ARCADE GAMES FOR THE COMMODORE 64

Fanfare House, Inc.
Mary Ann Chapman and David Harper, Editors



IMPORTANT NOTICE

Directions for using this diskette are contained in the introduction of the accompaning book.

- 1. Insert the diskette in your Commodore Disk Drive.
- 2. After the C64 displays the ready prompt type:

LOAD "NAME OF THE GAME",8

- 3. After a few seconds the C64 will give you the ready prompt.
- 4. Type: RUN

You are now ready to play.

WARNING: DO NOT SAVE ANY OTHER PROGRAM ON THIS DISKETTE.

Arcade Games for the Commodore 64

Arcade Games for the Commodore 64

Fanfare House, Inc.

Edited by: MaryAnn Chapman and David Harper

CBS Computer Books

HOLT, RINEHART AND WINSTON

New York Chicago San Francisco Philadelphia

Montreal Toronto London Sydney Tokyo

Mexico City Rio de Janeiro Madrid

Trademarks and Copyrights:

CP/M is a registered trademark of Digital Research Incorporated Radio Shack and TRS-80 are registered trademarks of Tandy, Inc. WORDSTAR is a registered trademark of MicroPro International Corporation Commodore 64 is a trademark of Commodore Electronics, Limited

Copyright[©] 1985 Fanfare House, Inc. All rights reserved. Address correspondence to: 383 Madison Avenue, New York, NY 10017

First distributed to the trade in 1985 by Holt, Rinehart and Winston general book division.

Library of Congress Cataloging in Publication Data

FANFARE HOUSE, INC. Arcade games for the Commodore 64.

1. Computer games. 2. Commodore 64 (Computer)-Programming. I. Fanfare House, Inc. GV1469.2.A73 1985 794.8'2 84-17632

ISBN 0-03-001049-7

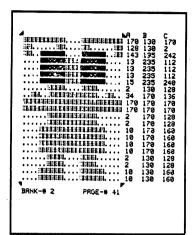
Printed in the United States of America

Published simultaneously in Canada

567 039 987654321

CBS COLLEGE PUBLISHING Holt, Rinehart and Winston The Dryden Press Saunders College Publishing

Table of Contents



Preface ix

Acknowledgements xi

Introduction xiii

Playing the Games xiii Information Provided for Each Game xvi Modifying the Games xx Try Programming Yourself! xxi

Rocket Lander 1

Sharpen your vehicle control skills with this rocket landing game.

Techniques Discussed: Sprites without DATA statements. Redefined characters.

Squirrel Away 7

Susie Squirrel needs to store her winter supply of nuts in her underground burrow.

Techniques Discussed: Moving sprites and characters together. Detecting sprite-to-character collisions.

11

Ghost Hunt 13

Three friends are out late at night trying to catch the slippery, tricky neighborhood ghost.

Technique Discussed: Detecting sprite-to-sprite collisions.

Firecracker Boy 19

Save the Fourth-of-July parade from interruption by a mischievous kid.

Techniques Discussed: Two-part design to handle large programs. Using variables to control processing sequence.

Into the Pot 27

Trap your dinner by moving boulders to guide chickens to your cooking pot.

Technique Discussed: Variables with fixed values.

Yorick's Revenge 33

Video pinball game, patterned after the action of a pinball machine.

Technique Discussed: Programming for speed.

Space Crash 39

Protect your home planet from swarms of evil aliens. *Technique Discussed:* Redefined characters.

The Blobbis 45

Break through the force field that protects three evil Blobbis.

Technique Discussed: Animation using arrays.

Sound the Whistle 53

Two-player game similar to football, without a ball. *Techniques Discussed:* Considerations in creating sounds. Upper/lower case mode.

Bumper Ball 59

A "pong" type of game for one player.

Techniques Discussed: Programming

Techniques Discussed: Programming for speed. Use of variables to position cursor. Complications in collision detection.

Ride the Wind 65

Learn to land an experimental hot-air balloon. *Technique Discussed:* Calculating effect of outside forces on moving objects.

Death Valley Patrol 71

Rescue paratroopers who have been stranded in the barren Death Valley.

Technique Discussed: Controlled random screens.

Mission: Tobor 79

Refuel automated atomic energy reactors on the planet Tobor by remote control.

Technique Discussed: Special use of the keyboard buffer.

Cosmo's Rescue 87

Free Cosmo from the hazardous cavern of the yellow monster Grunk-Snort.

Technique Discussed: Bank-switching to handle large programs.

Mazemaster 95

Find your way through complex mazes of threedimensional corridors.

Techniques Discussed: Completely random screen generation. Simulated three-dimensional display.

Appendix A 101

Preface

rcade Games For The Commodore 64 has been created for the C-64 owner who enjoys playing computer games and wants to learn more about the "magic" that makes them work. We have provided fifteen games which demonstrate the power of the Commodore 64 home computer. The games utilize most of the features of the C-64, including color, sound, music, animated character and sprite graphics, multicolored characters and sprites, and redefined characters. Some of the games produce multivoice sound effects. Some use more than thirty sprites! All offer colorful, animated fun.

All the games are written in BASIC and annotated with enough programming detail that you will be able to explore and experiment with many aspects of the programs. Even if you do not have a complete understanding of the BASIC language or any previous programming experience, you can exercise your own creativity by changing the games, using the modification hints given. You may find this to be even more fun than playing them!

For each game, in addition to play instructions and program listings, we have included a discussion of some of the programming techniques used in the game's creation. The explanation should satisfy a non-programmer's curiosity, or point out topics for further study by readers who want to program games themselves.

Our intent was to present enough information that you will feel you understand the general method by which the programmer achieved a certain result. If you wish to achieve the same result in a program of your own, studying our program as an example will be helpful. You will probably need to refer to other texts, however, to

obtain all the information you need. In many cases, we provide specific references for additional details.

Because we wanted to create games which would exploit the power of the Commodore 64 computer and be fun to play, each game program is elaborate and contains many different tricks and techniques. An attempt to thoroughly describe just one program could fill a book. The "Programmer's Notes" for each game highlight just one or two game programming topics. If you examine the programs in detail, you will uncover many more techniques which you might wish to study further.

Though we have made our explanations as clear as possible, you may find them easier to understand if you have some familiarity with the *Commodore 64 User's Guide* which came with your computer. In fact, we have often made reference to that guide for further details on topics which were beyond the scope of this book.

Whether you are a game player, a programming enthusiast, or just someone curious about computers, reading this book and looking over the program listings should give you a general understanding of the structure of a game program and some methods by which programmers achieve the effects which make games fun.

In addition to many hours of entertainment, we hope our games provide you with a learning experience as you discover how the various programmers have taken advantage of the Commodore 64's special features to implement their game ideas.

Mary Ann Chapman and David Harper Fanfare House, Inc.

Acknowledgements

This book benefitted from the efforts of many of the staff and friends of Fanfare House, Inc., some of whom made contributions in more than one area.

Credit is due to Mitch Bunnell, John Carmody, Carl Dobbs, Beryl Fielder, Dennis Fleming, Bud Gressett, Mike Kupka, Charles Manry, Frank Mikulastik, Charles Mott, Tracy Powell, Dick Ramsell, Stanley Rosenthal, Rick Ryznar, Penny Smith, Igor Tulchinsky, Steve Watson, David Williams, and especially Vera Gonzales, for their participation in past projects which made this one possible.

We appreciate the assistance of Bob Behrstock, Emery Froelich, Marilyn Harper, Hoyet Hemphill, Dick Ramsell, John Wheeler, and David Williams, all of whom reviewed the manuscript and made helpful suggestions. Leonard and Kim Brusatori parent-tested the text, and Todd and Amy Ramsel and Jim Dwyer kid-tested the games.

Dave Dusthimer of CBS Technical and Professional Publications gave us heart with his enthusiasm and happily accepted our format suggestions. We appreciate the efforts of Chris Titus, of The Blacksburg Group, who got us together with CBS.

Special thanks to Emery Froelich and Fred Burchard, who lent their support in many ways; to Jesse Rothstein and Arthur Falconer, who have been generous with their wisdom and advice; to our secretary Beryl Fielder, who deciphered programmers' scribblings; and to Marilyn Harper, whose technical assistance has saved the day on many occasions.

The manuscript was prepared using WORDSTAR on a Radio Shack TRS-80 Model II computer running the CP/M operating system.

Introduction

The games in this collection are all "arcade-style", requiring skill, coordination and strategy to achieve high scores. In order for you to understand and modify the game programs, we chose to write them in BASIC, which is the language most commonly used by Commodore 64 owners.

Although BASIC is simple to demonstrate and understand, it does not result in extremely fast-running programs. As a result, the games do not have quite the lightning speed found in true arcade games, which are generally written in faster languages, but they are still fun to play. The methods used to achieve the "arcade" effects are the same as those used in the faster games.

Our programmers have created games with everything from squirrels to space ships, including helicopters, hot-air balloons, ghosts, aliens and robots. You will be playing a pinball game, putting out firecrackers, storing nuts for the winter, and trying to rescue, capture or escape from all sorts of strange creatures. There are two games that simulate sports: one is a "pong" or handball-type game, while the other is similar to football, for two players.

Playing the Games

Loading and Running the Games

All games are loaded and run in the same way. Place the disk which came with this book into your disk drive. Be sure it is inserted correctly, with the label side facing up and the edge farthest away from the label inserted first.

Consult your disk drive user's manual if you need more assistance.

If you wish to display a list of all the game programs on the disk, type LOAD"\$",8 and press the RETURN key. This loads the disk's program directory into the C-64's memory. When the computer displays READY, type LIST and press the RETURN key, to see the directory of programs.

To run a particular game, determine the exact name of the game from either the disk directory or the description of the game in the book. Type LOAD"game name",8 and press the RETURN key. After a delay of 15 seconds or more, depending on the size of the game, the computer will display READY. This indicates that the game has been found and loaded into the C-64's memory and is ready to be played. Type RUN and press the RETURN key, and the game's title screen will appear. Further instructions on how to play are in the description for each game.

For example, one of the games is called **Ghost Hunt**. To load this game, type **LOAD**"**GHOST HUNT**",8 and press the **RETURN** key. Note: The space between *GHOST* and *HUNT* is part of the name and must be included, or the computer won't find the game on the disk.

Some of the games actually consist of two programs. To play a two-part game, you load and run one program, and that program automatically loads and runs the other. The second program always has a name beginning with &&. For example, the game called Firecracker Boy actually consists of the programs FIRECRACKER BOY and && FIRECRACKER. If you attempt to load and run && FIRECRACKER, it will run but it may not work correctly, and the game's characters will have a very strange appearance.

How to Stop a Game

When you are finished with one game and want to play a different one, you must stop the first game and get the computer ready to accept a new **LOAD** instruction.

Most of the games can be stopped by pressing and holding down the **RUN/STOP** key and then pressing the **RESTORE** key. The C-64 will respond by displaying

READY, meaning that it is ready for a new command. You may then load and run a different game or run the same one again by typing **RUN** and pressing **RETURN**. Even after pressing **RUN/STOP** and **RESTORE**, the original game is still in the computer's memory.

Games which use memory bank switching techniques (which will be discussed later) cannot be stopped quite so easily. The descriptions of those games direct you to this section for instructions on stopping them.

In those games, pressing RUN/STOP and RESTORE will cause the computer to behave very strangely. Instead, use the following procedures: Press RUN/STOP. Type RUN20000 and press the RETURN key. Then press and hold down RUN/STOP and press RESTORE. The computer will then say READY, and you can load and run another game, or type RUN and press RETURN to run the same game again.

Ordinarily these procedures will be sufficient to stop a game, load another one, and run it. However, due to memory manipulation techniques used in some games, you may occasionally have difficulty starting the next game. If that occurs, or if you want to stop a game and get the computer back to **READY** without going through the procedures above, simply turn the computer off and on again. This will completely clear the C-64's memory. Then load the game you want to play.

Joysticks. Many of the games are played with one joystick. For those games, plug a joystick into control port 2 on the right-hand side of the Commodore 64, next to the ON/OFF switch. One game, Sound the Whistle, is played by two people and requires two joysticks, one plugged into each control port. Be sure you hold the joystick in the proper position.

Sound. All of the games are spiced with interesting sounds as they are played. Adjust the volume on your television or monitor to a comfortable level so that the sound effects will add to your enjoyment of the games. Of course, if someone in the room doesn't share your appreciation for electronic game sounds, you can turn the volume off completely.

Information provided for each game

The initial section of information on each game explains the basic idea of the game, describes the scene, introduces the characters, and explains how to load and play the game. You may wish to read just that section and then play the game, before proceeding to the program documentation which follows.

In the Programmer's Notes, the programmer of the game explains some of the methods used to make the game work as it does. Often this section includes a description of a programming challenge inherent to the game design, and how it was solved. Our programmers have used many techniques, only some of which are explained. If you carefully examine the program listings, you will uncover and begin to understand other techniques that can be useful if you start writing your own programs.

Hints for Modifying the Game include both simple changes which you can make to adjust the speed or difficulty of the game, and general suggestions for more complex modifications. As you play the games, other ideas may occur to you. When you begin to implement your own ideas, you will see how really creative the programming process can be!

Under Major Routines, you will find a synopsis, by line number, of the major functions of the game program. The significant variables used in the program are listed in the Major Variables section, along with a brief description of their use. Both of these sections will aid your understanding of how the program works.

For each game which uses sprites, a Sprites section provides information for each sprite used. The Line Number column, next to the description of a particular sprite, indicates the line numbers in the program where the DATA statements for that sprite may be found. The location in memory where each sprite is stored is represented by the number under the Page heading.

Each game which uses redefined characters has a Characters section which provides information for each redefined or custom character used. The Line Numbers column, next to the description of a particular character, indicates the line numbers in the program where DATA statements used for the redefinition may be found.

Sprites

The sprite feature is the C-64's way of handling special shapes which are to be moved around on the screen. By creating moving objects as sprites, a programmer greatly simplifies the programming required to produce a game on the C-64.

Sprites are explained in Chapter 6 of the Commodore 64 User's Guide. Because this powerful feature is used extensively in most of the games in this book, a little further explanation here may be helpful. If your interest in technical details is minimal, this discussion is not essential to your understanding of the rest of the Introduction and the game descriptions.

To create a sprite, the programmer includes data in the program to tell the computer what size, shape, and color the sprite is to be and where it is to be located on the screen at any particular moment. The computer then automatically handles many functions that otherwise would require extensive programming.

The computer's VIC (Video Interface Controller) chip uses the programmer's data to display the sprite on the screen with the desired color, shape and size. By continuously redisplaying it at locations specified by the programmer, the VIC chip makes the sprite appear to move around. The VIC chip also detects collisions when the sprite is in the same place on the screen as another sprite or a background character.

The data which specifies the sprite's shape is usually coded into DATA statements in the program. With instructions which are executed very early when the program begins running, the sprite shape data is moved into an area of memory outside the program, where it is accessed later by the VIC chip.

The C-64's memory contains 64 Kilobytes of storage (lK = 1024 bytes or characters). It is organized into "banks" of 16K each, which are further divided into 256 "pages" of 64 bytes each. A sprite shape definition is 3 bytes wide by 21 bytes high, or a total of 63 bytes. The

64th byte of a page used to define a sprite always has a value of zero. In our documentation, the number under the Page heading is a reference to the page in memory, outside the program itself, where the shape data for a particular sprite is stored.

For example, if the page number for a rocket sprite stored in bank 0 is 192, the data that determines the shape of the rocket is stored in the C-64's memory beginning at page 192, or location 12288 (192 * 64 = 12288).

In most of these games, the sprites are stored along with the program in the first 16K bank of memory, which is known as bank 0. Some games use a technique called bank switching, which permits the sprites to be accessed by the VIC chip from a different bank. If a game uses a bank other than 0, its program documentation will so indicate. (For more discussion of bank switching, see the Programmer's Notes for the game **Cosmo's Rescue**.

A maximum of eight sprites may be on the screen at any one time. Eight locations in memory, beginning at location 2040, are used as sprite page number pointers. When a game is running and the program calls for a sprite object to be displayed, the VIC chip obtains the page number to use from the sprite page pointer location assigned to that sprite.

When an object is to be animated, several different sprites are used, each representing a different step in the animation. Then a sequence of different sprite pages, one for each sprite, is used to create the animation effect. The page number stored in the sprite page pointer is continuously changed to the next page number in the animation sequence.

For each game, we have printed a few of the major sprites in graphic form. To the right of each sprite printout, you will find three columns of numbers. Each row contains the actual numbers used in the program's DATA statements to define the corresponding row of the sprite. The numbers below the sprite represent the location in memory (bank and page) where the sprite data is stored.

If you would like to learn how the numbers in DATA statements translate into a particular sprite shape, how sprites can be more than one color, and other details of using sprites, refer to the Commodore 64 User's Guide or to

some of the other excellent books now available about sprite programming on the Commodore 64.

Sprites are somewhat complicated to set up initially, but then they can be controlled and moved around with a minimum of programming. In most cases they offer the most advantageous method of creating moving objects. Their disadvantage is that no more than eight may be on the screen at one time.

Characters and Redefined Characters

The C-64 also offers another method of creating movable shapes in a program. You may have seen "graphics" programs in which characters such as X and O are used to draw the shapes. In addition to alphabetic and numeric characters, graphics characters with many different shapes are available on the C-64. These graphics characters can be arranged to form the desired object, and program instructions can be written to move the group of characters around on the screen.

It is possible to invent entirely new graphics shapes if the ones available aren't suitable for the desired effect. The programmer then includes instructions which redefine the standard characters to give them the specified new form.

Some of the game programs in this book use redefined characters to create animated figures and other special shapes. Once again, consult one of the books on graphics programming for the Commodore 64 for complete details on how this is done.

Since there are 256 characters in the C-64 character set, there may be up to 256 redefined character shapes in a program. Therefore this technique is sometimes used when a game design calls for a large number of movable objects, even though much more programming is required to move them around on the screen than is necessary with sprites. The most advantageous use of redefined characters is for complex background shapes that do not require movement.

The Program Listing

For readers who do not have a printer, and to save time for those who do, we have included a program listing. It is printed exactly as it would be if you loaded the game and printed it on your own printer. A complete listing is invaluable if you want to learn how a program works or figure out how to modify it.

