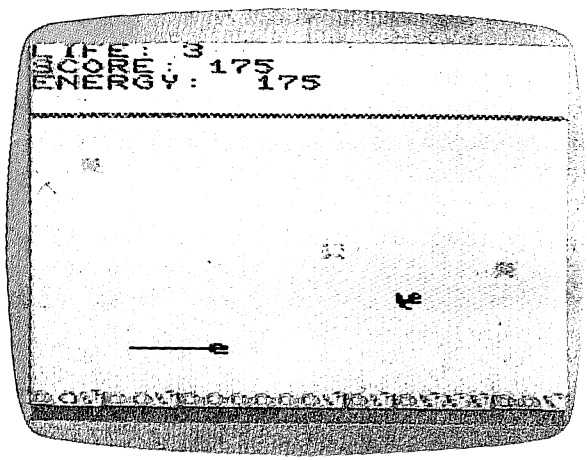
Flying the Skies of Dindrin

When the game begins, you (the hawkman) start off on the upper-left corner of the screen under the cloud layer. You will move across the screen and move lower after every line.

Maneuvering is accomplished with your joystick. You may move backwards by pulling left on the joystick. Whenever you want to go into a dive or fly upwards, you must pull down or up (respectively) on the joystick. One thing to keep in mind when ascending or descending is that you will move diagonally rather than straight up or down. The winds are powerful on Dindrin. Flying is an art form there.

Once in a while, an obstacle such as the sky skimmer (moving above the surface) or a floater (dominating the skies) will get in your way. When this happens, you can press the red button on the joystick to extend the hawkman's tongue to snap up its prey. You can also do this to obtain points and energy. The skimmer is worth 5 points and 10 energy credits. The floaters are worth 15 points and 20 energy credits.

There are a couple of things to consider before playing the game. As time passes, you will lose energy. If your energy gets too low, the screen border will disappear. If your energy runs out, the game will end. If you have sufficient energy, the border will remain on screen. Second, when you're flying don't run into anything or you'll lose one of your lives. Third, when the game starts after the program is loaded from tape, press the stop button on the cassette player. Otherwise, the joystick will not respond to your commands. To win, you must obtain 10,000 points.



Watch out for the sky skimmer in "Hawkmen of Dindrin."

The Programs

The Hawkmen of Dindrin is divided into two programs. The first program makes custom characters and automatically loads the main game program. Be sure to SAVE Program 6-7 immediately following Program 6-6 on the same tape.

Program 6-6: Hawkmen, Part 1

```
10 PRINT"{CLR}":POKE36879,93
20 PRINT"{3 SPACES}{RVS}HAWKMEN OF DINDRIN
   {OFF}"
```

```
50000 X=PEEK(56)-2:POKE52,X:POKE56,X:POK
      E51,PEEK(55):CLR
50010 CS=256*PEEK(52)+PEEK(51)
50020 FORI=CSTOCS+511:POKEI,PEEK(I+32768
      -CS):NEXT
50030 READX:IFX<0THEN50090
50040 FORI=XTOX+7:READJ:POKEI,J:NEXT
50050 GOTO50030
50090 DATA7512,0,28,54,255,48,30,0,0
50091 DATA7504,35,85,91,192,109,41,101,2
      6
50092 DATA7496,255,126,60,60,60,60,255,2
      55
50093 DATA7168,134,205,207,252,255,96,96
      ,24
50094 DATA7432,0,0,0,255,0,0,0,0
50095 DATA7440,0,90,60,90,126,24,36,66
50096 DATA7448,0,24,36,66,66,66,36,255
50097 DATA7456,24,135,133,199,71,103,103
      ,255
50098 DATA7464,24,207,141,239,255,255,25
      5,255
50099 DATA7472,0,0,0,170,85,0,0,0
50100 DATA7488,255,255,255,255,255,255,2
      55,255
50101 DATA-1
63000 POKE198,5:POKE631,78:POKE632,69:PO
      KE633,87:POKE634,13:POKE635,131:END
```