The many strange symbols in the listings are the Commodore 64's special graphics and control characters. When you list the program on your computer monitor, you will see the same symbols. To learn which keys to press to obtain these symbols and what the control characters do, consult Chapter 5 of the Commodore 64 User's Guide or one of the other books available on graphics programming for the Commodore 64.

Modifying the Games

If you want to follow our modification hints and experiment with changing the games, we suggest that you first set up a separate disk for your modified versions.

First, format a new disk as explained in the user's manual for your disk drive. Load the program into your computer from the original disk. Then remove the original disk, insert your new formatted disk, and save the program. You will still have the program loaded in the computer, and you can then begin making your changes.

Remember that when making changes, it is very easy to make errors that introduce "program bugs". The bugs can be severe enough to cause the computer to "lock up" when you try to run the game. If this happens, you will be unable to save your changes. Therefore, we suggest that each time you are ready to try running a newly modified program, you save it first. Even if it contains a bug, it may be easier just to correct the bug than to enter all your changes again.

Each time you save a version of the game, give it a different name, and keep each version until you are sure you will never want it again. For example, if you are working with the game Ghost Hunt, you may wish to name the modified versions GHOST HUNT 1, GHOST HUNT 2, and GHOST HUNT 3. After you are certain that GHOST HUNT 3 works, you may decide to delete the other two versions from the disk. When you complete a modified version that you plan to save indefinitely, give it yet another name, such as GHOST HUNT A or GHOST CHASE.

NEVER TRY TO SAVE A PROGRAM ON THE DISK THAT CAME WITH THIS BOOK! You might accidentally damage the disk and lose all its games forever! As a safeguard, you could place a write-protect tab over the square notch on the original disk.

Try Programming Yourself!

Many computer owners begin to explore programming because they are intrigued by the inner workings of the programs they have purchased. As you begin to learn more about programming and begin to understand the game programs in this book, you may have modification ideas of your own. Go ahead and try them!

You may find yourself becoming more and more involved in programming, just because you enjoy it. Writing programs can be a lot of fun!

Programming Guidelines

If you decide to do much programming, you will want to refer to one of the many programming texts on the BASIC language. To supplement the information they provide on the features of the language, we would like to offer some hints on style and structure which can make your programs easier to debug and modify. These techniques can make your programming efforts more successful and less frustrating, whether you are writing original programs or simply modifying the game programs in this book.

- 1) Plan first program later. Think through your design or proposed change before you begin to write any program instructions. Write down everything the program will do and the steps it will follow.
- 2) Don't be too clever. Use straightforward logic that will still make sense to you if you decide to work on the program again a month later.
- 3) Plan for speed, but don't make it your first priority. Get the program working correctly first. Then if some part of it seems to be running too slowly, you can concentrate your efforts on speeding up that routine.
- 4) Organize your program by dividing the program tasks into logical subroutines to be performed by one control section. Place the subroutines below the control

section. Then, when looking at or working on the program, you can see the sequence of tasks at a glance.

- 5) Make variable names as meaningful as possible. SC would be a better variable name than R9 to keep track of the score. Use a name like VI for the base address of the VIC chip (53248). VI +16 is more easily remembered and meaningful than 53264.
- 6) Document your programs internally with remarks. Identify the function of each subroutine with a remark at the beginning of the subroutine. Identify the usage of each group of DATA statements. Comment on any section of the program for which the function is not obvious. These remarks can be great timesavers later if you have to debug or modify the program.

As you examine the programs and their documentation in this book, ask yourself two questions: 1) What do you wish the programmers had done to make it easier for you to understand the program or to help you make changes? 2) What did they do which you found helpful? Your answers to these questions will provide useful guidance when you begin writing your own programs.

Compilers and Other Languages

If speed is important in your programs, you may find that the Commodore 64's BASIC interpreter cannot execute your program instructions quickly enough. Each time the program is run, an interpretive language must convert each statement in the program into machine language before it can be executed by the computer.

You may wish to acquire a BASIC compiler, which is a program used to convert your BASIC source program into machine language in a separate step, creating an object version of the program. No time is required for conversion when the object program is executed, and the program runs faster. You may find it runs so much faster that you actually have to make changes to slow it down!

Usually there are subtle differences between interpreter and compiler versions of the same language. Often some programming changes are required when moving from one to the other. You would be well-advised to wait until you are very comfortable with programming on the C-64 before you attempt to use a compiler.

You may also want to learn to use assembly language to speed up critical routines in your BASIC programs. Or, you might investigate other languages which are faster than BASIC, such as Pascal, FORTH, or 'C', all of which are compiled languages.

Programming References

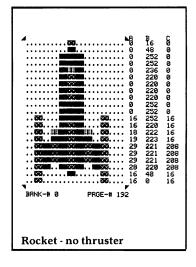
For programming information beyond the scope of the Commodore 64 User's Guide, a number of excellent publications are available. The Commodore 64 Programmer's Reference Guide (Commodore Business Machines, Inc. and Howard W. Sams, Inc., 1982), available in bookstores, is a complete technical reference for the C-64 and is essential for the serious programmer.

For a step-by-step tutorial, however, the many other books available on C-64 graphics programming will be more appropriate. It is difficult to recommend specific titles because, while many of them are excellent, they vary widely in content and level of complexity. Find a bookstore with a good selection of Commodore computer books, and select some that are designed for your level of knowledge and include the topics of interest to you.

Arcade Games for the Commodore 64

Rocket Lander

Charles Mott, Jr.



Stationed inside the red control tower, you are a rocket landing controller responsible for the safe landing of incoming rockets. Each rocket must touch down very gently on its black landing pad before running out of fuel, or it will explode.

Incoming rockets appear on the screen from the upper left and are controlled with function keys. Press F1 or F3 to fire the side thrusters, or F5 to turn on the main engines and slow your descent. When the landing pad is on the left, safe landings can be made easily. But if the landing pad is toward the right of the screen, be careful and watch your fuel supply! After a rocket has been landed safely or has crashed, press any key to start a new landing sequence with another rocket headed for a different landing pad.

You may wish to use **Rocket Lander** as a practice game to help you get a better feel for the delicate control skills necessary for such games as **Cosmo's Rescue** and **Ride the Wind**.

Load the game by typing LOAD"ROCKET LANDER",8 and pressing the RETURN key. After about 20 seconds, the computer will say READY, indicating that the game is loaded. Then type RUN and press the RETURN key. The title screen will appear, followed by an instruction screen with the message GET READY-ROCKETS APPROACHING. After about 40 seconds, the game will start. Happy landings!

Rocket Lander

Programmer's Notes

Sprites without DATA Statements. Five different sprites for the rocket in Rocket Lander are defined in DATA statements. A sixth sprite, for the landing pad, is defined using another technique instead of DATA statements.

The landing pad is simply a solid rectangular box with nothing showing on the screen except its top. If the sprite had been defined with DATA statements, all the numbers in the DATA statements would be 255. Instead, line 10 of the program contains a loop which stores the value 255 directly into all 63 bytes of memory set aside for the landing pad sprite definition.

Many sprites with simple designs can be created in a similar way. Instead of typing one value over and over again into DATA statements in the program, taking up valuable space in memory, let the computer do the work!

Redefined Characters. The futuristic landing control tower and the rugged landscape were created with redefined characters, defined in the eighteen lines of DATA statements at the end of the program. Line 170 replaces the standard characters with the values in the DATA statements. The Introduction to this book and the Programmer's Notes for the game Space Crash offer further discussion of redefined characters.

If you would like to see the standard characters which were redefined to obtain the characters that appear on the screen in Rocket Lander, do the following: Turn the computer off and on again. Load the game in the usual manner but do not type RUN. Type 170 and press RETURN. (This will delete line 170.) Now type RUN and press RETURN. The screen will be displayed with standard instead of redefined characters, but otherwise, the game will run as usual.

Rocket Lander

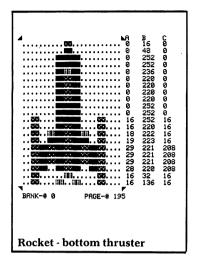
Hints for Modifying the Game

The variable K1 represents the vertical speed of the rocket. In line 440, the expression K1 < -.1 checks for the speed at landing time, -.1 being the maximum speed, allowed for a safe landing. Substituting other values for the .1 will make landing easier or more difficult.

The statement K1 = K1 - .1 in line 430 decreases the speed when the main engines are fired, and the statement K1 = K1 + .1 in line 437 simulates the force of gravity. Again, substituting other values for the .1 will change the difficulty of the landing.

The amount of fuel remaining is stored in the variable **PO**. The beginning fuel supply is set to 100000 in line 360 and can be changed to other amounts. Lines 410 and 420 decrease this amount by 100 when the side thrusters are fired. Line 430 decreases the amount by 500 when the main engines are fired. Changing these values will affect the fuel efficiency of the rocket.

You might want to modify the action of the side thrusters. The rocket moves sideways only when a thruster is fired and stops moving sideways when the thruster is released. Consider a method to keep the rocket's current horizontal momentum in effect until it is modified by further thruster firings.



Major Routines

0009-0025 INITIALIZATION

Generate landing pad sprite and set up array X\$ () with radar antenna characters.

0030-0060 PRINT TITLE SCREEN

Display title screen until sprite data is loaded into memory.

0070-0086 PRINT INSTRUCTION SCREEN

0090-0170 LOAD CHARACTER DATA

Lower top of BASIC memory. Copy standard character set into main memory and replace 18 of them with redefined characters. Point VIC chip to new location of character set.

0200-0320 PRINT BACKGROUND

Print background using regular and redefined characters.

Rocket Lander

0350-0370 INITIALIZE SPRITES

Set size, color, and location. Pick random location for landing pad. Set initial fuel supply.

0400-0470 MAIN GAME LOOP

Read current key pressed. If valid function key, select appropriate rocket sprite, decrease fuel supply, and adjust position of rocket. Prevent rocket from leaving screen to left or right. Add pull of gravity and check for good landing, crash, or out of fuel. Rotate radar antenna and print remaining fuel.

0500-0530 GOOD LANDING

Flash screen and make sound. Update good landing count and wait for any key pressed to begin again.

0600-0630 CRASH LANDING

Scroll screen, flash rocket sprites, and make sound. Update crash landing count, show "splat" rocket, and wait for any key pressed to begin again.

0700-0900 SOUND ROUTINES Three sound routines.

1000-1020 WAIT FOR KEY PRESSED

5000-5190 SPRITE DATA

5200-5360 CHARACTER DATA

Redefined characters. One per line.

Rocket Lander

Major Variables

General:

V Base address of VIC chip

SO Base address of SID chip

SL Address of hardware interrupt

SC Base address of screen memory

BA	Base address of relocated character set
AA	Current key pressed
V\$	String of cursor downs
H\$	String of cursor rights

Radar Antenna:

X\$() Character strings for animation TU Counter for animation

Rocket:

K5	Pointer to sprites
K1	Vertical movement
H1	Horizontal position
V1	Vertical position
CR	Number of crash landings
LN	Number of good landings
PΩ	Fuel

Sprites

Line Numbers	Page	Description
See notes	13	Landing pad
5000-5030	192	Rocket - no thruster
5040-5070	193	Rocket - right thruster
5080-5110	194	Rocket - left thruster
5120-5150	195	Rocket - bottom thruster
5160-5190	196	Rocket - splat

Redefined Characters

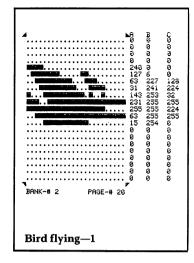
Line Number	Description
5200	Solid block
5210	Mountain part - 1
5220	Mountain part - 2
5230	Mountain part - 3
5240	Mountain part - 4
5250	Mountain part - 5
5260	Mountain part - 6
5265	Mountain part - 7
5270	Mountain part - 8

Rocket Lander

Rocket Lander

Squirrel Away

Frank Mikulastik



Susie Squirrel has gathered her winter supply of nuts from the trees, and now she needs to get them stored safely away. By aiming very carefully, she can drop them into holes in the ground which lead to her underground burrow. However, her timing must be just right, because the birds flying by will snatch them from the air if they get half a chance!

Control Susie's left and right movements with a joystick, and drop the nuts with the fire button. She has gathered 104 nuts — see how many you can successfully drop into her burrow.

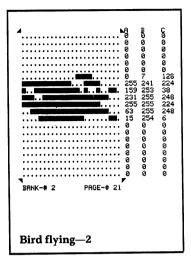
Load the game by typing LOAD"SQUIRREL AWAY",8 and pressing the RETURN key. After about 25 seconds, the computer will say READY, indicating that the game is loaded. Then type RUN and press the RETURN key. Susie will chatter at you from the title screen until you press the fire button and proceed to the instruction screen. Press the fire button again to start the game. When the game is over press the fire button to return to the instruction screen.

To stop the game, refer to the section in the introduction for stopping bank-switched games. This game uses memory bank 2 for sprites and graphics.

Programmer's Notes

Moving Sprites and Characters Together. This game design required 104 nuts, each of which must remain visible on the screen if it gets safely into the burrow.

Squirrel Away



Because the C-64 allows a limited number of sprites, the nuts were created as characters. The squirrel and birds, however, were created as sprites. (See the Introduction for a discussion of the differences between characters and sprites.)

Programming the squirrel carrying a nut presented an interesting problem. The squirrel and nut must move in unison, but the squirrel sprite is positioned with pixel coordinates, while the nut character is positioned with character coordinates. To make the squirrel appear to be carrying the nut as it moves, the squirrel's coordinates are changed by eight pixel units for each character unit by which the nut moves.

Making the motion appear natural when the squirrel turned around still remained a problem. This was solved by experimenting, which is part of the fun of developing video games. When you play the game, observe how this was worked out — when the squirrel turns in place, the nut jumps several character units to the other side of her.

Detecting Sprite-to-Character Collisions. When a moving sprite and a character come together on the screen and touch one another, a "sprite-to-character collision" occurs. Since the logic of a program often requires detection of these collisions, the Commodore 64's VIC chip has a special sprite-to-character collision register, located at memory location 52379 (sometimes referred to as the sprite-to-background collision register). The computer uses this register to reflect the current collision status of all the sprites.

Each sprite is monitored by one bit in the register — for example, bit 1 monitors sprite #1. Bit 1 ordinarily has a value of 0, but is set to a value of 1 when sprite #1 is involved in a collision. By reading the register as a binary number, the program can determine which sprites are currently involved in a collision. It is a good practice to save the register's contents in a variable, because once it is read, the register is automatically cleared.

In **Squirrel Away**, the sprite-to-character collision register is checked in line 2172 to detect a collision between any of the bird sprites and the falling nut.

Squirrel Away

Hints for Modifying the Game

Squirrel Away would be more difficult to play if the birds were larger or moved faster. Add a new line 305 reading 305 **POKEV** +29,254 to make the birds longer. Currently the speed of the birds is calculated by adding N*2+8 to BX(N) in line 3010. Try changing the 2 to other values.

The birds could be made to fly across the screen in some fashion other than a straight line. Be careful, however, not to let them hit a background character other than the nut, because the program would then detect a sprite-to-character collision and proceed as if a nut had been eaten. The birds may hit each other safely, because they are all sprites and there is no check in this program for sprite-to-sprite collisions.

You can make it easier to aim the nut by letting the nut remain stationary when the squirrel changes direction. Instead of rotating the squirrel in place and making the nut jump from one side to the other, make the squirrel jump to the opposite side of the nut.

Major Routines

0030-0340 INITIALIZATION

Set VIC memory to bank 2. Read sprite data into memory. Initialize variables and sprites. Print title, instruction, and play screens.

0500-0550 MAIN LOOP

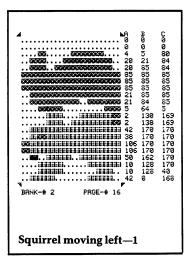
Move birds and read joystick. If fire button pressed, drop nut. If there are no nuts left to drop, end game. Move squirrel and nut.

0700-0790 INITIALIZE SPRITES

Set up starting positions for squirrel and birds.

0800-0890 PRINT PLAY SCREEN

Squirrel Away



1500-1550 MOVE SOUIRREL LEFT Calculate squirrel's new position, update sprite pointers, and make a sound.

1600-1650 MOVE SOUIRREL RIGHT Calculate squirrel's new position, update sprite pointers, and make a sound.

2000-2020 MOVE THE NUT Move nut with squirrel.

2100-2250 DROP THE NUT Move nut down the screen and move birds. Check for nut hitting ground, birds, or burrow. Decrease nut count.

2300-2395 NUT DROPS IN THE HOLE Move nut to the right of burrow and blink birds. If no nuts left, end game.

3000-3110 MOVE THE BIRDS Update positions and sprite pointers. If bird moves off screen, relocate on left.

4000-4010 NUT DROPS IN THE HOLE SOUND

4100-4110 NUT MISSED THE HOLE SOUND

4200 SQUIRREL MOVEMENT SOUND

4300-4320 NEW SCREEN SOUND

4400-4440 INITIALIZE SOUND

5000-5090 PRINT TITLE SCREEN

5100-5130 GAME IS OVER

5200-5290 PRINT INSTRUCTION SCREEN

6000-6950 SPRITE DATA

Squirrel Away

Major Variables

General:

V Base address of VIC chip
S1 Base address of SID chip
SM Base address of screen memory
CM Base address of color memory
M1 Base address of sprite pointers
BS() Sprite page numbers for birds

Squirrel:

SP() Sprite page numbers
D Direction of movement
SN Animation sequence counter
X X position in graphics mode
X1 X position in sprite mode
Y Y position in graphics mode
Y1 Y position in sprite mode

Nut:

T1 ASCII code for shape

T2 Color

TP Position in screen memory (1-1000)

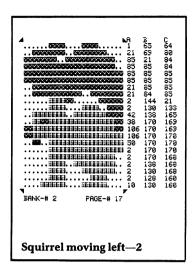
W1 Number left to drop

W2 Number dropped into burrow

Sprites

(NOTE: All sprites are in bank 2)

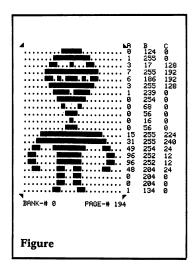
Line Numbers	Page	Description
6000-6130	16	Squirrel moving left - 1
6140-6270	17	Squirrel moving left - 2
6280-6400	18	Squirrel moving right - 1
6410-6540	19	Squirrel moving right - 2
6550-6680	20	Bird flying - 1
6690-6820	21	Bird flying - 2
6830-6950	22	Bird flying - 3



Squirrel Away

Ghost Hunt

Igor Tulchinsky



Three friends are out late at night trying to catch the neighborhood ghost. She is a slippery, tricky ghost, and in order to capture her, all three figures must touch her at once without touching each other.

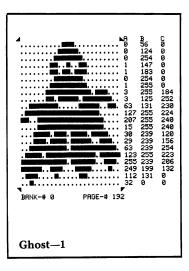
With a joystick, you control the movement of one figure while the other two stand still. The figure you are controlling is yellow, and you can switch to a different one by pressing the fire button. If you hold down the fire button while moving the joystick, all three figures will move in the same direction.

You are allowed a certain amount of time to catch the ghost. After you have caught her, you can play again, but each time she will be moving faster and the amount of time allowed to catch her will be shorter. You will accumulate points based on the length of time you take to catch each ghost. The game ends when the time runs out and the ghost has not been captured.

Load the game by typing LOAD"GHOST HUNT", 8 and pressing the RETURN key. After about 20 seconds, the computer will say READY, indicating that the game is loaded. Then type RUN and press the RETURN key.

Programmer's Notes

Detecting Sprite-to-Sprite Collisions. When two or more moving sprites come together on the screen and touch one another, a "sprite-to-sprite collision" occurs.



Since the logic of a program often requires detection of these collisions, the Commodore 64's VIC chip has a special sprite-to-sprite collision register at memory location 53278. The computer uses the register to store the current collision status of all the sprites.

Each sprite is monitored by one bit in the register – for example, bit 1 monitors sprite #1. Bit 1 ordinarily has a value of 0 but is set to a value of 1 when sprite #1 is involved in a collision. By reading the register as a binary number, the program can determine which sprites are currently involved in a collision. It is a good practice to save the register's contents in a variable, because once the register is read, it is automatically cleared.

In **Ghost Hunt**, the ghost is sprite #0. The figures are sprites #2, #3, and #4. When all of the sprites are involved in a collision, bits 0, 2, 3, and 4 will be set to 1, giving the collision register a value of 1+4+8+16, or 29. In line 210, the sprite-to-sprite collision register is read, and its contents are stored in variable Q1.

The game design specified that for the ghost to be captured, none of the three figures could be touching each other. This presented a programming problem, because when each figure is touching the ghost, each is involved in a collision regardless of whether it is touching another figure. This problem is solved in Line 231 of the program, in the following way:

- 1) The ghost sprite is turned off. As long as it is off, the sprite collision register will not detect its collisions.
- 2) The sprite collision register is cleared.
- 3) A delay of at least 1/60th of a second is executed, during which the system can detect any collisions between the figures and update the register.
- 4) The sprite collision register is read again. If no collisions are detected, then no figures are touching each other and the capture is valid. However, if any figure is touching any other, then the sprite collision register will read greater than zero and the ghost will not be captured.
- 5) The ghost sprite is turned back on and the program continues.

This all occurs so quickly that you never see the ghost disappear!

Hints for Modifying the Game

The statement IFCT = 40 in line 52 controls how long the ghost will stay in one position before heading off in a new direction. Change the 40 to a larger number to have the ghost remain in one spot longer. A smaller value will create a very slippery ghost!

Each time the fire button is pressed, the next figure in sequence is selected for movement. To select the next figure randomly when the fire button is pressed, change line 12 to read:

12 IF(NOTBS % AND16) = 16THENN1 % = N %: N % = INT(RND(8)*3+1)

You might want to try changing the distance (in pixels) by which the figures and the ghost move, as the score increases. This would best be done in the ADJUST SCORE routine.

Major Routines

00010-00040

MOVE FIGURE

Check fire button. If it has been pressed. switch figures by changing colors. Check joystick and move figure selected.

00051-00110

MOVE GHOST

Make a sound. Check whether counter has reached a certain value. If so, pick new spot for ghost to move toward. Move ghost toward hidden spot. Switch to other ghost sprite, causing animation. Make a sound.

00200-00206

INITIALIZATION

00210-00300

MAIN CONTROL LOOP

Move figure and ghost, and read sprite collision register. Print remaining time on screen. If less than ten seconds left, flash screen and sound alarm. If no time left, end game. If ghost has not been cap-

tured, start over. If ghost has been captured, increase score and speed, and resume.

SPRITE DATA 10000-10026

11000-11090 PRINT GAME SCREEN

Draw earth, sky, and stars while making

noises.

PRINT TITLE SCREEN 12000-12500

> Initialize sound. Draw border dots while making musical sounds. Print title. Read in sprite data for ghost. Reinitialize sound. Make ghost fly around, while making ghostly whine and waiting for

fire button to be pressed.

13000-13116 SPRITE INITIALIZATION

> Read sprite data into memory. Set color and other sprite parameters. Initialize

sprite-related variables.

SOUND INITIALIZATION 13120-13180

Clear sound chip. Define voices used.

Print "Score" at top of screen.

GAME OVER 15000-15040

> Make whine sound. Print farewell message. Wait until either fire button is pressed or 45 seconds have elapsed, then

restart game.

16000-16020 ADJUST SCORE

Print score after converting it to string

variable. Increase speed of game.

PRINT INSTRUCTION SCREEN 18000-18210

> Print instructions and wait for player to press fire button before starting game.

Major Variables

General:

V Base address of VIC chip S Base address of SID chip

BS% Joystick status

TZTime left

NP Score points awarded

SC Score

VZFrequency of sound SI Increment of frequency

MM Number of pixels moved by all sprites

VLTime allowed to catch ghost

Figure:

N% Number selected

CL Color

D3% Horizontal movement E3% Vertical movement X%() Horizontal positions Y%() Vertical positions

PX Address of horizontal position PY Address of vertical position

Ghost:

CT Counter for movement

X8% Horizontal position to head toward Vertical position to head toward Y8%

X3% Horizontal position

Y3% Vertical position

Horizontal change in position D% F% Vertical change in position

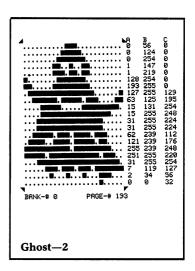
DX% Horizontal movement

DY% Vertical movement

M%() Magnitude of movement

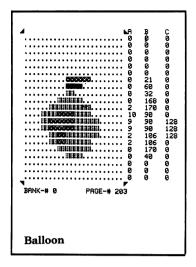
Sprites

Line Numbers	Page	Description
10000-10006	192	Ghost - 1
10010-10016	193	Ghost - 2
10020-10026	194	Figure



Firecracker Boy

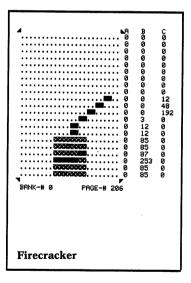
Charles Mott, Jr.

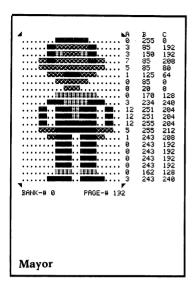


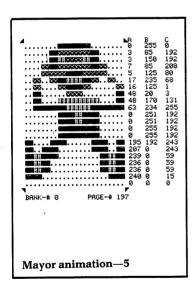
The Mayor's Fourth-of-July parade is in danger of interruption by a mischievous boy who is lighting firecrackers. The Mayor is on the reviewing stand watching the colorful floats and cars, and if he hears a firecracker explode, he will do a flip and stop the parade! You are a firefighter assigned to drop water balloons on the fuses of the firecrackers before they explode.

Control your movements left and right on the top floor of the blue Fire Station with a joystick, releasing the water balloons with the fire button. The net moving back and forth in front of the fire station may hamper your efforts by intercepting your balloons as they fall. The boy can trick you by hiding his firecrackers behind the door of the building, but you have your own tricks: you can drop balloons from behind the window pillars, and you can hit a firecracker even when it is behind a float. Points are scored for each firecracker you manage to put out.

Load the game by typing LOAD"FIRECRACKER BOY",8. Then press the RETURN key. After about 25 seconds the computer will say READY, indicating that the game is loaded. Then type RUN and press the RETURN key. After the title screen is displayed, the program will pause briefly before the game begins.







Firecracker Boy

Programmer's Notes

Two-Part Design to Handle Large Programs. To achieve its animation effects, Firecracker Boy uses a total of 33 sprites, including 10 Mayor sprites, 10 firecracker sprites, and 2 sprites for the mischievous boy. In addition to the great number of sprites, the features of the game required a very large program. To accommodate all this, the program was written in two parts. Part 1 begins by displaying the title and instruction screens, while it loads the sprite data from the program into the sprite pages. It then automatically loads and runs Part 2 (the actual game) from disk.

Line 670 of Part 1 loads one byte of sprite data into memory. In a number of different places throughout the routines which display the title and instruction screens, **GOSUB 670** instructions gradually load all the sprite data. After the sprites are loaded, line 710 protects them from being over-written by specifying a smaller portion of memory to be used for variable storage in Part 2.

Lines 720 and 721 use the keyboard buffer to load Part 2 of the game. See the Programmer's Notes for the game **Mission: Tobor** for an explanation of this special use of the keyboard buffer.

Using Variables to Control Processing Sequence. Often the sequence of processing in a program depends on whether or not a certain condition exists. One method for controlling program sequence is the use of variables as "flags". At a logical place in the program, a test is made for the condition, and a variable is set to indicate whether or not the condition exists. When the course of the game at some later time depends on that condition, the variable is checked.

Part 2 of **Firecracker Boy** uses the array **FG()** for flags to monitor various conditions which may occur during the game. Each element of the array is designated to indicate a particular condition. When the condition does not exist, or is *FALSE*, the flag is cleared, or set to 0. If the condition then becomes *TRUE* during the course of the game, the flag is set to a positive number.

For example, **FG(2)** is the "firecracker-on-screen" flag. It is set to *FALSE* when the boy walks onto the screen carrying a firecracker and set to *TRUE* when he drops it.

When a water balloon hits the firecracker, the flag is set back to *FALSE*. After the boy walks off the right side of the screen, the flag is continuously tested to see whether he should walk back onto the left side with a new firecracker. He will not reappear until a water balloon has put out the current firecracker and the flag is found to be *FALSE* once again.

The array **DR()** is used in a slightly different way to control sequence. The DATA statement at line 190 contains numbers coded to point to certain lines of the program. In lines 140-150, those numbers are loaded into **DR()**. Then in line 250, that array is used to determine what process is to be done next, based on the position of the joystick. The joystick port is read by a **GOSUB** to line 570, and the result is stored in the variable **JV**. If **JV** was set to 4, then the fourth element of array **DR()** is used to select the line in the program which is to be executed next.

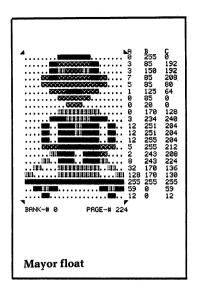
For more information about **ON GOTO** and **ON GOSUB** instructions, see the *Commodore 64 User's Guide*.

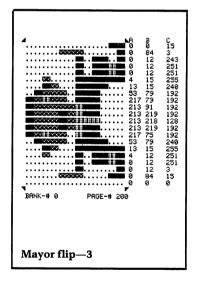
Hints for Modifying the Game

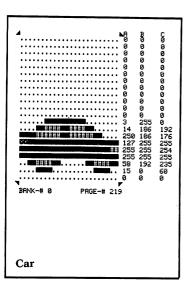
Part 1. Lines 380-440 use PRINT instructions to display the contents of arrays X\$(), Z\$() and C\$(), which are the word FIRE, the word CRACKER, and a picture of a fire-cracker. For some ideas on other ways of printing these arrays, see the hints for modifying the game Mission: Tobor.

An alternative way of writing **Firecracker Boy** would be to load the sprite data from a disk file instead of having it in **DATA** statements. For a programming exercise, try creating a disk file of all the sprite data and turning **Firecracker Boy** into a one-part program. Move the title and instruction screen routines from Part 1 to Part 2, and add a routine in Part 2 to read the sprite data from a disk file. This could be an involved process, but it would be a good lesson in creating and working with disk files.

Part 2. The variables PH and AP, which control the number of pixels by which each of the main figures moves, are initially set to 9 in line 70. Smaller initial



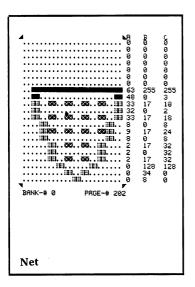




values will slow down the action and larger values will speed it up. However, take care not to make the values so high and the moves so large that collisions will not be detected properly.

PH controls how fast the net moves, and is incremented as your score increases. When you press the fire button to start the game over, PH is re-initialized to 9 in line 930. This means that each game will be relatively easy in the beginning, but will get harder as your score climbs. To make each game harder than the last, change PH = 9 to read PH = PH + 1 in line 930.

The variable EN determines how long it takes for the firecracker to pop after it is dropped. In line 1560, the statement EN = EN - 2 has the effect of shortening the fuse as the score increases. Consider changing the value of EN in other ways as the game progresses.



Major Routines

PART 1

0080-0360 INITIALIZE VARIABLES

Begin loading sprite data into memory and set up title screen array variables.

0370-0470 PRINT TITLE SCREEN

> Print out title screen arrays as more sprite data is loaded into memory.

0480-0490 INITIALIZE FALLING SOUND

0500-0520 FALLING SOUND

0530-0660 PRINT INSTRUCTION SCREEN

0670 POKE 1 BYTE OF SPRITE DATA

Read and store sprite data until last byte

(with value of 256).

LOAD PART 2 0700-0720

> Start long sound. Lower top of memory. Load keyboard buffer and use contents to load Part 2.

PART 2

0050-0200 INITIALIZE VARIABLES
Set up joysticks and flag arrays. Initialize sprites.

0240-0530 MAIN GAME LOOP

Move boy and drop off firecracker at a random spot. Move firefighter, net, float, and water balloon. Check for balloon hitting firecracker. Check for end of game.

0570 READ JOYSTICK

0610 NEW BALLOON

Give a new balloon to firefighter: Turn off balloon sprite. Then, using firefighter's current position for new position of balloon, turn balloon sprite back on.

0650-0760 INITIALIZE SPRITES
Set position, color, and size.

0800-0960 GAME END

Firecracker burns down and explodes. Mayor does a flip.

1000-1240 PRINT GAME SCREEN

1280-1350 COLLISION DETECTION

Check for balloon hitting net or firecracker.

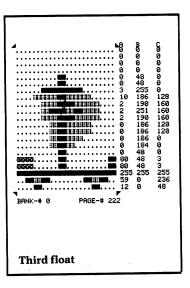
1410-1440 SOUND ROUTINES

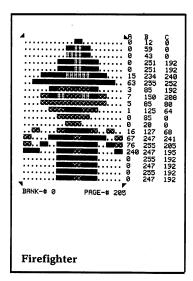
Background sound, splash sound, and falling sound.

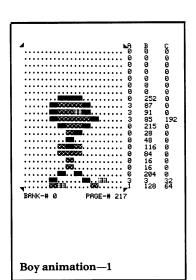
1480-1510 MOVE FLOAT ACROSS SEAM

Move float sprites past horizontal position
255.

1550-1590 PRINT SCORE
Speed up net, shorten fuse, and print score.







Firecracker Boy

Major Variables

PART 1

S	Base address of SID chip
Ö	Pitch of falling sound
X\$()	Title word <i>FIRE</i>
Z\$()	Title word CRACKER
C\$()	Title picture of a firecracker

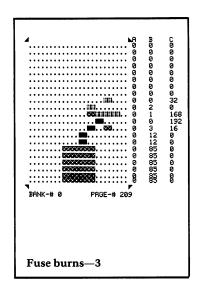
PART 2

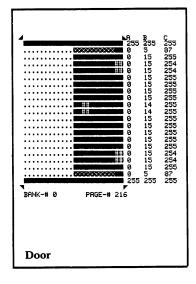
V	Base address of VIC chip
SC	Base address of screen memory
JS	Address of joystick port
BR	Address of border color
FG()	Flags
DR()	Joystick direction array
S1-S8	Sprite pointers
EN	Length of fuse
PH	Pixels moved by net
N	Position of net
XX	Pixels moved by float
AU	Position of float
HH	Pixels moved by boy
ВО	Position of boy
AP	Pixels moved by firefighter
MX	Position of firefighter
FC	Position of firecracker
BP	Score
TS	Top score

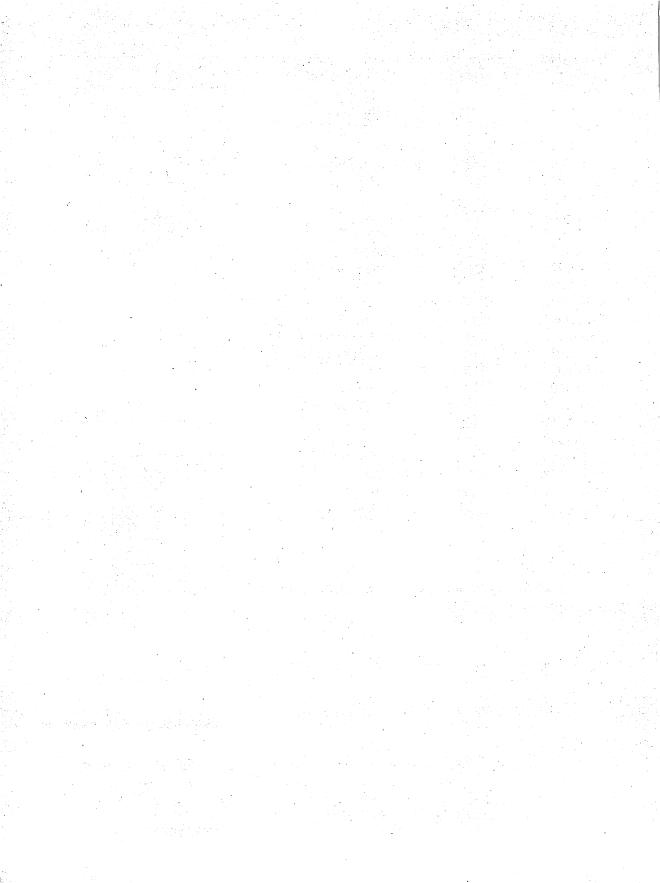
Sprites

Line Numbers	Page	Description
5000-5003	192	Mayor
5004-5007	193	Mayor animation - 1
5008-5011	194	Mayor animation - 2
5012-5015	195	Mayor animation - 3
5016-5019	196	Mayor animation - 4
5020-5023	197	Mayor animation - 5
5024-5027	198	Mayor flip - 1
5028-5031	199	Mayor flip - 2

5032-5035	200	Mayor flip - 3
5036-5039	201	Mayor flip - 4
5040-5043	202	Net
5044-5047	203	Balloon
5048-5051	204	Balloon - popped
5052-5055	205	Firefighter
5056-5059	206	Firecracker
5060-5063	207	Fuse burns - 1
5064-5067	208	Fuse burns - 2
5068-5071	209	Fuse burns - 3
5072-5075	210	Fuse burns - 4
5076-5079	211	Fuse burns - 5
5080-5083	212	Fuse burns - 6
5084-5087	213	Fuse burns - 7
5088-5091	214	Firecracker explosion - 1
5092-5095	215	Firecracker explosion - 2
5096-5099	216	Door
5100-5103	217	Boy animation - 1
5104-5107	218	Boy animation - 2
5108-5111	219	Car
5112-5115	220	First float
5116-5119	221	Second float
5120-5123	222	Third float
5124-5127	223	Fourth float
5128-5131	224	Mayor float

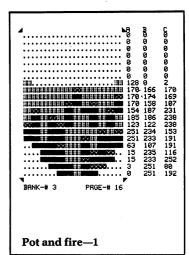






Into the Pot

Frank Mikulastik



The cooking pot is boiling, but you haven't yet caught your dinner! You had planned to prepare your favorite recipe, which calls for two fat chickens, and two succulent-looking specimens are roaming around the rocky plain next to your camp. How quickly can you get them into your pot?