Program 6-7: Hawkmen, Part 2

```
6 TI$="000000"
7 POKE36869,255
8 CLR:S1=36876:S2=36875:S3=36874:S4=3687
  7
9 E=200:FC=38554
10 PRINT"{CLR}":POKE36879,122:SC=0:Y=781
   3:V=36878:YC=38533:SK=8080:MKC=38800:
   TN=4:FL=7834
20 PRINT"{HOME}{4 DOWN}&&&&&&&&&&&&&&&&&
   &&&&&";
21 PRINT"{HOME}{DOWN}SCORE:":PRINT"ENERGY:"
```

```
22 FORA=ØTO21:POKE8164+A,4Ø:POKE38884+A,
   6:NEXTA
3Ø FORA=ØTO21:POKE8142+A,35:POKE38862+A,
   5:NEXTA
35 PRINT"{HOME}{18 DOWN} !!!+"
4Ø POKE36878,15
5Ø POKEYC,Ø:POKEY,Ø
51 J=PEEK(37151):E=E-1:GOSUB6ØØ
52 IFJ=94THENGOSUB1ØØ
53 IFJ=11ØTHENQ=-1
54 IFJ=126THENQ=1
55 IFJ=118THENQ=23:POKES2,13Ø
56 IFJ=122THENQ=-21:POKES2,21Ø
57 POKEY,32:POKES2,Ø:IFY+Q<7790ORY+Q>=81
   63THEN5ØØ
58 PRINT"{HOME}LIFE:";TN:PRINT"{HOME}
   {DOWN}{6 RIGHT}";SC:PRINT"{7 RIGHT}
   {7 SPACES}{6 LEFT}";E:IFE<=ØTHEN1ØØØ
59 IFPEEK(Y+Q)=35THEN9Ø
6Ø IFPEEK(Y+Q)=32THEN75
71 IFPEEK(Y+Q)<>32THEN5ØØ
75 Y=Y+Q:YC=YC+Q:POKEYC,Ø:POKEY,Ø
76 IFPEEK(Y+22)=36THEN3ØØ
8Ø GOTO51
9Ø Y=Y+Q:YC=YC+Q:POKEYC,Ø:POKEY,Ø:SC=SC+
   1ØØ:E=E+1ØØ
92 FORI=1TO2Ø:POKES1,22Ø+I:NEXTI:POKES1,
   Ø
93 POKEY,32:Y=Y-22:YC=YC-22:POKEYC,Ø:POK
   EY,Ø
96 GOTO51
1ØØ IFY>=8138THENRETURN
1Ø1 E=E-3:IFE<=ØTHEN5ØØ
1Ø2 G=25Ø
1Ø5 FORX=1TO3
11Ø IFPEEK(Y+X)<>32THEN135
115 POKEY+X,33:POKES3,G:G=G+1:NEXTX
119 B=3
12Ø FORX=BTO1STEP-1
125 POKEY+X,32:POKES3,G:G=G-1
129 POKES3,Ø
13Ø NEXTX:RETURN
```

```
135 IFPEEK(Y+X)=34THENE=E+20:SC=SC+15
140 IFPEEK(Y+X)=33ORPEEK(Y+X)=43THENE=E+
    10:SC=SC+5
145 POKES1,195:POKEY+X,42:B=X:FORJ=0TO60
    :NEXTJ
149 POKES1,0
150 GOTO120
300 POKEY,0
305 POKEYC+22,2:POKEYC,2:POKEY+22,41:POK
    EY,36
310 FORB=1TO100
315 POKEY,36
320 POKES4,187
325 POKEY,37
330 POKES4,127
335 NEXTB:POKES4,0
345 POKEY,32:POKEY+22,35
346 E=INT(E*.5)
350 TN=TN-1:IFTN=0THEN1000
351 Y=7813:YC=38533:GOTO50
500 IFPEEK(Y+Q)=36THENPOKEY+23,35:POKEY+
    22,36:GOTO300
501 Y=Y+Q:YC=YC+Q:POKEYC,2:POKEY,42
502 FORH=240TO220STEP-1
503 POKES3,H:NEXTH:POKES3,0
505 POKEY,32
506 TN=TN-1:IFTN=0THEN550
507 YC=38533:Y=7813:GOTO50
550 GOTO1000
600 O=INT(RND(1)*22)+1:IFO=22THENO=0
602 D=INT(RND(1)*2)+1
603 IFD=1THENPOKE38862+O,4:POKE8142+O,36
604 IFD=2THENPOKE38862+O,5:POKE8142+O,35
605 POKE(SK-3),32:POKESK,33
606 IFPEEK(SK+1)=0THENY=Y+1:GOTO500
608 MKC=MKC+1:SK=SK+1
609 IFSK=8098THENSK=8076:MKC=38796:POKE8
    096,32:POKE8097,32:POKE8095,32
610 POKEMKC,8:POKESK,43
611 Z=INT(RND(1)*10)+1:IFZ<>5THEN650
```

```
612 F=INT(RND(1)*242)+1:IFPEEK(FL+F)<>32
    THEN612
614 W=INT(RND(1)*7)+1
615 IFW=7THEN614
616 POKEFC+F,W:POKEFL+F,34
625 IFE<100THENPOKE36879,127
626 IFE>100THENPOKE36879,122
650 RETURN
1000 PRINT"{CLR}"
1005 PRINT"IF YOU WISH TO PLAY":PRINT"AG
     AIN, PRESS THE FIRE BUTTON."
1010 PRINT"{PUR}IF NOT,THEN PRESS THE":P
     RINT"JOYSTICK DOWN."
1015 PRINT"YOUR SCORE:";SC
1020 PRINT"YOUR TIME :";VAL(TI$)
1023 IFSC>=10000THENPRINT"{RVS}{RED}YOU
     HAVE WON!{OFF}"
1025 J=0
1030 J=PEEK(37151)
1035 IFJ=94THENRUN
1040 IFJ=118THENPOKE36869,240:PRINT"{RVS}
     {BLU}{2 SPACES}HAWKMEN OF DINDRIN
     {2 SPACES}{OFF}{2 SPACES}GAME TERMIN
     ATED.":END
1050 GOTO1025
```