The blue blocks are heavy boulders which you cannot move, but you can push the lightweight patterned rocks around to create a path into the camp and steer the chickens toward the pot. Just in front of your camp there is a ready-made canyon entrance — if you can get them into the canyon, they're as good as in the pot!

Since it could take a while to round up the chickens, you might want to take a snack break and come back to the chicken chase a little later. Pressing the F7 key will pause the game, saving your elapsed time. By entering Y to continue, you can pick up where you left off.

If you're not careful, you may trap yourself or a chicken in such a position that the game cannot possibly be won. In that case, pressing the **F7** key and entering **N** is the only way to quit and start a new game.

Load the game by typing LOAD"INTO THE POT",8 and pressing the RETURN key. After about 20 seconds, the computer will say READY, indicating that the game is loaded. Then type RUN and press the RETURN key. After a delay of about 15 seconds, the title will flash on the screen. Press the fire button to proceed to the instruction screen. After you win a game, pressing the fire button will return you to the instruction screen.

Into the Pot

Programmer's Notes

Variables with Fixed Values. Experienced programmers often use variables to represent numbers that are fixed in the program and are not expected to change. One reason is to plan ahead for future program modifications. Although a particular number will never change in the current version of the program, assigning it a variable name makes it easier to find and change in future versions.

When a value is used in many different places in the program, assigning it to a variable name is even more important. Changing the value of a variable in one place in the program is simple, compared with the chore of tracking down and changing a numeric value which occurs throughout the program.

In Into the Pot, the number of blocks the cook can push at one time is set with the statement NM = 2 in line 780. The variable NM does not change values as the game is played, and it is used only once, in line 1130. The statement FORI = 1TONM in line 1130 could have read FORI = 1TO2 with the same result. You might decide later, however, to modify the game by enabling the cook to push a bigger stack of rocks as the game progresses. Having already defined the variable NM would simplify that change. The simplification would be even greater for a variable that is used several times throughout the program.

Often variable names are used simply to make the program instructions more meaningful. For example, it is common practice to include the instruction V = 53248 to define the variable V as the base address of the VIC chip (53248). Then it can be referenced as V, rather than as 53248.

Into the Pot

Hints for Modifying the Game

To have the cook character start in a different position, change the number assigned to **P1** in line 700. A value of 0 would place it at the upper left-hand corner of the screen, and a value of 1000 would place it at the bottom right-hand corner. The starting positions of the chickens can be

changed in a similar way by assigning different values to **P2(1)** and **P2(2)** in line 720.

Try adding more chickens to the game. For each additional chicken, add a statement to line 720, setting the next element in the **P2()** array to the new chicken's desired starting position. For example, use **P2(3)** to establish the third chicken's initial position. Below line 740, add a line similar to lines 730 and 740 but using **P2(3)**. Then change the value of **NU**, set in line 780, to the new total number of chickens.

Major Routines

0013-0350 INITIALIZATION

Set up new character set. Set VIC memory to bank 3. Turn on new character set. Turn on multicolor character mode. Initialize variables. Print title screen and instruction screen.

0400-0420 START GAME

Print play screen. Initialize cook, chickens, and timer.

0500-0590 MAIN LOOP

Move cook and make a sound. Move chicken. Check to see if in no-win situation. If so, save timer and do NO WIN SITUATION. Repeat loop.

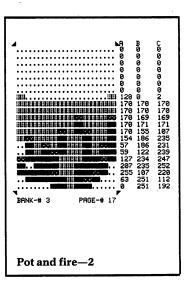
0700-0790 INITIALIZE CHARACTERS

0900-0980 PRINT PLAY SCREEN

Print borders, blocks and box. Blank out blocks in box. Print timer.

1000-1290 MOVE COOK

Read joystick. Set cook's direction accordingly and see if move possible. If no movement requested, return to MAIN LOOP. If location in front of cook is blank, move cook. Otherwise, return to MAIN LOOP. If cook is in front of a block, check to see if it can be moved. If it can, move block and cook, and increment the animation counter.



Into the Pot

MOVE CHICKEN 1500-1690

Get a random number between 1 and 4 for direction. If direction selected in front of chicken is blank, move it and increment its animation counter.

WIN SITUATION 2000-2091

Make a sound. Check for fire button pressed to start a new game. If fire button pressed, print instruction screen and start a new game.

2100-2190 NO-WIN SITUATION

Check for keyboard input of Y or N to stop game or not. If game is to be stopped, print instruction screen to start another game. If game is to continue, reset timer and return to MAIN LOOP.

3000-3060 PRINT TITLE SCREEN

Make a sound and clear the screen. Switch character colors between red and yellow, while making a sound and checking for fire button to continue.

3100-3146 PRINT INSTRUCTION SCREEN

Clear screen and set background and border colors and make a sound. Wait for F1 or F7 to be pressed. If F1, continue with game. If F7, stop game.

4000-4048 SOUND ROUTINES

Five separate sound effects.

4100-4220 CHARACTER DATA

One redefined character per line.

5140-5400 SPRITE DATA

Into The Pot

Major Variables

Memory Pointers:

V Base address of VIC chip

S1 Base address of SID chip CM Base address of color memory SM Base address of character memory U Base address of sprite memory M1 Base address of sprite data

Cook:

S() Animation array N1 Animation counter P1 **Position Counter** T1 X direction movement **T2** Y direction movement

Chickens:

S2() Animation array N2() Animation counters P2() **Positions** BN() Directions to move NU Number of chickens

Pot:

SP Animation counter

Sprites

Line

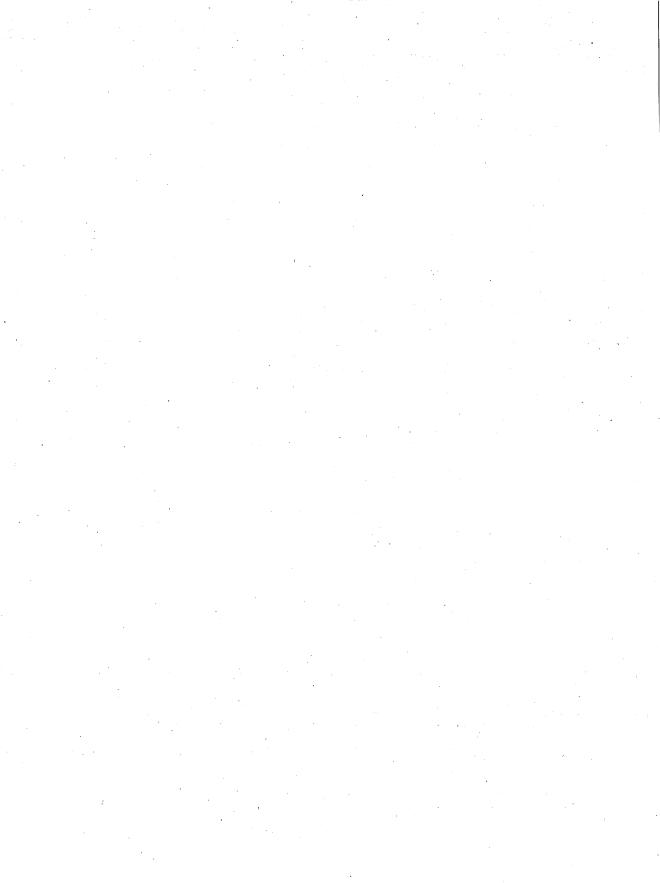
(NOTE: All sprites are in bank 3)

Line Numbers	Page	Description
5140-5270	16	Pot and fire - 1
5280-5400	17	Pot and fire - 2

Redefined Characters

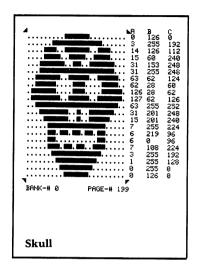
LIIIC	
Number	Description
4100	Chicken - 1
4110	Chicken - 2
4120	Chicken - 3
4150	Cook - 1
4160	Cook - 2
4170	Cook - 3
4200	Block pattern - 1
4210	Block pattern - 2
4220	Block pattern - 3

Into **The Pot**



Yorick's Revenge

Igor Tulchinsky



Yorick's Revenge is a video game patterned after the action of a pinball machine. The object is to toss the ball into the high scoring areas, using the flippers located at the bottom of the screen, and to keep the ball in play for as long as possible.

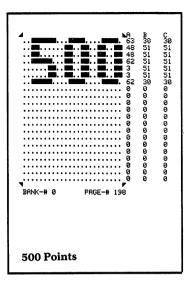
Why is the game called **Yorick's Revenge**? When the ball crosses over the skull of Yorick, in the center of the screen, it departs in a new direction which appears to be random. But actually, as your score increases, Yorick will direct the ball closer to the gutter between the flippers, making it more and more difficult to keep the ball in play. Yorick will get his revenge!

The flippers are controlled by pressing any key on the keyboard. When the flippers are down, the ball bounces off them as if it had hit the bottom of the screen. When the flippers are up, the direction and speed of the ball's bounce depends on which part of the flipper hits the ball, as well as the ball's previous speed and direction.

Points are scored when the ball passes over the 100 and 500 point markers. The ball is lost if it drops in the gutter between the flippers. When three balls are lost, the game is over. Pressing the fire button will then start a new game.

Load the game by typing LOAD"YORICK'S

Yorick's Revenge



REVENGE",8 and pressing the RETURN key. After about 20 seconds, the computer will say READY, indicating that the game is loaded. Type RUN and press the RETURN key. About 10 seconds later, the title screen will appear, and then the game will begin. When the game is over, press the space bar to play again.

Programmer's Notes

Programming for Speed. Achieving reasonable speed in a game written in BASIC can be a major problem. Because of the nature of the BASIC language, many good game ideas become dull games to play because of lack of speed. However, BASIC is an attractive first language for a beginning programmer, it does have some advantages, and often it is the only language available! Here are some pointers for overcoming the speed problems.

The first trick is to organize the frequently processed steps into a main control loop along with tests for conditions which occur infrequently. When such conditions are detected, handle them with program logic outside the loop. For example, the main loop in this program just moves the ball and checks for collisions. The bouncing of the ball, scoring, and other less frequent events are controlled elsewhere.

The main control loop should be placed toward the top of your program, for two reasons. This placement helps the program's speed, since the first lines in a BASIC program are executed more quickly than those toward the bottom. For purposes of good program structure, it makes sense to place the main control loop at the top where it can be found easily. Instruction screens, sprite data, character definitions, and other routines that are rarely executed should be placed at the end of the program.

Putting several statements on a line can speed things up a little, but it will also make the program harder to read, understand, debug, and modify. Concentrate your efforts on efficient program logic and organization.

Yorick's Revenge

Hints for Modifying the Game

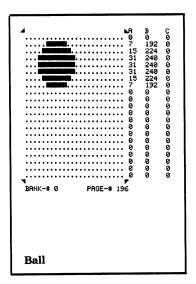
To make **Yorick's Revenge** more challenging by decreasing the size of the ball, change line 20070 to read:

20070 POKE V + 29,251:POKE V + 23,120:POKE V + 16,0

This will turn off vertical and horizontal expansion for the ball sprite. Or instead, test the score in the POINTS COLLISION routine, and make the ball become smaller when a certain score is reached. Similar approaches can be used on all the other sprites.

The force of gravity controlling the ball can be changed by adjusting the amount added to DY in line 42.

You could add more interest by making Yorick and/or the 100 and 500 point images move around on the screen. The instructions to make them move should be placed in the scoring routine or the sprite collision routines, to minimize loss of ball speed.



Major Routines

00010-00016 INITIALIZATION

Initialize all variables and prepare for

new game.

00020-00051 CONTROL LOOP

Check to see if a key is pressed, and choose appropriate sprites for flippers. Advance ball and test if collision has occurred. If no collision, go back to

beginning of loop.

00060-00140 SPRITE-BACKGROUND COLLISION

Determine whether ball has hit a bumper by checking if it is adjacent to a non-space character. If so, sound beep

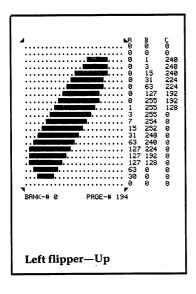
and bounce ball.

00150-00156 SKULL COLLISION

Sound hiss, then choose new direction

for ball based on score.

Yorick's Revenge



00160-00195 POINTS COLLISION

> Sound high beep. Decide if 100 or 500 points have been hit. Adjust score.

00200-00231 **GUTTER COLLISION**

> Sink ball into gutter while making sound. If no balls left, end game. Otherwise, adjust number of balls remaining and

pitch new ball.

00235-00260 FLIPPERS DOWN COLLISION

Make sound, then bounce ball.

FLIPPERS UP COLLISION 00270-00380

> Check which flipper, and which part of flipper, was hit. Calculate ball speed and direction. Bounce ball, adjusting direction and speed based on previous speed

and place which it hit on flipper.

10000-10086 SPRITE DATA

SPRITE INITIALIZATION 20000-20130

> Read sprite data into memory. Define sprite size, placement, and other characteristics. Then position all sprites 40

units to left to center them.

BACKGROUND INITIALIZATION 21000-21900

> Draw pinball game, first by using POKE statements to load data directly into screen memory and then by printing the

rest.

22000-22040 SOUND INITIALIZATION

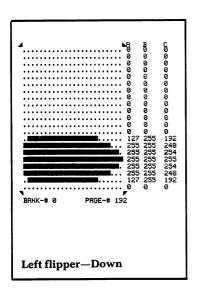
Initialize three voices.

23000-23090 **END OF GAME**

> Sound Yorick's laugh. If flag is set, return, because routine is being accessed for the first time. If flag is not set, wait for fire button to be pressed and start new

game.

24000-24030 PRINT TITLE



Yorick's Revenge

Major Variables

General:

SC Score

NL Number of balls left S\$ String value of SC

N\$ String value of NL

F Flag

Z Sprite collision flag

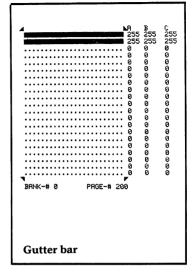
V Base address of VIC chip

Ball:

X Horizontal pixel coordinateY Vertical pixel coordinate

DX Horizontal movement in pixelsDY Vertical movement in pixelsT Angular direction in radians

L Speed



Sprites

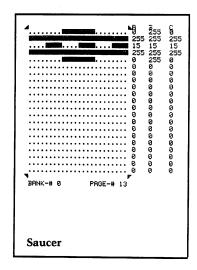
Line Numbers	Page	Description
10000-10006	192	Left flipper - Down
10010-10016	193	Right flipper - Down
10020-10026	194	Left flipper - Up
10030-10036	195	Right flipper - Up
10040-10046	196	Ball
10050-10056	197	100 points
10060-10066	198	500 points
10070-10076	199	Skull
10080-10086	200	Gutter bar

Yorick's Revenge



Space Crash

Charles Mott, Jr.



Swarms of purple aliens are invading your planet, appearing without warning from hyper-space and floating to the ground. You must hold them off for as long as possible by intercepting and destroying them with your saucer. Each alien hit is worth ten points. When thirty-two aliens have landed, they will have taken over the planet by filling up the landscape, and the game will end.

If a falling alien encounters a tree or other object, it will be destroyed, taking with it whatever it hit. Eventually, this will clear the way for other aliens to reach the ground safely. To maximize your score, try especially hard to intercept those whose paths have already been cleared.

You will have maximum maneuvering room if you stay high in the sky, but this approach can be dangerous. The aliens are appearing from hyper-space in a band across the top half of the screen. If one comes in at the same spot where your saucer is at that moment, the saucer will be destroyed. You can also lose the saucer by flying it into the ground as you try to catch an alien low on the screen. Five saucers are available for your use. When they are all destroyed, the game is over.

Control the saucer with the cursor movement keys. Pressing the CRSR- ‡ key will move the saucer down,

while SHIFT-CRSR- t will move it up. To move it left and right, use CRSR- → and SHIFT-CRSR- →.

Load the game by typing LOAD"SPACE CRASH",8 and pressing the RETURN key. After about 20 seconds, the computer will say READY, indicating that the game is loaded. Then type RUN and press the RETURN key. The title screen will be displayed for about 30 seconds, before the saucer flies across the screen to begin the game.

Programmer's Notes

Redefined Characters. None of the graphics shapes available among the standard Commodore 64 graphics characters were suitable for some of the objects in this game, so special graphics shapes were customized by use of redefined characters. Setting up redefined or custom characters in a program involves many programming steps.

Every character or graphic symbol displayed on the screen is defined by a series of eight bytes. The eight-byte definitions of the C-64's standard character set are stored in the computer's read-only memory (ROM). To redefine them, first you must copy the regular character set into an area of memory where it can be modified. Then for each character that is to be redefined, the eight bytes in the appropriate place in the relocated character set must be replaced with other bytes that will create the desired new shape.

The method used in Space Crash to make the replacements can be a timesaver and can help maintain flexibility for future changes. The bytes used to redefine the characters are in DATA statements beginning at line 1380. Each line contains nine numbers, but only the last eight are used for character redefinition. The first number is the relative position, within the character set, of the character being replaced. (See appendix E of the Commodore 64 User's Guide for the numeric position of each character.)

Lines 310 and 320 actually do the redefinition, storing the last eight numbers in each DATA statement into the position in the relocated character set that was indi-

cated by the first number. The last DATA statement contains only a -1, which indicates the end of the redefinition data. In line 310, the statement IFA = -1 THEN330 stops the moving of data when a -1 is found.

Using this method, new lines of DATA statements for new custom characters can be added or deleted, with no concern for their position within the other DATA statements, and existing DATA statements can be reassigned to different characters. The only requirement is that the last DATA statement contain a-1.

This is just a summary of the method used for character redefinition in this game. To learn all the details of the procedure, you should consult a book on C-64 graphics programming.

Hints for Modifying the Game

The total number of aliens on the screen at once is controlled by the variable **NA**, set to 5 in line 40. Changing the 5 to a larger number will increase the difficulty of the game. Or instead, increment **NA** as the score increases. Too large a number for **NA**, however, will make the game move very slowly.

The saucer's speed in all four directions is set in lines 520-550. The size of the saucer's moves, for any cursor key pressed, is set to 8 pixel units for each cursor key pressed. Change some or all of these 8's to make the saucer move at different speeds when it goes in different directions. For example, the saucer could move down quickly but rise up very slowly. In order for the saucer/alien collision routines to work correctly, all values must be multiples of 4.

Currently, no matter how many aliens are prevented from landing, more will keep appearing. You might want to consider having a fixed number of aliens to contend with, and award a bonus if they are stopped successfully.

If you prefer to control the saucer with a joystick instead of the cursor keys, change lines 520-550, using lines 1000-1050 in the game **Into the Pot** as a model.

Major Routines

INITIALIZATION 0030-0336

Initialize variables and sound. Print title screen. Set color and position of saucer sprite and load its data into memory, setting unused bytes in the page to 0. Lower top of BASIC memory. Load new character set if not already present. Do title animation and print game screen.

SPIN SAUCER 0370-0380

Make saucer sound. Update sprite memory with new shape.

MAIN LOOP 0510-0570

Check for keyboard input and adjust saucer's position. Descend an alien and spin saucer. Check for saucer hitting ground or an alien. Move saucer.

SAUCER CRASH 0660-0711

Lower saucer as its shape data in memory is being randomly changed. Re-initialize saucer and check for end of game.

0740-0775 SAUCER HIT ALIEN

Determine which alien was hit. Erase alien and adjust score.

0780-0830 NEW ALIEN APPEARS

Pick random position, set position variables, and display alien.

0850-0890 ALIEN DESCENDS

If object at ground level was hit, erase it. If ground was hit, display landed alien and update score. Check for end of game. Make new alien appear.

Space Crash

0910-1120 PRINT GAME SCREEN

1160-1280 PRINT TITLE SCREEN

1300-1315 END OF GAME

Move saucer and print game over message.

1330 SPRITE DATA

1380-1670 CHARACTER DATA

Major Variables

General:

V Base address of VIC chipSO Base address of SID chip

SC Base address of screen memory CL Base address of color memory

BA Base address of relocated character set

BR Address of border color

BK Address of background color

SR Score

XP\$ String to position cursor on X axis YP\$ String to position cursor on Y axis

IN Address for line of blocks at bottom of screen

BL Blocks left to fill

Saucer:

FS() Addresses of sprite data to change for spin

GA Value for spin - 1 AG Value for spin - 2

U Pixels moved in Y directionVE Pixels moved in X directionOM\$ Saucer shape for score

OM Pointer to score saucers

Aliens:

NA Number allowed on screen at one time

AA\$ First appearance shape

WA() Redefined characters for animation

AC Animation counter

A() Current position in screen memory B() Equivalent X positions in pixels

C() Equivalent Y positions in pixels

RX Random X position

RY Random Y position

Sprites

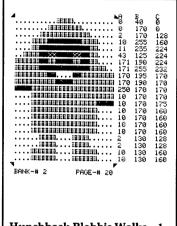
Line Number	Page	Description
1330	13	Saucer

Redefined Characters

Line Number	Description
1380	Solid block
1500	Top of tree - 1
1510	Top of tree - 2
1520	Asterisk
1530	Alien - 1
1540	Alien - 2
1560	Alien on ground
1570	Alien - 3
1580	Alien - 4
1640	Score saucer - 1
1650	Score saucer - 2
1660	Score saucer - 3

The Blobbis

Frank Mikulastik



Hunchback Blobbis Walks - 1

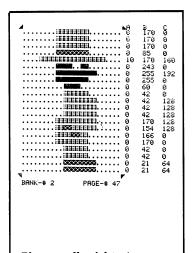
quipped with the latest and most fashionable fighting tuxedo and shooting hat, you are out to break through the force field that protects three evil Blobbis. The force field has five levels, and each shot from your top hat will destroy a section of one level.

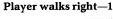
The different force fields move in opposite directions, so when you shoot through one and create a hole in the next one, the two holes will soon drift apart. Opening up a hole through all five levels just as a Blobbis passes over requires clever strategy!

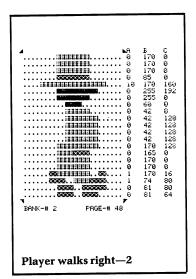
Each Blobbis has a different method of preventing you from penetrating the force field. The most dangerous is the Hunchback Blobbis, who drops a deadly pulsar from the center of the screen on each trip across the screen. If it hits you, it's all over! The Bug-eyed Blobbis frustrates you by rebuilding the top two layers of the force field on each trip. And the Flashing Blobbis makes a confusing and hypnotizing display.

Hitting any one of the Blobbis will win the game for you. As your strategy improves, you should be able to do it with fewer shots. Use the joystick to move the player sprite, and press the fire button to fire your hat.

Blobbis







The Blobbis

Load the game by typing LOAD"THE BLOBBIS",8 and pressing the **RETURN** key. After about 50 seconds the computer will say READY, indicating that the game is loaded. Then type RUN and press the RETURN key. After the title screen is displayed for about 20 seconds, the instruction screen will appear. When the game is over, pressing the fire button will take you back to the instruction screen.

To stop the game, see the special instructions in the Introduction for stopping bank-switched games. This game uses bank 2 for sprites and graphics.

Programmer's Notes

Animation Using Arrays. The main figures in The Blobbis are animated with a procedure which uses animation arrays. The explanation of the procedure is involved. Before continuing, you may wish to refresh your knowledge of sprites by reviewing that section of the Introduction.

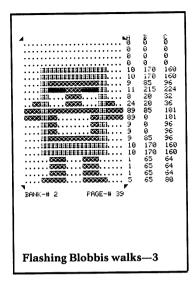
The animation array for each sprite object contains the sprite page (memory) locations, or page numbers, in the sequence in which the different sprites will appear when the figure is animated. A particular page number can be used more than once during an animation sequence.

For example, the animation array for the walking Bug-eyed Blobbis is S3(). In line 170, the sprite page numbers for each of the eight sprites used in that particular animation sequence are stored into S3(1) through S3(8). B3, which is used to indicate the number of elements in the array S3(), is set to 8 in line 175. N3 is used to indicate which step in the animation sequence is currently being executed.

Remember, the sprite page pointers are at fixed locations in memory which are accessed by the VIC chip when displaying sprites. In The Blobbis, the variable M1 contains the location of the first sprite page pointer, which points to the shape data for sprite #0. Since only one Blobbis is on the screen at any particular time, sprite #1 is used for all Blobbis sprite objects. The sprite page pointer for sprite #1 is one memory location after that for

sprite #0, or M1+1. At all times, that location will contain the page number to be used for the next Blobbis to be displayed.

Now look at lines 4150 and 4160 in the MOVE BUG-EYED BLOBBIS section. First N3, the current sequence step counter, is incremented. If it becomes larger than B3, the sequence element counter, it is reset to 1 to begin a repeat of the sequence. Then sprite pointer location M1+1 is loaded with the contents of the element of the animation array indicated by the current value of N3. M1+1 will now contain the page number of the next sprite in the sequence. Each time these instructions are executed, the next sprite in the walking sequence for the Bug-eyed Blobbis is displayed.



Hints for Modifying the Game

The speed of each figure in the game can be changed in the following way: The position of each Blobbis is incremented by 6 pixels at the beginnings of lines 4010, 4120, and 4210. Change the +6 in these lines to other values to make them move more quickly or slowly. Lines 4000, 4100, and 4200 determine when each Blobbis does its special trick by monitoring its location. The amount of movement for the player sprite is set to +9 or -9 in lines 1040 and 1050, depending on the position of the joystick. Change these values for a faster or slower player sprite.

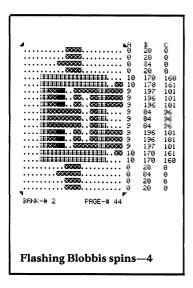
Try adding an additional Blobbis. Using lines 4100-4190 as a model, set up new animation arrays and major variables. You might consider having your new Blobbis rebuild different rows of the force field.

Major Routines

0010-0430 INITIALIZATION

Set up memory variables. Set VIC memory to bank 2. Read sprite data and make sound. Set up sprite animation arrays. Print title and instruction screens. Build force field and initialize sprites.

The Blobbis



0500-0590 MAIN LOOP - NOT SHOOTING Move player and current Blobbis. Check for game over. Select next force field row and move it.

0700-0790 INITIALIZE SPRITES Set starting positions.

0800-0860 BUILD FORCE FIELD

0865-0890 BUILD FLOOR

1000-1090 PLAYER MOVEMENT DRIVER Read joystick. If fire button pressed, begin MAIN LOOP - SHOOTING. If joystick moved, do MOVE PLAYER.

MOVE PLAYER 1100-1190 Change pointer to animation array. If player not at edge of screen, move player.

2000-2090 MOVE ROW TO THE RIGHT Move one row of the force field to the right.

2100-2190 MOVE ROW TO THE LEFT Move one row of the force field to the left.

3000-3090 MAIN LOOP - SHOOTING Main in control loop for player shooting. Fire the bullet and move it until it hits something. Move a Blobbis and check for its being hit.

MOVE HUNCHBACK BLOBBIS 4000-4090 Do special action if in correct position. Change pointer to animation array. If at edge of screen, select next Blobbis.

4100-4190 MOVE BUG-EYED BLOBBIS Do special action if in correct position. Change pointer to animation array. If at edge of screen, select next Blobbis.

4200-4290 MOVE FLASHING BLOBBIS Do special action if in correct position.

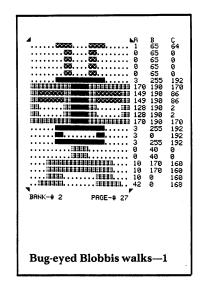
Blobbis

Change pointer to animation array. If at edge of screen, select next Blobbis. 5000-5091 PRINT TITLE SCREEN 5100-5180 PRINT INSTRUCTION SCREEN 5200-5290 GAME OVER Make a sound while waiting for fire button to start a new game. 6000-6004 SCREEN CHANGE SOUND 6010-6013 BLOBBIS MOVEMENT SOUND 6020-6026 INITIAL BULLET SOUND 6030-6036 A BLOBBIS IS HIT Flash and make a sound. 6040-6046 FORCE FIELD SOUND 6050-6056 PLAYER WALKING SOUND 6060-6069 MOVING BULLET SOUND 6080-6089 GAME OVER SOUND 6100-6119 BUG-EYE ACTION Dances and fills in top 2 rows of force field. 6200-6290 FLASHER ACTION Flashes the screen and does some flips. 6300-6390 HUNCHBACK ACTION Drops the pulsar and does a dance. Check for pulsar hitting player.

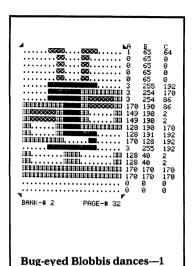
6400-6490 PULSAR HITS PLAYER

7000-12310 SPRITE DATA

Player flashes and sound.



The Blobbis



Major Variables

Memory Pointers:

V Base address of VIC chipS1 Base address of SID chip

M1 Base address of sprite pointers
SM Base address of screen memory
CM Base address of color memory

Player:

S1() Animation array for walking B1 Length of walk sequence

D Direction of movementX X position

N1 Animation counter

Hunchback Blobbis:

S2() Animation array for walking
B2 Length of walk sequence
N2 Walk animation counter
D2() Animation array for dance
C2 Length of dance sequence
N6 Dance animation counter
P2 X position

Bug-Eyed Blobbis:

S3() Animation array for walking
B3 Length of walking sequence
N3 Walk animation counter
D3() Animation array for dance
C3 Length of dance sequence

P3 X position

Flashing Blobbis:

S4() Animation array for walking B4 Length of walking sequence N4 Walk animation counter D4() Animation array for dance C4 Length of dance sequence

P4 X position

The Blobbis

Pulsar:

S5() Animation array for pulsing B5 Length of pulse sequence N5 Animation counter

P5 Y position

General:

WS Which Blobbis is on screen

F Game status

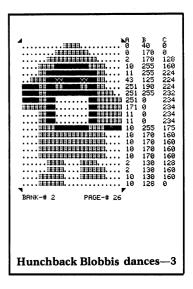
K Which row of force field to move

BL Number of bullets fired

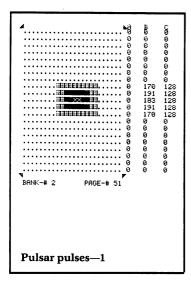
Sprites

(NOTE: All sprites are in bank 2)

Line Numbers	Page	Description
07550-07670	20	Hunchback Blobbis walks - 1
07680-07810	21	Hunchback Blobbis walks - 2
07820-07950	22	Hunchback Blobbis walks - 3
07960-08080	23	Hunchback Blobbis walks - 4
08090-08220	24	Hunchback Blobbis dances - 1
08230-08350	25	Hunchback Blobbis dances - 2
08360-08490	26	Hunchback Blobbis dances - 3
08500-08630	27	Bug-eyed Blobbis walks - 1
08640-08770	28	Bug-eyed Blobbis walks - 2
08780-08900	29	Bug-eyed Blobbis walks - 3
08910-09040	30	Bug-eyed Blobbis walks - 4
09050-09170	31	Bug-eyed Blobbis walks - 5
09180-09310	32	Bug-eyed Blobbis dances - 1
09320-09450	33	Bug-eyed Blobbis dances - 2
09460-09580	34	Bug-eyed Blobbis dances - 3
09590-09720	35	Bug-eyed Blobbis dances - 4
09730-09850	36	Bug-eyed Blobbis dances - 5
10000-10130	37	Flashing Blobbis walks - 1
10140-10270	38	Flashing Blobbis walks - 2
10280-10400	39	Flashing Blobbis walks - 3
10410-10540	40	Flashing Blobbis walks - 4
10550-10670	41	Flashing Blobbis spins - 1
10680-10810	42	Flashing Blobbis spins - 2
10820-10950	43	Flashing Blobbis spins - 3
10960-11080	44	Flashing Blobbis spins - 4
11090-11220	45	Flashing Blobbis spins - 5
11230-11350	46	Flashing Blobbis spins - 6
11360-11490	47	Player walks right - 1



The Blobbis

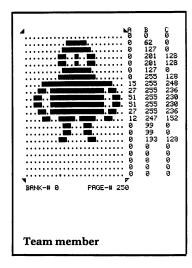


11500-11630	48	Player walks right - 2
11640-11760	49	Player walks left - 1
11770-11900	50	Player walks left - 2
11910-12030	51	Pulsar pulses - 1
12040-12170	52	Pulsar pulses - 2
12180-12310	53	Pulsar pulses - 3

The Blobbis

Sound the Whistle

Igor Tulchinsky



Sound the Whistle is a two-player game similar to football or soccer, but without a ball. The game requires strategy and quick response to the opposing player's actions. Each player controls a four-member team. The object is to score points by running one of your team members through the opponent's goal line and off the edge of the screen, while preventing a member of the opposing team from running past your own goal line. If you run off your own side of the screen, you earn a point for the other team. The first team to score ten points wins the game.

The currently selected team member, or runner, is displayed in a different color than the rest of the team, and is moved with the joystick in the desired direction of movement. The runner can be advanced toward the goal line or used to block the other team's runner. To change runners, hold down the fire button and point the joystick either up, down, left, or right. Each direction corresponds to a particular team member. If a runner bumps into another team member, the runner will be blocked and forced back.

Load the game by typing in LOAD"SOUND WHISTLE",8 and pressing the RETURN key. After about 20 seconds, the computer will say READY, indi-

Sound the Whistle

cating that the game is loaded. Then type RUN and press the **RETURN** key.

Programmer's Notes

Considerations in Creating Sounds. Because sound is one of the basic means of communicating with a computer user, it is a very important element of most games. Clever use of sound effects can add interest to any game and improve even the best ones.

The sound chip in the Commodore-64 computer has three separate voices which may be used alone or together to achieve an amazing range of sound effects. However, synthesizing a particular sound can be very difficult.

Imagine the sound you want to create, and think about its "color" or texture. Is it random, or does it follow a pattern? Is it buzzy or smooth? Is it shrill and piercing, or is it soft? Gunshots, explosions, and wind are noise-like sounds and may be reproduced quite well by the noise waveform. Piercing sounds are best produced by the sawtooth waveform. The triangle waveform is effective for soft sounds, while the pulse waveform is best for brassy sounds.

The C-64 provides a tremendous assortment of interrelated methods to control sounds, and experimentation is often required when the desired sound doesn't fit any easily defined category. Refer to the Commodore 64 User's Guide for an introduction to the many possibilities for sound and music creation. Varied and novel examples of sound creation can be found in all the games in this book. Other books dedicated to C-64 sound and music programming are available in bookstores.

Upper/Lower Case Mode. Lines 20220 through 21220 of the **Sound the Whistle** program listing contain some symbols that appear, at first glance, to be errors. Actually they are the upper case characters of text that was entered in upper/lower case mode. When this data is printed to the screen, it will be converted correctly to upper and lower case.

Sound the Whistle

Hints for Modifying the Game

The statement FL%(I) = 3 in line 675 sets to 3 the distance a player bounces back after hitting another player. Change the 3 to a larger number for greater bounce.

You might want to modify the way the game is played, by making the following changes.