Appendix A

Creating Your Own Maze

Maze Generator

Charles Bond

Here's a remarkably short algorithm which produces random mazes on your TV screen.

To understand how it works, refer to the flowchart and the program listing. The following explanation should clarify the details.

The Background Field

The algorithm operates on a background field which must be generated on the screen prior to line number 200. The field must consist of an odd number of horizontal rows, each containing an odd number of cells: a rectangular array. It's convenient to think of the field as a two dimensional array with the upper-left corner having coordinates X=0 and Y=0, where X is the horizontal direction and Y is vertical. No coordinates are used to identify absolute locations by the program, but the concept is useful in configuring the field.

Given that the upper-left cell of the field has coordinates 0,0, then the terminal coordinates both horizontally and vertically must be even numbers. In addition, the background field must be surrounded on all sides by memory cells whose contents are different from the number used to identify the field. That is, if the field consists of reversed (or inverse video) spaces, then the number corresponding to that character must not be visually adjacent to the field.

This could happen inadvertently if the screen RAM and system ROM have contiguous addresses. A sufficient precaution is to avoid covering the entire screen with field. Leave at least one space at the beginning or end of each line and, in general, leave the uppermost and lowermost lines on the screen blank.

The Maze Generator

The creation of the maze begins by placing a special marker in a suitable starting square. The program here always begins at the square just inside the upper-left cell of the previously

drawn field. (Note that with our coordinate scheme this would be cell 1,1.) Any cell with odd-numbered coordinates would work, however, as long as it is internal to the field.

Next, a random direction is chosen by invoking the random number generator in your machine and producing an integer from 0 to 3. This integer, with the aid of a short table, determines a direction and a corresponding cell just two steps away from the current cell. This new cell is examined (PEEKed) to see if it is part of the field. If it is, the direction integer is put there as a marker, and the barrier between it and the current cell is erased.

In addition, the pointer to the current cell is moved to point to the new one. This process is repeated until the new cell fails the test; that is, it is not a field cell. When this happens, the direction vector is rotated 90 degrees and the test is repeated. Thus, the path carved out of the field will continue until a "dead end" is reached.

A dead end, incidentally, could occur in as few as five steps. When it does occur, we can make use of the markers which were dropped along the way "Hansel and Gretel" style. These can be checked to determine which direction we came from, so that we can back up and look for untrodden paths. So long as none can be found, the program will back up, one step at a time, erasing the markers as it goes. When a new direction can be taken, the pointer is set off in that direction, and the process continues as before.

Ultimately, the pointer will return to the start, a condition which is detected by the recovery of the special starting (now "ending") marker. This cell is then blanked and the program is done, leaving the pointer as it was at the start.

The Program

The direction table set up in lines 100 and 110 converts an integer to an address offset. In this case (22 column screen), we wish to be able to step two cells to the right, up, left, or down.

Line 120 contains the variable SC, which is the memory address of the start of screen RAM. Lines 130-160 establish the background field on the screen.

The rest of the program draws the maze, as previously explained. Line 310 is simply a convenient stopping point which prevents the screen from scrolling.

It may not be immediately obvious that this algorithm always produces a maze with only one nontrivial path between any two points, or that the maze will always be completely filled, but this can be proved. While the proofs will not be provided here, math buffs may find it interesting that for a maze of any size there will be exactly:

$$\frac{(H-1)(V-1)}{2} - 1$$ empty cells in the completed maze,

where *H* is the number of cells in each field row and *V* is the number of rows.

An interesting feature of this algorithm is that it works equally well in certain types of nonrectangular fields. U-shaped fields or fields with holes in them are quite suitable — as long as certain restrictions are observed. Just make sure that the coordinates of the upper-left and lower-right cells of any cut out area are pairs of odd numbers. Also, if there is a single row of field cells between any cut out areas and the outside of the original field, it may be removed.