- 1) Make a new rule that all team members on the team must cross the goal line before a point may be scored.
- 2) Remove from the game any team member who has been blocked more than three times.
- 3) Make the selected team member's figure invisible by changing it to the same color as the field.
- 4) Make each team member a different color, to simplify selecting them.

Major Routines

00196-00209

INITIALIZATION

Initialize sprites and sound. Blow whistle and display the title screen. Wait for a key to be pressed while flashing title in different colors. Display instructions and again wait for a key to be pressed. Reinitialize sprites.

00215

MAIN LOOP

Alternate between each game player.

00500-00680

MOVE TEAM MEMBERS

Read joystick for one player. If fire button is pressed, select appropriate team member. If block flag for that player is not set, read joystick for team member direction. If there is movement, make sound. Determine which sprite to use, and move it. If border has been hit, bounce back. If score has been made, print score and check for end of game. Check for sprite collisions and set block flags.

Sound the Whistle

10000-10006 SPRITE DATA 12000-12050 PRINT BACKGROUND Print the play field. 13000-13150 INITIALIZE SPRITES Initialize positions and colors. Blow whistle 15000-15070 **END GAME** Turn everything off and make game ending sound. Wait until either a fire button is pressed or 45 seconds have elapsed, then restart game. 16000-16010 **UPDATE SCORE** Print each player's current score. 17010-17030 SELECT TEAM MEMBER Choose team member and change its color. 18000-18020 **INITIALIZE SOUND** SCORE SOUND 19000-19010 20000-20040 PRINT TITLE Print title and wait for key to be pressed. 21000-21250 PRINT INSTRUCTION SCREEN

Print instructions and wait for key to be

Major Variables

Powers of 2

pressed.

General:

Sound
the Whistle

V
Base address of VIC chip
S
Base address of SID chip
JS%
Joystick status
S%()
Score
Q1%
Sprite collision flag
CQ%
Block counter

P%()

Team members:

CL%() Colors

Team member numbers N%()

Block flags FL%()

DX%() Change in horizontal positions Change in vertical positions DY%()

Horizontal positions X%()

Vertical positions Y%()

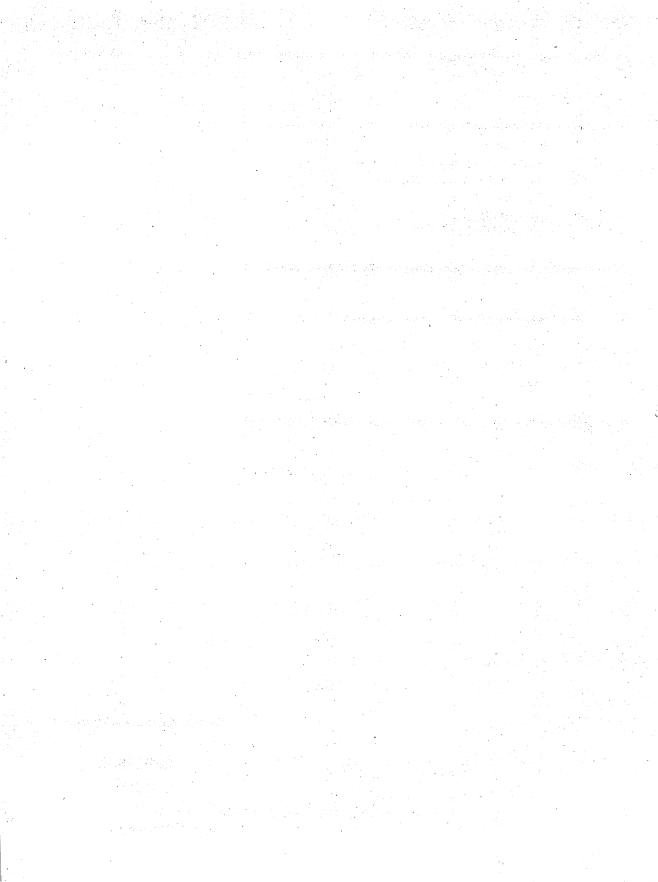
Sprite being moved Z%()

Sprites

Line Numbers Description Page 250 10000-10006

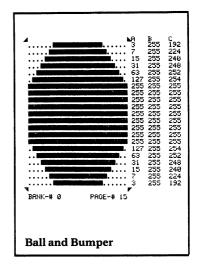
Team Member

Sound the Whistle



Bumper Ball

Charles Mott, Jr.



Bumper Ball is a "pong" or handball-type game pitting one player against the computer. You score one point each time you hit the white ball, controlling the black paddle with the joystick. The computer gets one point each time you miss and let the ball go off the edge of the screen. To have the next ball served, press the fire button. The red bumper ball sometimes adds confusion by deflecting the white ball.

Hitting the ball into the white plunger on the right side of the screen will earn you a bonus of 30 points. Hitting the plunger, however, will freeze your paddle until the ball bounces back from the plunger and hits either the paddle or the red bumper ball, or is lost off the screen. If neither your paddle nor the red bumper ball is lined up to return the ball again, the computer will also get a 30-point bonus! Therefore it is good strategy always to return the paddle to the center of the screen after hitting the ball.

The white ball will start out moving slowly, but its speed will increase if your paddle is moving when it hits the ball. The speed of both the white ball and the red bumper ball are increased when bonus points are scored.

The game is over when you miss the fifth ball. The program will then subtract the computer's score from

yours to give you a final score. If you decide to quit while you still have balls left to play, press **F7** to get your total score. Then you can press **F7** to start a new game.

Load the game by typing LOAD"BUMPER BALL",8 and pressing the RETURN key. After about 15 seconds, the computer will say READY, indicating that the game is loaded. Then type RUN and press the RETURN key. When the game screen displays the flashing message YOUR SCORE, press the fire button to have the first ball served.

Programmer's Notes

Programming for Speed. The MAIN CONTROL LOOP routine in this program is located at the beginning, where it will run most quickly. The initial setup routine, however, must be executed first. Therefore the GOTO170 statement in line 10 jumps over the main loop to execute the setup routines, and then the GOTO20 statement in line 1370 returns to execute the main loop. For further discussion on programming for speed, see the Programmer's Notes for the game Yorick's Revenge.

Use of Variables to Position Cursor. In programming screen displays, it is often necessary to position the cursor at certain locations on the screen in order to begin displaying at those locations with PRINT statements. This usually involves printing a series of CURSOR-DOWN and CURSOR-RIGHT commands, which appear in the program listing as special symbols, before printing what is to appear on the screen.

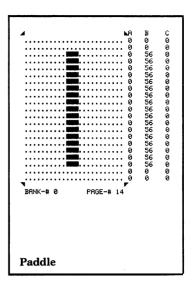
Bumper Ball accomplishes this without cluttering up the program with cursor symbols. In line 360, the variable PY\$ is set to contain one HOME-CURSOR character code and twenty-five CURSOR-DOWN's. In line 370, the variable PX\$ is loaded with forty CURSOR-RIGHT's. (The character codes for all keyboard commands are listed in appendix F of the Commodore 64 User's Guide.)

Calculations made elsewhere in the program then position the cursor within the C-64's twenty-five rows and forty columns. For example, when the ball hits the paddle, the player's updated score is displayed by the

PRINT statement in line 680. LEFT\$(PY,6) and LEFT\$(PX,32) cause the leftmost 6 characters of PY\$ and the leftmost thirty-two characters of PX\$ to be printed. The effect, therefore, is to position the cursor on row 6 at column 33. By simply looking at the PRINT instruction, you can easily tell where on the screen the displayed data will appear.

Complications in Collision Detection. When the game logic requires that collisions between two moving objects be detected, the size and speed of the objects always involve trade-offs. If the distance one object travels in each move is greater than the size of the other object, it is possible for them to appear to pass directly over one another without a collision being detected.

In **Bumper Ball**, for example, the ball's maximum move of 30 pixels may allow the ball to pass over the paddle occasionally. On the other hand, it does permit the game to build up to a fast pace. A lower maximum would permit more reliable collision detection but would make the game less exciting to play.



Hints for Modifying the Game

In implementing any of the following suggestions, bear in mind the preceding discussion regarding collision detection, and carefully observe the effects on the reliability of the collision detection as you experiment with different values for the variables.

The statements **POKEV+23,0:POKEV+29,0** in line 300 set the sizes of the paddle and ball sprites. For a larger red bumper ball, change both 0's to 4's. For a larger paddle, change the first 0 to 2. And to make both the bumper ball and the paddle larger, change the first 0 to 6 and the second 0 to 4.

As the white ball is accelerated, it is prevented from moving too quickly by the check **ABS(CM)** < **30** in line 710. The ball is allowed to travel a maximum of 30 pixels per move. Experiment with different limits. A similar check for the speed of the red bumper ball is made in line 1170.

When the joystick is activated, the paddle travels up

or down the screen 20 pixels per move. For a paddle that is more or less responsive, adjust the +20 in line 30 and the -20 in line 40.

The white ball's initial horizontal and vertical speed are controlled by the variables **BM** and **CM**, respectively, which are set in line 220. The ratio of these two numbers determines the angle of bounce. Experiment with other values for these two variables.

The numbers after the **IFCX**>... portion of line 130 are the horizontal and vertical pixel coordinates of the range over which the red bumper ball can move. To alter the bumper ball's boundaries, change those numbers.

Since the main loop has to be as efficient as possible to maintain adequate speed, we suggest that any changes be limited to variable values, sprite sizes, and features outside the main loop.

Major Routines

0020-0150 MAIN CONTROL LOOP

Read joystick and move paddle. Check for ball hitting bonus plunger or being missed by paddle. Move ball and bumper ball. Bounce ball if it hits wall, bumper ball, or paddle.

0170-0200 PRINT TITLE SCREEN

0220-0370 INITIALIZATION

Initialize variables and load sprites into memory while flashing title screen and making sounds.

0390-0660 PRINT GAME SCREEN

Initialize game screen variables and print screen.

0680-0720 BALL HIT PADDLE

Update and print score. If paddle was moving, speed up white ball.

0740-0790 BALL MISSED PADDLE

Update and print computer score. Make sound and flash COMPUTER SCORE block.

0810 RANDOM NOTE

Sound a note of random frequency.

0840-0920 PREPARE FOR NEXT BALL

Check for no balls left to play. Sound random notes and flash YOUR SCORE block while waiting for fire button or F7 to be pressed.

0940-1020 SPRITE DATA

1040-1150 BALL HIT PLUNGER

Speed up ball and bumper ball. Capture ball with plunger. Sound random notes and flash BUMPER BALL block. Add and print bonus points one at a time. Release next ball.

1170-1180 SPEED UP BUMPER BALL

Check maximum speed before incrementing.

1200-1270 PRINT FINAL SCORE

Sound random notes as the difference between your score and computer score is calculated and printed. Run game again if F7 is pressed.

1290-1380 RELEASE NEXT BALL

Pick random spot for bumper ball in front of plunger. Move plunger in and out to deliver next ball into play. Move ball across screen until it hits bumper ball. Continue normal play.

Major Variables

General:

JS Address of joystick port

JV Value at joystick port

V Base address of VIC chip

SO Base address of SID chip

CL Base address of color memory SC Base address of screen memory

BR Address of border color

BK Address of background color

MY Y position of paddle

MS Computer score

GG Your score

PO Points to add to score

PY\$ String of down cursors

PX\$ String of right cursors

BL\$() Balls left to play strings

XX Pointer to BL\$()

Ball:

BX X position

BY Y position

BM Pixels moved in X direction CM Pixels moved in Y direction

Bumper ball:

CY Y position

CX X position

MC Pixels moved

Sprites

Line Numbers	Page	Description
0940-0970	14	Paddle
0990-1020	15	Ball and Bumper

Ride the Wind

Frank Mikulastik

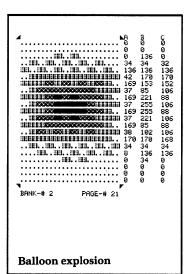
Balloon with heater off

a s a hot air balloon test pilot, your assignment is to test an experimental balloon by safely landing it on each of the eight landing pads at the balloon company's testing grounds.

The balloon's left and right movement is determined by tricky wind currents, which you read from the motion of the clouds. Your only control is the balloon's heater, activated by pressing the space bar. This causes an upward thrust, counteracting the downward pull of gravity and causing the balloon to rise. Using the heater to move up and down through the wind currents, you can control the sideways motion.

Points are scored by landing gently on each pad. Touching down at too great a descent speed will result in a crash. You must complete a round of testing by landing on each of the eight pads once, before you can receive more points by landing on any of them a second time. A color bar at the top left of the screen indicates the pads on which you have successfully landed. The game ends when you have crashed three times.

You will probably have to play this game several times before you begin to make successful landings. Becoming a skillful balloon test pilot requires persistence!



Load the game by typing LOAD"RIDE THE WIND",8 and pressing the RETURN key. After about 35 seconds, the computer will say READY, indicating that the game is loaded. Then type RUN and press the RETURN key. When clouds appear on the title screen, press the space bar to go to the instruction screen, and press it again to start the game. When the game is over, pressing the space bar will return you to the instruction screen.

To stop the game, see the special instructions in the introduction for stopping bank-switched games. This game uses bank 2 for sprites and graphics.

Programmer's Notes

Calculating Effect of Outside Forces on Moving Objects. The motion of the balloon sprite in Ride the Wind is controlled by calculating the effect of three different forces — gravity; wind direction, which is different at various heights above the ground; and the upward thrust imparted by a blast from the balloon's heater.

On each pass through the control loop (lines 500-520), 1.5 units of gravitational attraction are added to the balloon's downward velocity. When the space bar is pressed, 3 units of lift are subtracted from the downward velocity. Thus use of the heater overcomes gravity and causes the balloon to rise gradually.

The wind is calculated with the help of a sine wave and the Y position (height) of the balloon. This gives the effect of wind blowing to the left at one level, no wind at the next level, and then wind blowing to the right at the next.

The speed of the balloon's horizontal motion is changed by the wind as it passes through a particular level. If the balloon has no horizontal velocity or is traveling in the direction of the wind, it picks up speed until its speed matches that of the wind. If its horizontal motion is against the wind, the balloon begins to slow down.

Hints for Modifying the Game

You can make **Ride the Wind** more or less difficult by changing the gravity, wind, heater lift values, and number of balloons. In the statement Y1 = Y1 + 1.5 in line 1010, change the 1.5 to increase or decrease the force of gravity. In the statement Y1 = Y1 - 3 in line 1020, change the 3 to increase or decrease the heater lift amount. To change the wind velocities, represented by variable **Q** in line 1050, change the 2.5 to multiply the sine function by a different value. For example, if you multiplied by 5.0, the winds would be twice as strong at each level. In line 780, set **M** to a value other than 3 to allow more or fewer crashes before the game ends.

Addition of a penalty for colliding with a cloud would be another way to add interest to this game. You might also want to use a different shape of sprite for the clouds, perhaps one which would be easier or harder to avoid with the balloon.

Major Routines

0009-0430 INITIALIZATION

Set VIC memory to bank 2, initialize variables, and load sprite data into memory. Print title, instruction, and game screens. Set up balloon and cloud sprites, and heater sound.

0500-0520 CONTROL LOOP

Move balloon and clouds. Check for good landing or balloon crash.

0700-0790 SET UP BALLOON AND CLOUDS

Initialize sprite pointers.

0800-0890 SET UP HEATER SOUNDS

0900-0990 PRINT GAME SCREEN

1000-1190 MOVE BALLOON

Add gravity pull. If space bar pressed, select balloon sprite with heater, add heater lift

force, and set heater sound volume. If space bar not pressed, select regular balloon sprite. Calculate and add wind force to balloon's motion.

MOVE CLOUDS 2000-2090

Calculate new position for each cloud. If a cloud moves off edge of screen, reposition it to other side.

3000-3090 COLLISION DETECTION

If balloon hit edge of screen or platform supports, make explosion and sound. If more balloons to play, begin again. Otherwise, end game.

3100-3190 LANDING DETECTION

Check to see if position of balloon is same as a landing pad.

3200-3290 LANDED

If pad has not been landed on before, do cloud spin and sound.

4000-4190 PRINT TITLE SCREEN

4200-4390 PRINT INSTRUCTION SCREEN

5000-5290 SOUND ROUTINES

Three sound routines.

5500-5590 GOOD LANDING

Spin clouds and make a sound.

6300-8000 SPRITE DATA

Ride the Wind

Major Variables

General:

V Base address of VIC chip

S1 Base address of SID chip

Wind direction at Y level of a cloud Q1

SC Score

N() Frequency of notes

Balloon:

Horizontal velocity X1 **Y**1 Vertical velocity X X position

Y position Y

SP Sprite page number to use (17 or 18)

Sound volume of heater SV

Balloons left to play M

Wind direction at balloon's position Q

Landing Pads:

L Pad just landed on

All pads balloon has landed on LD

Position K1

K2 Color

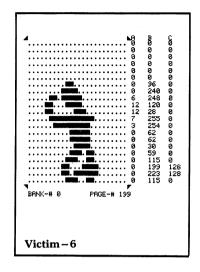
Sprites

(**NOTE:** All sprites are in bank 2)

Line Numbers	Page	Description
6300-6430	16	Cloud 1
6580-6710	17	Balloon with heater off
6720-6850	18	Balloon with heater on
6900-7030	19	Balloon explosion - 1
7040-7170	20	Balloon explosion - 2
7180-7300	21	Balloon explosion - 3
7310-7440	22	Balloon explosion - 4
7450-7570	23	Balloon explosion - 5
7600-7730	24	Cloud 2
7740-7870	25	Cloud 3
7880-8000	26	Cloud 4

Death Valley Patrol

Igor Tulchinsky

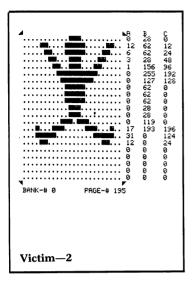


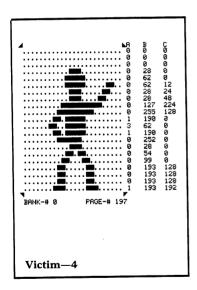
atastrophe in the barren Death Valley! Aircraft crews dropping paratroopers on maneuvers have been given faulty maps, and many are being dropped in the wrong areas and stranded. As the commander of the Death Valley Patrol, you must rescue as many of these victims as you can. However, raging winds in the ravines and side canyons are waiting to smash your rescue helicopter against the canyon walls.