The Mouse

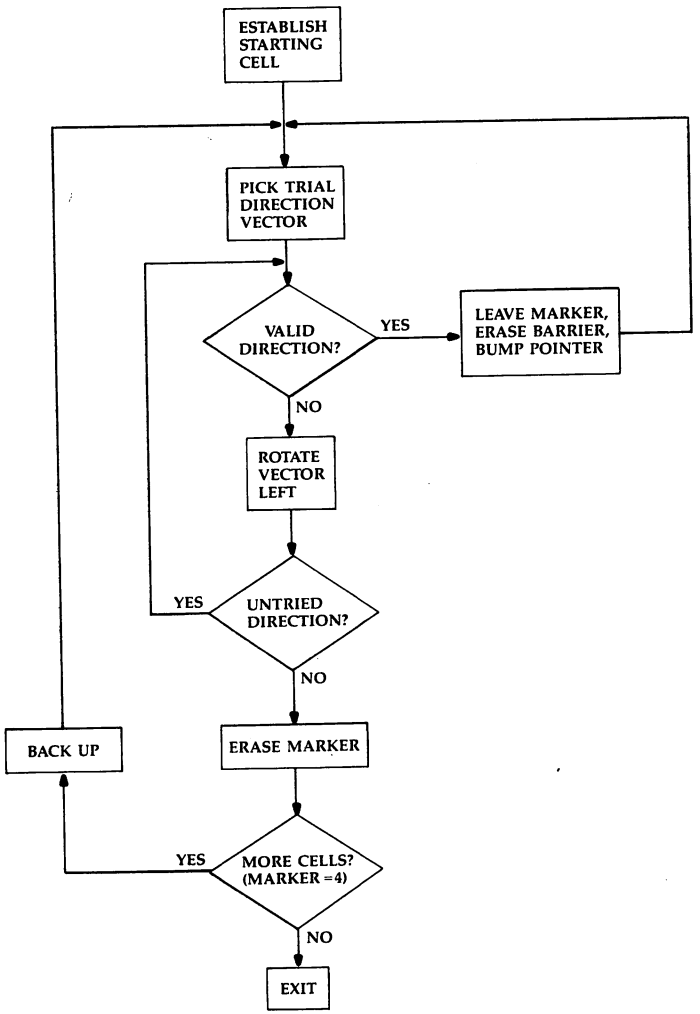
The subroutine on lines 1000 to 1020 produces an artificial "mouse" which roams the maze endlessly. The mouse adheres to a "left-hand rule" when a choice of directions is possible. That is, when it is confronted with a branch-point, it will move off to the left, if possible. Otherwise it will go forward. When no choice is available, it will turn around. These lines are unnecessary for the creation of the maze and may be deleted.

Program A-1. Maze Generator

```
100 DIMA(3)
110 A(0)=2:A(1)=-44:A(2)=-2:A(3)=44
120 WL=160:HL=32:SC=7680:A=SC+45
130 PRINT"{CLR}"
140 FORI=1TO21
150 PRINT"{RVS}{21 SPACES}"
160 NEXTI
210 POKEA,4
220 J=INT(RND(1)*4):X=J
230 B=A+A(J):IFPEEK(B)=WLTHENPOKEB,J:POK
    EA+A(J)/2,HL:A=B:GOTO220
```

```
240 J=(J+1)*-(J<3):IFJ<>XTHEN230
250 J=PEEK(A):POKEA,HL:IFJ<4THENA=A-A(J)
    :GOTO220
310 GETC$:IFC$=""THEN310
1000 POKEA,81:J=2
1010 B=A+A(J)/2:IFPEEK(B)=HLTHENPOKEB,81
     :POKEA,HL:A=B:J=(J+2)+4*(J>1)
1020 J=(J-1)-4*(J=0):GOTO1010
```

Figure 1. Maze Generator Flowchart



Appendix B

Writing Your Own Games

A Reference Guide

Writing Your Own Games: A Reference Guide

Dan Carmichael

We hope that the games in this book have provided you with many hours of fun along with some new and advanced programming techniques.

While it is always enjoyable to play games that other programmers have written, you might want to explore the world of game programming yourself. When you create the games and write the programs yourself, not only is there the fun of play, but also you receive the personal gratification of a job well done. Following are some hints, tips, and references to help you create your own game programs.

The Advantages of Machine Language

The majority of games in this book were written in BASIC, although some have incorporated a few small machine language subroutines. But machine language is the native language of the VIC-20. There are two main advantages of programming your games in machine language.

The first is speed. Machine language, properly written, can be hundreds, if not thousands, of times faster than BASIC. When a program written in BASIC is executed, the BASIC Interpreter looks at each BASIC statement, decides what it means, and interprets it into machine language to be executed. It is this interpretation process that slows BASIC down. When you program in machine language, the BASIC interpretation process is bypassed, and execution speed improves dramatically.

The second advantage of machine language is memory space. A BASIC program can use up to 50 percent more memory than the same program written in machine language. When writing programs on the unexpanded VIC-20, where

you are limited to approximately 3500 bytes, memory usage is an important programming criterion. Saving 25 or 50 percent of memory means that you can add some nice extras to your game program, like enhancing your sound effects or color usage, or even adding a bonus round or two.

If you're going to tackle the task of learning to program in machine language, a machine language monitor/assembler will make programming in machine language easier and more readable. Commodore's version is called the *VICMON*, and the *Hesmon* is produced by Human Engineered Software. Both are available at your local computer dealer for about $50, and both are in cartridge form. Also available are monitor/assemblers in printed (listed) form that are ready to type in and run. Many have been published in *COMPUTE!'s* magazines and books.

Books are almost essential to learning machine language. There are many fine books on the market that deal with programming the 6502 (the microprocessor chip in the VIC-20, the Commodore 64, the Atari, and many other computers). One book especially suited to the novice who wishes to learn machine language programming is Richard Mansfield's *Machine Language for Beginners* (*COMPUTE!* Books, 1983). (For more information on books and monitor/assemblers, see the reference section at the end of this appendix.)

High-Resolution Graphics

High-resolution graphics is the ability to control (turn on or off) each individual pixel (a very tiny dot) on the TV screen. It is used to produce those beautiful, smoothly animated pictures you see on the arcade-style video games.

Standard graphics uses blocks of characters that are composed of eight rows and columns of pixels. When you animate with standard graphics, you are actually moving each character one block, or eight rows or columns at one time. The problem with this method is that it produces jumpy animation. With the ability of high-resolution graphics to turn each pixel on or off, you can produce smooth animation by moving *one* row or column at a time.

There are limitations to high-resolution graphics. One is the amount of memory needed to control the graphics display. To produce high-resolution graphics you have to initiate a process known as "bitmapping" the screen. This means

that you have to assign each individual pixel on the TV screen to a memory location (one bit each, or eight pixels to the byte) inside the computer. This takes up a large amount of memory. So much, in fact, that to bitmap the entire screen would take 4048 bytes of memory. You couldn't accomplish this in the unexpanded VIC.

There is a technique that bitmaps only small portions of the screen but large amounts of memory are still needed. Memory expansion is the easiest solution to this problem.

Another drawback of high-resolution graphics is speed. When you move objects one row of pixels at a time, you have to program more routines to animate the characters. In standard graphics, one command can move the character eight rows at a time, but it would take eight commands to move the same high-resolution display the same distance. Since BASIC is already slow, machine language is a must with high-resolution graphics.

Multicolor Graphics

When programming using standard or high-resolution graphics, you have the choice of setting each character to one of two color possibilities. You can give each character or pixel the background screen color (which will seem to produce a blank space on the screen), or you can assign it one of the eight standard VIC colors. With the multicolor graphics option, you can assign each character or pixel any one of four possibilities: screen color, character color, border color, or auxiliary color. There are eight auxiliary colors, so with multicolor graphics you have the choice of 16 color possibilities.

There is a sacrifice, however, when using a multicolor graphics mode: horizontal resolution. In multicolor mode, each dot is twice as wide as in high-resolution graphics; that is, each dot is two pixels wide. Your drawings will have to be made with thicker columns; but often this problem is far outweighed by the greater number of colors.

Sound Effects

Good sound effects can turn a good game into a great game. And sound isn't just an extra. You can use sound to tell the player that certain events have occurred without forcing him or her to look away from the main action on the screen. You

don't have to *watch* to know that an explosion has occurred, or that you have won a bonus round.

Sound also provides a rhythm, sets a mood, and generally makes the "world" of the game more inviting and complete.

One problem in some BASIC game programs is that the action stops while the sound effects are being generated. There are ways that you can generate sound effects while the action in other parts of the program is still running. One method is by adding user-generated interrupt routines in machine language, of course. (References for texts on sound effects can be found in the reference section at the end of this appendix.)

If you are a beginning or intermediate BASIC programmer, there are many books and articles that can help you advance into higher levels of BASIC proficiency, and if you feel that you have thoroughly mastered BASIC, there is always machine language to be conquered. In fact, programming itself is like the best arcade games — no matter how good you get, there are always new challenges ahead. And instead of just getting your initials on a vanity board, you have a finished game that you and others can play again and again.

References

Machine Language

Butterfield, Jim. "TINYMON1: A Simple Monitor for the VIC." In *COMPUTE!'s First Book of VIC.* Greensboro, NC: COMPUTE! Books, 1982.

Finkel, A., N. Harris, P. Higginbottom, and M. Tomczyk. "Chapter 3: Machine Language Programming Guide." *VIC-20 Programmer's Reference Guide.* Wayne, PA: Commodore Business Machines, 1982, pp. 107-226.