There's no time to waste. All the stranded victims are slowly wasting away — even the ones in canyons you haven't reached yet. Their waving will become less frantic, and eventually they will fall to the ground and be attacked by vultures. When their predicament becomes so bad that future efforts would be in vain, the game will end.

Controlling the helicopter with a joystick, you can rescue a victim by landing very close or by touching the victim. Landings must be made at very low speed, or you will crash. Controlling the helicopter is very tricky. Use a light touch when you move the joystick in the direction you want to travel. After you have successfully rescued the three victims in a particular canyon, fly off the edge of the screen, either above ground or in a tunnel, and you will enter a new canyon with three more victims to save.

Death Valley Patrol





Death Valley Patrol

You are allowed to crash and try again four times in each canyon. The game will end on the fifth crash or if you run out of fuel. If your fuel gauge shows that you are getting low, you may refuel by returning to base and gently landing. However, you can eventually lose by using up all the fuel available.

Landing beside the victim is the preferred method of rescue, and is rewarded with the most points. In each canyon, the first victim rescued by landing is worth ten points; the second, a hundred; and the third, a thousand. When the rescue is made by touching the victim, you receive only five points. If any victim in the valley has been rescued by landing, a bonus is awarded, based on the time required to perform all the rescues in that valley.

Load the game by typing in LOAD"DEATH VALLEY", 8 and pressing the RETURN key. After about 35 seconds, the computer will say READY, indicating that the game is loaded. Then type RUN and press the RETURN key. The title screen will be displayed for about 20 seconds before the game begins. When the game is over, press the fire button to start a new game.

Programmer's Notes

Controlled Random Screens. Death Valley Patrol uses a number of different play screens as the game advances. There are three different ways that these could have been programmed: completely predefined screens, completely random screens, or controlled random screens.

Completely predefined screens can create an interesting game, but they have some disadvantages. The screen definitions use a large amount of memory and generally require extensive programming time. Once completed, the game lacks variety because the player will always advance through exactly the same screens in the same sequence each time the game is played.

Random screens are generated at the time the game is played. Their basic design is controlled by general rules or parameters, but the details are determined by a random number generator. Completely random screens are relatively quick to program and present an infinite variety of challenges for the game player. However, the scores on such games do not accurately represent the player's ability, since the ease of playing each screen is unpredictable, and repeated playing will not necessarily result in greater skill.

The method used in **Death Valley Patrol** retains the programming ease and variety of challenges achieved by completely random screens, but adds an element of predictability. The result is an adventurous game which a player can learn to play with increasing skill.

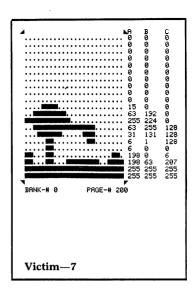
Controlled random screen generation makes use of the "seed" which drives the C-64's BASIC random number generator. The same set of random numbers is generated each time a particular seed value is used. In **Death Valley Patrol**, the CREATE BACKGROUND routine executes the random number generator function, with the contents of the variable **RP** as the seed. The routine then uses the resulting random numbers to calculate the contours of the landscape on the screen and the points at which the victims fall.

The contents of **RP** are determined by the course of the game. Each time the player successfully rescues the victims on a screen and flys the helicopter off the edge, the value of **RP** is incremented by an amount that depends on where the helicopter exits: 0.5 for the left edge, 1.0 for the right edge, or 3.0 for the top of the screen.

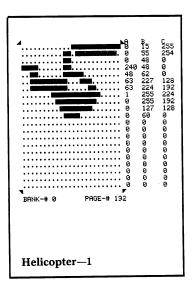
The sequence of the game screens follows a treestructure. As long as a player follows a particular sequence of screens and exit points, the game will be identical each time it is played. But if the helicopter is flown off any screen in a direction different from that taken previously, a different screen will appear and the sequence for the rest of the game will be changed.

Hints for Modifying the Game

Several key variables affect the difficulty of playing **Death Valley Patrol**. You can alter the number of helicopter crashes allowed by changing the 5 to some other value in line 33140. The statement **DF**=**DF**+.1 in line 31005 strengthens the force of the wind. Change the .1 to a different positive value to alter the degree of strengthening.



Death Valley Patrol

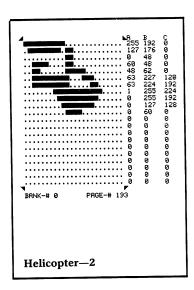


The speed required for a safe base landing can be altered by changing the two occurrences of >2 in line 33006 to other values. Similarly, the speed for safe landings in the canyon can be changed in line 33301.

The length of time required for the victims to completely waste away can be adjusted in the statement C1 = C1 + 1 in line 210. Changing the 1 to a larger value will speed up the process, making the game more challenging.

Try adding a requirement that each victim must be brought back to the base before the next rescue attempt. When a rescue is achieved, set a flag which makes it impossible to pick up other victims. Then clear the flag in the base landing routine.

Experiment with creating a totally new set of screens, by adjusting the values that are added to **RP** in lines 220, 230, and 240.



Major Routines

00009-00010 INITIALIZATION

Print title screen. Initialize sprites and sounds. Create first background screen.

00020-00260 MAIN LOOP

Check fuel supply and test for collisions. Check for expired victims. Adjust helicopter velocity according to joystick direction. Adjust position of helicopter according to velocity, and change pitch of helicopter sound. Check to see if helicopter is outside of screen bounds. If it is and all three victims are gone, advance to new screen. If all victims are not gone, make sure helicopter stays within screen.

Death Valley Patrol

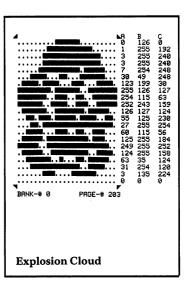
01000-01030

ADJUST FUEL GAUGE

If conditions are right, increase strength of wind in a random wind sector.

10000-10156 SPRITE DATA

30000-30100 SPRITE INITIALIZATION Do checksum to see if sprites already loaded. If not, load sprites. Define sprite size, color, etc. Set all needed constants. 31000-31650 CREATE BACKGROUND Create mountains, cave, and split screen. Drop victim. Set landing detectors. Draw score and fuel gauge. If subroutine is accessed from REFUEL, return. Create wind matrix. 31700-31710 CLEAR A HOLE 31900 CREATE A PSEUDORANDOM NUMBER 32000-32010 SOUND INITIALIZATION 33000-33150 SPRITE-BACKGROUND COLLISION Check for base landing and landing speed. If too high, explode. If low enough, refuel. Check for cliff collision.



33300-33400 VICTIM RESCUE

If landing speed is too high, explode. Determine type of rescue and increment

If collision detected, explode.

score.

34000-34040 SPRITE-SPRITE COLLISION

Check to see which victim was rescued.

Branch accordingly.

35000-35020 PRINT SCORE

Change score to a five-character string

value and print it.

36000-36090 PRINT TITLE SCREEN

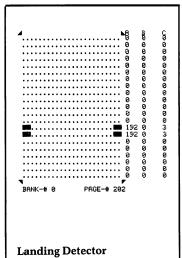
Print title and make sprites jump and

wave.

40000-40030 ENDING

Make sound and end game.

Death Valley Patrol



Major Variables

General:

PK Joystick status

V Base address of VIC chip

C1 Game time counter

RP Random number seed

SC Score

C2 Random wind counter

DF Difficulty

AG Fuel left at base

E Current mountain height

D Last mountain height

A Mountain position

R Random number

F Refueling flag

LV Level

TP Highest canyon number of last stage

DX() Horizontal wind

DY() Vertical wind

Helicopter:

S() Sprite pointers

DX Change in horizontal movement

DY Change in vertical movement

X Horizontal position

Y Vertical position

NH Number of crashes

GS Fuel left

Victims:

T() Sprite pointers

NK Number rescued by touching

NM Number rescued by landing

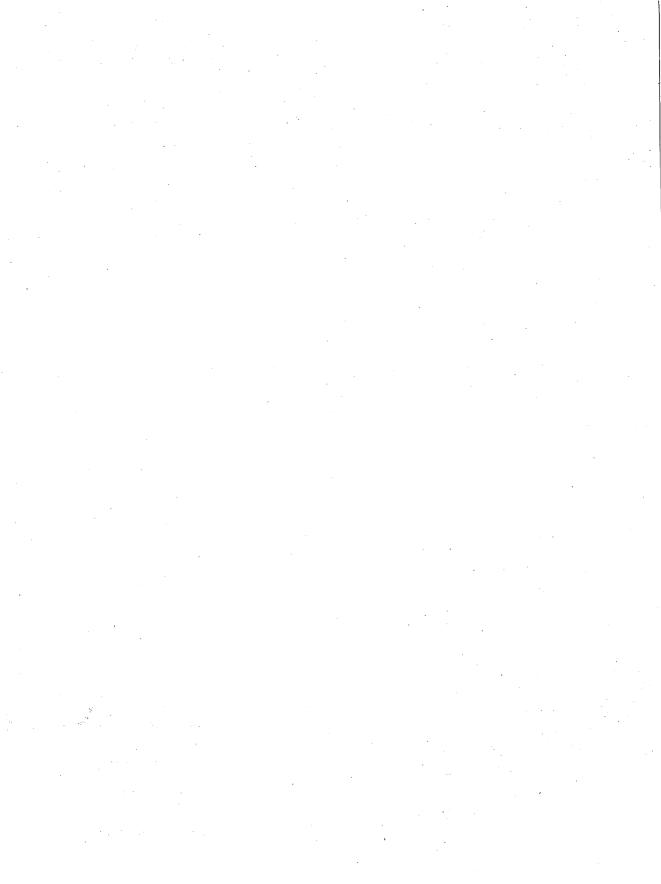
Death Valley Patrol

Sprites

Line Numbers	Page	Description
10000-10006	192	Helicopter - 1
10010-10016	193	Helicopter - 2
10020-10026	194	Victim - 1

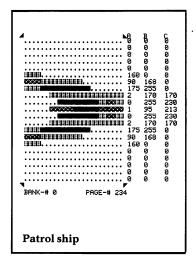
10030-10036	195	Victim - 2
10040-10046	196	Victim - 3
10050-10056	197	Victim - 4
10060-10066	198	Victim - 5
10070-10076	199	Victim - 6
10080-10086	200	Victim - 7
10090-10096	201	Victim - 8
10100-10106	202	Landing detector
10150-10156	203	Explosion cloud

Death Valley Patrol



Mission: Tobor

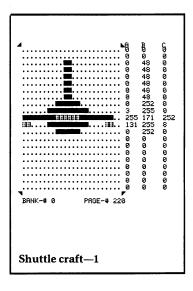
Charles Mott, Jr.



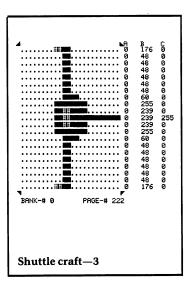
A utomated atomic energy reactors provide power for operations on the planet Tobor. The reactors are shielded by a protective force field and guarded by an automated patrol ship. This protection system has developed a serious malfunction and is preventing delivery of essential coolant packs. As a result, the reactors are now in danger of overheating.

You have been dispatched to drop new coolant packs from your space shuttle to the robots below. These robots continuously move back and forth between each reactor and its supply area. If a coolant pack falls just in front of a robot, the robot will catch the pack and take it to the reactor. Keep an eye on the reactor's coolant level gauge. To save the reactor, you must successfully deliver enough coolant packs to fill it up.

The protective force field constantly pushes your shuttle up and away from the reactor. If you penetrate the shield, the defective protection system will strengthen the field and repel you, using more coolant in the process. If you drop a coolant pack from just above the shield, the protection system will dispatch a patrol ship to investigate. A collision with the patrol ship will not harm your shuttle, but it will cause the reactor to use still more coolant.



Mission: Tobor



When you have returned a reactor to normal operation, you will be moved on to the next one. The robot at each new reactor will move faster than the previous one, frantically trying to find coolant packs. If you allow any reactor's coolant level to drop below the critical point, the reactor will melt down and the game will end, displaying your rating as a shuttle pilot.

Use the joystick like the control stick in an airplane to move the shuttle. Push it forward to dive, pull it back to climb, and move it left or right to move sideways. To release a coolant pack, press the fire button.

Load the game by typing LOAD"MISSION TOBOR",8 and pressing the RETURN key. After about 30 seconds, the computer will say READY, indicating that the game is loaded. Then type RUN and press the RETURN key. The title screen will be built on the screen, complete with instructions. Then after a delay of about 20 seconds, the game will begin.

Programmer's Notes

Special Use of the Keyboard Buffer. One way of handling a program which uses many sprites and requires a great deal of memory is to write it in two parts. Part 1 of this game is called MISSION TOBOR on the disk. It prints the title screen, scrolls the instructions, and loads the sprite data into memory in preparation for Part 2, the actual game, which is called && MISSION. Part 2 is then run by making special use of the keyboard buffer.

The Commodore 64 has a special section of memory called the keyboard buffer, beginning at memory location 631, for temporary storage of keyboard input. It can hold up to ten characters and is automatically cleared as its contents are executed. Any time a key is pressed, a special code for that key is stored in the keyboard buffer until it can be processed.

For the C-64 to load and run Part 2, it must be given instructions similar to those that were entered to load and run Part 1. To do this, Part 1 prints the instruction LOAD"&& MISSION",8 onto the screen, in the same color as the background so that it will be invisible. Then, in lines 3610-3650, it stores the following directly into the keyboard buffer: two CURSOR-UP commands, a

Mission: Tobor

RETURN command, the characters *R*, *U*, *N*, and a colon, and then another RETURN command. (The character codes for all keyboard commands are listed in appendix F of the *Commodore 64 User's Guide*.)

When Part 1 ends, the C-64 executes the commands that are in the keyboard buffer. The two CURSOR-UP commands position the invisible cursor to the invisible LOAD line. The RETURN command causes the LOAD"&& MISSION",8 to be executed, loading Part 2. The next positions in the keyboard buffer contain R U N: and a RETURN. This is equivalent to typing in RUN and pressing RETURN, so Part 2 then begins running.

Hints for Modifying the Game

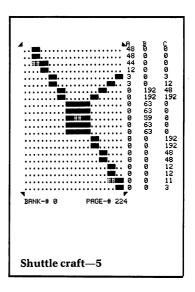
Part 1. To present the title screen in a different way, modify lines 700 and 730. The RIGHT\$ instructions are string functions that control the point in the string variables X\$ and Z\$ at which the print statement is to begin, and how many characters after that point will be printed. Replacing the RIGHT\$ instructions with LEFT\$ or MID\$ will create different effects.

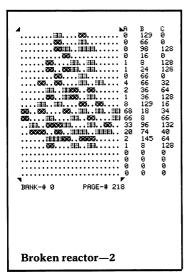
The PRINT TITLE SCREEN routine is designed to gradually build the game name on the screen as the sprites are loaded into memory. The name is camouflaged until it is completely built. To have it presented more quickly, eliminate the **GOSUB2000** in line 720.

If you study the program carefully and learn to use the string commands, you can find other, more complicated ways of changing the title screen presentation.

Part 2. To change the initial speed of the robot, modify the value assigned to PH in line 103. The speed of the robot at each new reactor is increased in lines 3205-3209. Change the values assigned to PH to make the robot speed up by a different amount, or slow down instead. To change the speed of the shuttle, assign a different value to AP in line 103. To change the speed of the patrol ship, assign a different value to U in the same line.

The strength of the force field's push against the shuttle is set with the statements MX = MX + 1: MY = MY - 1 in line 390. Change the +1 and -1 to other





values for the force field to have a different effect on the shuttle's behavior.

The time before a reactor's coolant runs out is controlled by the variable **PI**, which is set to 1 in line 90. Changing the 1 to a 2 will cause the reactors to melt down in half the time. The initial state of the next reactor, determined by its sprite pointer, is set with the statement **PS=202** in line 948. Changing the 202 to a higher number, not greater than 216, will make the next reactor start out in a more critical state.

Set a limit to the number of coolant packs available to be dropped during the game. Or set a limit for each reactor, and decrease it for each reactor serviced. You will have to add variables for the limit that is set and the number of packs dropped.

Major Routines

PART 1

0070-0320 INITIALIZATION

Begin loading sprite data into memory. Initialize array variables for title and instructions.

0510-0730 PRINT TITLE SCREEN

Manipulate and print title array variables with sound and extended background color, while loading sprite data.

0740-0840 SCROLL INSTRUCTIONS MESSAGE
Scroll instruction message in Y\$() acr

Scroll instruction message in Y\$() across middle of screen while loading sprite data.

2000-2100 READ SPRITE DATA INTO MEMORY Read one byte of sprite data into memory.

3500-3680 RUN PART 2

Lower top of BASIC memory and use keyboard buffer to run Part 2.

Mission: Tobor

5000-5131 SPRITE DATA

Data statements for sprites. Every four lines is a complete sprite.

PART 2

0040-0140 INITIALIZATION

Read data statements for score ratings and joystick control array. Initialize variables and sprites. Print background screen.

0147-0480 MAIN CONTROL LOOP

If joystick activated, move shuttle; otherwise let it drift. Move robot and pack, if one was dropped. Check for robot catching pack. If robot is at reactor with pack, switch to reactor sprite with more coolant. If reactor full, print score and start new mission. If reactor empty, end game.

0500 READ JOYSTICK

0600-0620 MOVE SHUTTLE

0800-0895 INITIALIZE SPRITES

Sprite positioning and definitions.

0900-0940 END OF GAME

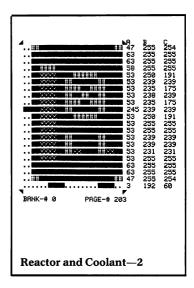
Flash border and background while making melt-down noises. Print overall rating. Wait for fire button to start new game.

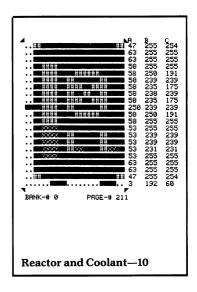
0955-1000 CATCH COOLANT PACK

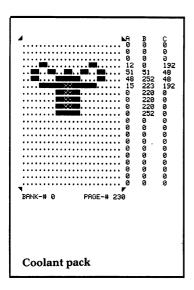
If pack is within 10 pixels of robot, switch robot sprite to one carrying a pack, and set flags.

1100-1155 PRINT BACKGROUND SCREEN

2000-2310 SOUND ROUTINES
Three sound routines.