Kavanagh, Russell. "Entering TINYMON1 Directly into Your VIC-20." In *COMPUTE!'s First Book of VIC.* Greensboro, NC: COMPUTE! Books, 1982.

Leventhal, Lance A., and Winthrop Saville. *6502 Assembly Language Subroutines.* Berkeley, CA: Osborne/McGraw Hill, 1982.

Mansfield, Richard. *Machine Language For Beginners.* Greensboro, NC: COMPUTE! Books, 1983. Contains several previously published monitors, maps, routines, etc.

Zaks, Rodnay. *Programming the 6502.* Berkeley, CA: Sybex, Inc., 1979.

High-Resolution Graphics

Calloway, James. "Pixelator." In *COMPUTE!'s Second Book of VIC.* Greensboro, NC: COMPUTE! Books, 1983.

Finkel, A., N. Harris, P. Higginbottom, and M. Tomczyk. *VIC-20 Programmer's Reference Guide.* Wayne, PA: Commodore Business Machines, 1982, pp. 88-92.

Trendowski, Roger N. "Understanding VIC High Resolution Graphics." In *COMPUTE!'s Second Book of VIC.* Greensboro, NC: COMPUTE! Books, 1983.

Multicolor Mode Graphics

Banis, Bud. "UFO Pilot: VIC Custom Characters for Game Graphics." In *COMPUTE!'s Second Book of VIC.* Greensboro, NC: COMPUTE! Books, 1983.

Finkel, A., N. Harris, P. Higginbottom, and M. Tomczyk. *VIC-20 Programmer's Reference Guide.* Wayne, PA: Commodore Business Machines, 1982, pp. 92-94.

Sound Effects

Finkel, A., N. Harris, P. Higginbottom, and M. Tomczyk. *VIC-20 Programmer's Reference Guide.* Wayne, PA: Commodore Business Machines, 1982, pp. 95-105.

Lee, Robert. "VIC Sound Generator." In *COMPUTE!'s Second Book of VIC.* Greensboro, NC: COMPUTE! Books, 1983.

Appendix C

A Beginner's Guide To Typing In Programs

A Beginner's Guide To Typing In Programs

What Is a Program?

A computer cannot perform any task by itself. Like a car without gas, a computer has *potential,* but without a program, it isn't going anywhere. The programs published in *COMPUTE!'s First Book of VIC Games* are written in a computer language called BASIC. BASIC is easy to learn and is built into the VIC.

BASIC Programs

All but one of the programs in this book are for the Unexpanded VIC. Most require that any additional memory cartridges be removed or disabled. The exceptions are the enhanced version of "Air Defense" in Part 3 and "Word Hunt" in Part 4, both of which require 8K or more of expansion memory.

Computers can be picky. Unlike the English language, which is full of ambiguities, BASIC usually has only one "right way" of stating something. Every letter, character, or number is significant. A common mistake is substituting a letter such as "O" for the numeral "0", a lowercase "l" for the numeral "1", or an uppercase "B" for the numeral "8". Also, you must enter all punctuation such as colons and commas just as they appear in the book. Spacing can be important. To be safe, type in the listings *exactly* as they appear unless the article recommends "crunching" techniques to save memory.

Braces and Special Characters

The exception to this typing rule is when you see the braces, such as "{DOWN}", or special brackets, such as [< + >]. Anything within a set of braces or special brackets is a special character or characters that cannot easily be listed on a printer. When you come across such a special statement, refer to the section of this book entitled "Listing Conventions."

About DATA Statements

Some programs contain a section or sections of DATA statements. These lines provide information needed by the program. Some DATA statements contain actual programs (called machine language); others contain graphics codes. These lines are especially sensitive to errors.

If a single number in any one DATA statement is mistyped, your machine could "lock up," or "crash." The keyboard may seem "dead," and the screen may go blank. Don't panic — no damage is done. To regain control, you have to turn off your computer, then turn it back on. This will erase whatever program was in memory, so always SAVE a copy of your program before you RUN it. If your computer crashes, you can LOAD the program and look for your mistake.

Sometimes a mistyped DATA statement will cause an error message when the program is RUN. The error message may refer to the program line that READs the data. *The error is still in the DATA statements, though.*

Get to Know Your Machine

You should familiarize yourself with your VIC before attempting to type in a program. Learn the statements you use to store and retrieve programs from tape or disk. You'll want to save a copy of your program, so that you won't have to type it in every time you want to use it. Learn to use your machine's editing functions. How do you change a line if you made a mistake? Do you know how to enter inverse video, lowercase, and control characters? It's all explained in your computer's manuals.

A Quick Review

1) Type in the program a line at a time, in order. Press RETURN at the end of each program line. Use backspace or the back arrow to correct mistakes.

2) Check the line you've typed against the line in the listing. You can check the entire program again if you get an error when you RUN the program.

3) Make sure you've entered statements in braces as the appropriate control key (see "Listing Conventions" elsewhere in this book).

4) Be sure to SAVE the program on disk or tape *before* running the program.

Listing Conventions

Many of the programs which are listed in this book contain special control characters (cursor control, color keys, inverse video, etc.). To make it easy to know exactly what to type when entering one of these programs into your computer, we have established the following listing conventions.

Generally, any VIC-20 program listings will contain words within braces which spell out any special characters: {DOWN} would mean to press the cursor down key. {5 SPACES} would mean to press the space bar five times.

To indicate that a key should be *shifted* (hold down the SHIFT key while pressing the other key), the key would be underlined in our listings. For example, S would mean to type the S key while holding the shift key. This would appear on your screen as a "heart" symbol. If you find an underlined key enclosed in braces (e.g., {10 N}), you should type the key as many times as indicated (in our example, you would enter ten shifted N's).

If a key is enclosed in special brackets, |< >|, you should hold down the *Commodore key* while pressing the key inside the special brackets. (The Commodore key is the key in the lower-left corner of the keyboard.) Again, if the key is preceded by a number, you should press the key as many times as necessary.

Rarely, you'll see a solitary letter of the alphabet enclosed in braces. You should never have to enter such a character on the VIC-20, but if you do, you would have to leave the quote mode (press RETURN and cursor back up to the position where the control character should go), press CTRL-9 (RVS ON), the letter in braces, and then CTRL-0 (RVS OFF).

About the *quote mode:* you know that you can move the cursor around the screen with the CRSR keys. Sometimes a programmer will want to move the cursor under program control. That's why you see all the {LEFT}'s, {HOME}'s, and {BLU}'s in our programs. The only way the computer can tell

the difference between direct and programmed cursor control is the quote mode.

Once you press the quote (the double quote, SHIFT-2), you are in the quote mode. If you type something and then try to change it by moving the cursor left, you'll only get a bunch of reverse-video lines. These are the symbols for cursor left. The only editing key that isn't programmable is the DEL key; you can still use DEL to back up and edit the line. Once you type another quote, you are out of quote mode.

You also go into quote mode when you INSerT spaces into a line. In any case, the easiest way to get out of quote mode is to just press RETURN. You'll then be out of quote mode and you can cursor up to the mistyped line and fix it.

Use the following table when entering cursor and color control keys:

When You Read:	Press:	See:	When You Read:	Press:	See:
{CLEAR}	SHIFT CLR/HOME		{PUR}	CTRL 5	
{HOME}	CLR/HOME		{GRN}	CTRL 6	
{UP}	SHIFT ↑ CRSR ↓		{BLU}	CTRL 7	
{DOWN}	↑ CRSR ↓		{YEL}	CTRL 8	
{LEFT}	SHIFT ← CRSR →		{F1}	f1	
{RIGHT}	← CRSR →		{F2}	f2	
{RVS}	CTRL 9		{F3}	f3	
{OFF}	CTRL 0		{F4}	f4	
{BLK}	CTRL 1		{F5}	f5	
{WHT}	CTRL 2		{F6}	f6	
{RED}	CTRL 3		{F7}	f7	
{CYN}	CTRL 4		{F8}	f8	

Index

If you've enjoyed the articles in this book, you'll find the same style and quality in every monthly issue of **COMPUTE!** Magazine. Use this form to order your subscription to **COMPUTE!**

For Fastest Service,
Call Our **Toll-Free** US Order Line

800-334-0868

In NC call 919-275-9809

COMPUTE!
P.O. Box 5406
Greensboro, NC 27403

My Computer Is:
☐ PET ☐ Apple ☐ Atari ☐ VIC ☐ Other _____ ☐ Don't yet have one...

☐ $20.00 One Year US Subscription
☐ $36.00 Two Year US Subscription
☐ $54.00 Three Year US Subscription

Subscription rates outside the US:
☐ $25.00 Canada
☐ $38.00 Europe, Australia, New Zealand/Air Delivery
☐ $48.00 Middle East, North Africa, Central America/Air Mail
☐ $68.00 Elsewhere/Air Mail
☐ $25.00 International Surface Mail (lengthy, unreliable delivery)

Name _____

Address _____

City _____ State _____ Zip _____

Country _____

Payment must be in US Funds drawn on a US Bank; International Money Order, or charge card.

☐ Payment Enclosed ☐ VISA
☐ MasterCard ☐ American Express
Acct. No. _____ Expires ____/____

13-2

COMPUTE! Books

P.O. Box 5406 Greensboro, NC 27403

Ask your retailer for these **COMPUTE! Books**. If he or she has sold out, order directly from **COMPUTE!**

For Fastest Service
Call Our **TOLL FREE US Order Line**

800-334-0868
In NC call 919-275-9809

Quantity	Title	Price	Total
_____	The Beginner's Guide to Buying A Personal Computer	$ 3.95**	_____
_____	COMPUTE!'s First Book of Atari	$12.95*	_____
_____	Inside Atari DOS	$19.95*	_____
_____	COMPUTE!'s First Book of PET/CBM	$12.95*	_____
_____	Programming the PET/CBM	$24.95***	_____
_____	Every Kid's First Book of Robots and Computers	$ 4.95**	_____
_____	COMPUTE!'s Second Book of Atari	$12.95*	_____
_____	COMPUTE!'s First Book of VIC	$12.95*	_____
_____	COMPUTE!'s First Book of Atari Graphics	$12.95*	_____
_____	Mapping the Atari	$14.95*	_____
_____	Home Energy Applications On Your Personal Computer	$14.95*	_____
_____	Machine Language for Beginners	$12.95*	_____

* Add $2 shipping and handling. Outside US add $5 air mail; $2 surface mail.
** Add $1 shipping and handling. Outside US add $5 air mail; $2 surface mail.
*** Add $3 shipping and handling. Outside US add $10 air mail; $3 surface mail.

Please add shipping and handling for each book ordered. _____

Total enclosed or to be charged. _____

All orders must be prepaid (money order, check, or charge). All payments must be in US funds. NC residents add 4% sales tax.

☐ Payment enclosed Please charge my: ☐ VISA ☐ MasterCard
☐ American Express Acc't. No. _____ Expires ___/___

Name _____

Address _____

City _____ State _____ Zip _____

Country _____

Allow 4-5 weeks for delivery.

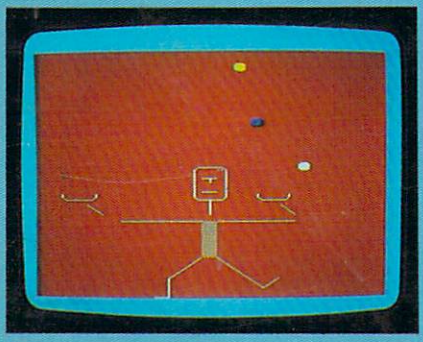
COMPUTE!'s
First Book of VIC Games

Twenty-four games for the VIC-20 computer, complete and ready to type in, so no programming knowledge is necessary.

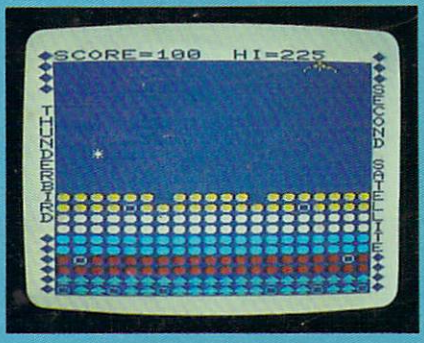
Here are 15 of the best games from *COMPUTE!* Magazine and *COMPUTE!'s Gazette*. Many of these have been updated and improved since their original appearance in the magazines.

You also get nice, never-before-published games, including the fast-action arcade-style "Richthofen's Revenge" and an adventure game for the unexpanded VIC, "Pharaoh's Treasure."

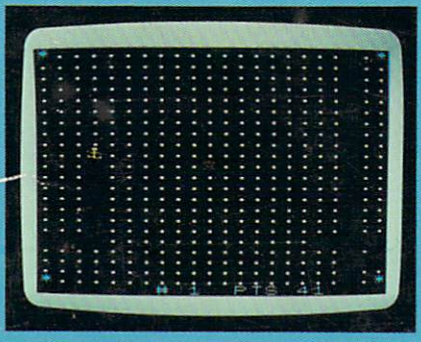
Since you can follow the complete BASIC listing, you can see for yourself how programmers create games. Also, several chapters are devoted to showing you just how to program your own games.



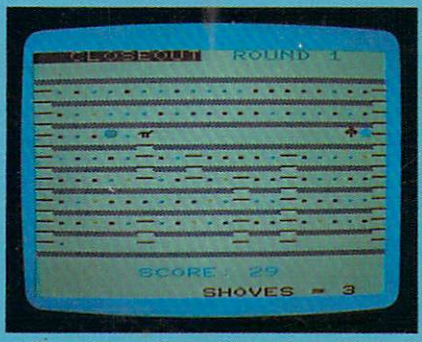
Juggler



Thunderbird



Marble Hunt



Closeout

ISBN 0-942386-13-2

$12.95