3200-3250 PRINT SCORE

Print rating points and increase speed of robot.

Major Variables

PART 1

General:

Base address of SID chip Title word "MISSION" X\$() Title word "TOBOR" Z\$()

Y\$() Instructions

A\$ Temporary variable for instruction scroll

PART 2

General:

V Base address of VIC chip Base address of SID chip SI SC Base address of screen memory BR Border color Background color BK

IS Address of joystick port DR() Joystick directions

FG() Event flags S1-S8 Sprite pointers

Number of reactors saved YL

LP Pointer to ratings

G\$() Ratings

Shuttle:

P1 Sprite pointer MX X position MY Y position Pixels moved AP

Mission: **Tobor**

Reactor:

PS Sprite pointer

PΙ Sprite pointer increment

Mission: Tobor

Robot:

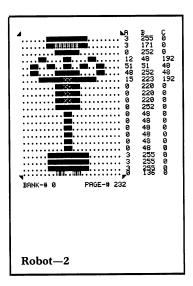
RB Sprite pointer R X position PH Pixels moved

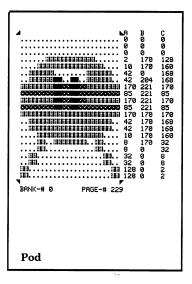
Patrol ship:

BP X position U Pixels moved

Sprites

Line Numbers	Page	Description
5000-5003	202	Reactor and coolant - 1
5004-5007	203	Reactor and coolant - 2
5008-5011	204	Reactor and coolant - 3
5012-5015	205	Reactor and coolant - 4
5016-5019	206	Reactor and coolant - 5
5020-5023	207	Reactor and coolant - 6
5024-5027	208	Reactor and coolant - 7
5028-5031	209	Reactor and coolant - 8
5032-5035	210	Reactor and coolant - 9
5036-5039	211	Reactor and coolant - 10
5040-5043	212	Reactor and coolant - 11
5044-5047	213	Reactor and coolant - 12
5048-5051	214	Reactor and coolant - 13
5052-5055	215	Reactor and coolant - 14
5056-5059	216	Reactor and coolant - 15
5060-5063	217	Broken reactor - 1
5064-5067	218	Broken reactor - 2
5068-5071	219	Broken reactor - 3
5072-5075	220	Shuttle craft - 1
5076-5079	221	Shuttle craft - 2
5080-5083	222	Shuttle craft - 3
5084-5087	223	Shuttle craft - 4
5088-5091	224	Shuttle craft - 5
5092-5095	225	Shuttle craft - 6
5096-5099	226	Shuttle craft - 7
5100-5103	227	Barrel
5104-5107	228	Pipework
5108-5111	229	Pod
5112-5115	230	Coolant pack
5116-5119	231	Robot - 1

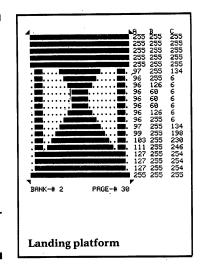




5120-5123	232	Robot - 2
5124-5127	233	Robot - 3
5128-5131	234	Patrol ship

Cosmo's Rescue

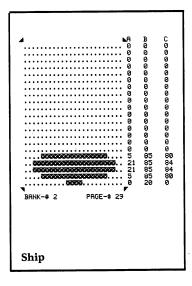
Frank Mikulastik

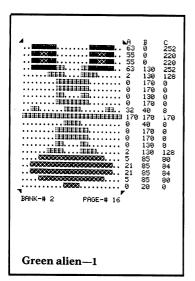


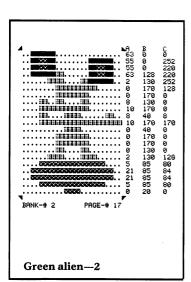
hile exploring the hazardous cavern of the yellow monster Grunk-Snort, you and your purple friend Cosmo have become separated, and he has been captured by the dreaded Grunk-Snort. You must guide your rescue ship through the cavern, pick up Cosmo, and make your way back home. You have only three chances to rescue Cosmo. If you touch a wall, bump into Grunk-Snort, or run out of fuel, you will have to start all over again. Your ship awaits you on the landing platform at the upper left-hand corner of Grunk-Snort's cavern.

Keep an eye on the fuel indicator. The landing platform also serves as a fuel station at which you may refuel if necessary. To make a safe landing on the platform, your ship must be moving very slowly with no side motion. If you make it back to the landing platform after picking up Cosmo, your heroism and skill will be rewarded with one point for each unit of fuel remaining.

Control the ship with the joystick. Gravity will be pulling you down at all times. Pressing the fire button fires the main engines, which thrust upward. Left or right motion of the joystick fires the side thrusters. Apply thrust very cautiously. Controlling the ship can be ticklish, and you may have to play many times before you learn to do it perfectly.







Cosmo's Rescue

Load the game by typing in LOAD"COSMO'S RESCUE",8 and pressing the RETURN key. After about 45 seconds, the computer will say READY, indicating that the game is loaded. Then type RUN and press the RETURN key. After the title screen is displayed, the instruction screen will appear and pause for about 15 seconds. Then you will be asked to press the fire button to start the game. When the game is over, press the fire button to return to the instruction screen.

To stop the game, see the special instructions in the Introduction for stopping bank-switched games. This game uses bank 2 for sprites and graphics.

Programmer's Notes

Bank Switching to Handle Large Programs. Because Cosmo's Rescue is a very large game, and in addition, uses 26 sprites for animation effects, a special technique was required to load the game and all the sprite data into memory at the same time.

The "Sprites" section of the Introduction explains that the C-64's memory is divided into four banks of 16K each. The computer's VIC chip is designed to "see" and access only one 16K bank at a time. Since the VIC chip performs all sprite handling, it must be set to access the bank containing the pages where the sprite shape data is stored.

Bank 0 of the computer, the first 16K of memory, is ordinarily used for most purposes unless other provisions are made. When the computer is first turned on, the VIC chip is set automatically to bank 0, and when programs are loaded, they are stored there. When a program and its sprite data both fit into 16K, no further action is needed. The sprite data will be stored into pages in bank 0, where the VIC chip is set to find it, and processing will begin.

If the program and its sprite data require more than 16K, however, the sprite data must be stored into pages in another bank. The VIC chip must then be "bank-switched" or set to access the other bank for the sprite data. The program remains in bank 0 and is executed there.

In addition to sprites, the VIC chip also controls other graphics features. Bank switching must be implemented carefully, so that all related functions are handled properly.

Cosmo's Rescue stores its sprite data in bank 2. The instructions which bank-switch the VIC chip are in lines 37 through 40. You may also want to study **Ride the Wind**, **Squirrel Away**, and **The Blobbis**, all of which use bank switching. For information on the instructions necessary to use bank switching in other programs, refer to one of the books available on Commodore 64 programming.

Hints for Modifying the Game

The beginning fuel supply is set with the statement FU=600 in line 3030. The refueling amount is set in a similar statement in line 3618. Change the 600 to get a different amount of fuel, and change the **PRINT** statement in line 3110 to match. The number of crashes allowed before the game is over is set with the statements ME=3 and LEFT:3 in line 335. Change the two 3's to other values.

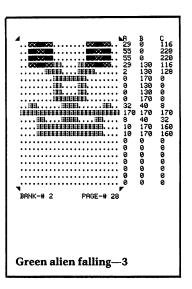
The starting positions for Cosmo, Grunk-Snort, the player figure, and the landing platform can be altered by making changes to lines 3000 through 3030. These changes can be tricky, so be very careful in making them.

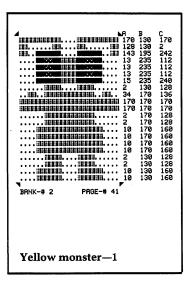
For Cosmo's vertical position, change the statement **POKEV** +13,92 in line 3010 to use a value other than 92. Set G1 to a different value in line 3005 to change the figure's horizontal position. To change the left and right movement boundaries for Cosmo, change the values of **E1** and **F1** in line 3005.

For Grunk-Snort's vertical position, change the statement **POKEV** +15,200 in line 3020 to use a value other than 200. Set **G2** to a different value in line 3005 to change the figure's horizontal position. To change the left and right movement boundaries for Grunk-Snort, change the values of **E2** and **F2** in line 3005.

For the landing platform's horizontal position, change the statement **POKEV** + **10**,**32** in line 3000 to use a value other than 32. Change the value of **T5** in line 3000 to change its vertical position.

The green player figure starts out just above the landing platform, so its starting vertical position is calcu-





lated in line 700 using T5. To move the green figure's starting position, change the values added to Y and assigned to X in line 700.

In programming Cosmo's Rescue, care was taken to make the layout of the cavern easy to change. To design your own cavern, change the PRINT statements in lines 3112 through 3158. Getting the new cavern screen to look just the way you want can be tedious and frustrating, but after it is done, you will have a new game to play.

You might want to create several cavern screens, each in its own subroutine. Then program the game to move on to the next screen after each one has been played successfully. You would then have programmed an example of completely predefined screens (see the Programmer's Notes for the game **Death Valley Patrol**).

Major Routines

0010-0335 INITIALIZATION

Set VIC memory to bank 2. Print instruction screen and load sprite data. Set up sprite arrays for animation and turn on sprites. Print play screen.

0400-0490 MAIN LOOP

Move green alien and check remaining fuel. Fall, if out of fuel. Check for landing on platform, hitting cavern or monster, picking up purple alien and successful rescue. Move purple figure and yellow monster.

0700-0790 INITIALIZE GREEN ALIEN

Set starting position, size, and color of green alien and his ship.

1000-1490 MOVE GREEN ALIEN

Read joystick and add pull of gravity. If fire button pressed, subtract from gravity and adjust fuel. If joystick moved, update horizontal speed and adjust fuel. Calculate new position and select proper thruster sprite. Sequence to next green alien sprite and print remaining fuel.

3000-3090	INITIALIZE SPRITES		
	Set starting positions and colors		

3100-3190 PRINT PLAY SCREEN

3200-3290 GAME OVER

Do animation sequence for purple alien and yellow monster while waiting for fire button.

3300-3390 PRINT INSTRUCTION SCREEN

3500-3590 SCREEN CHANGE Screen change sound and flash.

3600-3690 GREEN ALIEN LANDS ON PLATFORM Fall if landing speed is too high. Check for purple alien being rescued. Refuel.

3700-3790 GREEN ALIEN FALLS

Do falling animation sequence and check for end of game.

4000-4090 MOVE PURPLE ALIEN
Sequence to next sprite in animation and calculate new position. Reverse direction if movement boundary reached.

4100-4190 MOVE YELLOW MONSTER
Sequence to next sprite in animation and calculate new position. Reverse direction if movement boundary reached.

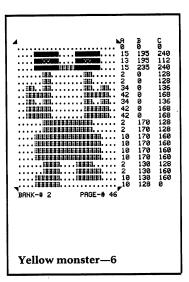
4300-4390 PICK UP PURPLE ALIEN SOUND Sound for green alien rescuing purple alien.

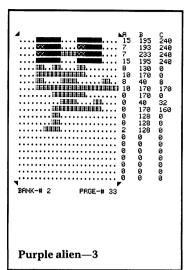
4500-4590 FALLING SOUND

4600-4690 SUCCESSFUL RESCUE

Calculate and print score. Make sound and continue game.

7000-10540 SPRITE DATA





Major Variables

General:

FU Fuel remaining FS Score counter

CT Successful rescue flag
ME Rescue attempts left
V Base address of VIC chip
S1 Base address of SID chip

SM Base address of screen memory
 CM Base address of color memory
 M1 Base address of sprite data
 T5 Landing platform Y coordinate

Green Alien:

MS() Animation array for flying A1() Animation array for falling T1 Length of flying sequence T Length of falling sequence

X X positionY Position

X1 Number of pixels to move in X directionY1 Number of pixels to move in Y direction

X2 New pixel count for X directionY2 New pixel count for Y direction

K1 Page number for bottom thruster sprite
K2 Page number for left thruster sprite

K2 Page number for left thruster spriteK3 Page number for right thruster sprite

SP Animation counter

Purple Alien:

GS() Animation array for walking
T3 Length of animation sequence
E1 Maximum left position
F1 Maximum right position

A3 Animation counter

G1 X position

D1 Direction of movement flag

Cosmo's Rescue

Yellow Monster:

MM() Animation array for walking

T4 Length of walking sequence

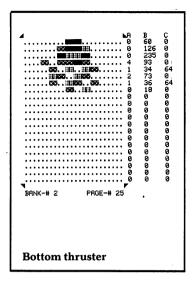
A4 Animation counter

- E2 Maximum left position
- F2 Maximum right position
- G2 X position
- D2 Direction of movement flag

Sprites

(NOTE: All sprites are in bank 2)

(110 12: 1111 opinico di c ini canni 2)				
Line Numbers	Page	Description		
07000-07130	16	Green alien - 1		
07140-07270	17	Green alien - 2		
07280-07400	18	Green alien - 3		
07410-07540	19	Green alien - 4		
07550-07670	20	Green alien - 5		
07680-07810	21	Green alien - 6		
07820-07950	22	Green alien - 7		
07960-08080	23	Left thruster		
08090-08220	24	Right thruster		
08230-08350	25	Bottom thruster		
08360-08490	26	Green alien falling - 1		
08500-08630	27	Green alien falling - 2		
08640-08760	28	Green alien falling - 3		
08770-08900	29	Ship		
08910-09040	30	Landing platform		
09050-09180	31	Purple alien - 1		
09190-09320	32	Purple alien - 2		
09330-09450	33	Purple alien - 3		
09460-09590	34	Purple alien - 4		
09600-09720	35	Purple alien - 5		
09730-09860	41	Yellow monster - 1		
09870-09990	42	Yellow monster - 2		
10010-10130	43	Yellow monster - 3		
10140-10270	44	Yellow monster - 4		
10280-10400	45	Yellow monster - 5		
10410-10540	46	Yellow monster - 6		





Mazemaster

Igor Tulchinsky

azemaster tests your ability to maintain a sense of direction in a complex network of corridors. The game simulates 3-D by providing a perspective view of the corridor down which you are moving. The view shows the floor and ceiling, the right and left walls, and the surface facing you at the end of the corridor. The floor and ceiling are alternately red and white, each color panel marking the distance you will move with one step. Doors in the walls are indicated by black panels. In addition, a top view of the maze is available.

There are two different ways of playing this game. In either option, the object is to finish in the shortest time possible. A final score is awarded, based on your playing time and the complexity of the maze.

In option 1, PAINT THE FLOOR, you must paint the floor yellow by traveling through the entire maze. Whenever you retrace your steps, the floor in front of you will be yellow instead of red and white. (Since you will start at a random point within the maze, it may be necessary to retrace some of your steps in order to travel through the entire maze.

In option 2, FIND THE TREASURE, a treasure is hidden in the maze for you to find. The treasure's location is shown in the top view but invisible to you as you approach it in the corridor. Your distance from the treasure is displayed, however, so you can tell whether you are moving toward or away from it.

When you select an option, you will also determine the complexity of the maze by entering a number between 1 and 200 and pressing **RETURN**. The program will then randomly generate the maze, displaying a map as it develops. After the maze is drawn, different areas will be colored to indicate the wall color in those areas. The wall colors will help you relate the corridor perspective view to the top-view map as you play.

Pressing the space bar moves you forward one step, and the R and L keys turn you right and left. To turn around and go back, turn in the same direction two times. In order for you to turn and go through a door, the door must be at the edge of the screen, indicating that you are standing next to it. After you turn, you will have a perspective view down the corridor you have turned into. If you turn and are not standing next to a door, you will just be looking at a wall.

If you become confused as you are playing and would like another top view of the maze, press \mathbf{H} for help. This time, your position will be marked on the map with a U, D, R, or L (for up, down, right, or left), which indicates the direction you are facing within the maze as it is displayed. If you are playing PAINT THE FLOOR, the areas already painted will be yellow.

Load the game by typing in LOAD "MAZEMASTER",8 and pressing the RETURN key. After about 20 seconds, the computer will say READY, indicating that the game is loaded. Then type RUN and press the RETURN key. Select an option from the game menu and enter the desired complexity. The program will then begin drawing the maze. After you finish a game, press the space bar to return to the menu screen.

Programmer's Notes

Completely Random Screen Generation. The mazes in Mazemaster are created by a completely random method. See the Programmer's Notes for the game **Death Valley Patrol** for a discussion of completely predefined, completely random, and controlled random screen generation.

Simulated Three-Dimensional Display. One of the unusual features of this game is its three-dimensional display of a maze. Each interior view of a corridor shows the nearest five steps, with the floor, ceiling, and walls converging toward the center of the screen. Alternating red and white ceiling and floor panels mark the steps. If a wall in one of the five steps opens onto a side corridor, it is displayed as a black panel. When the player is within five steps of the end of a corridor, the center of the screen shows the color of the wall that is being approached.

One set of array variables is used for all the different interior views of the corridors. The string arrays G\$(), C\$(), R\$() and L\$() represent the colors of the floor, ceiling, left walls and right walls, respectively. Each array has five elements, one for each step in the view. As the end of the corridor is approached and there are fewer than five steps to display, the other array variables are assigned to be the color of the end wall, so that it appears to grow larger and larger with each step.

Other variables are used to indicate the direction the player is facing in the maze and to monitor the features of the corridor that are to be displayed on the next step. As the player moves forward in the maze, the contents of the string array variables are modified to reflect the new view.

Hints for Modifying the Game

The score is calculated and printed in lines 40040-40055, based on time and maze complexity. You might want to add a penalty for looking at the maze map. First, add the line 1005 NA = NA + 1 to count the number of times the maze map is displayed. Then add 40051 SC = SC - NA*10 to include the new counter in the score calculations. The numbers 1005 and 40051 are line numbers which indicate the location in the program where you should place the new lines.

Mazemaster has instructions to end a game when a time limit is reached, but the maximum number of seconds is set with the statement TM = 1E9 in line 22070. The expression 1E9 is BASIC notation for ten raised to the ninth power – a gigantic number! Change this line to set

the time limit to a realistic number based on maze complexity.

You might want to try playing in the dark: turn off the three-dimensional display and just make a "bump" sound when you hit the end of a corridor. Or make the treasure periodically move around within the maze.

Major Routines

00013 INITIALIZATION

Set up sound. Print title screen. Offer and accept game and complexity choice. Create selected maze. Jump to selected

main loop.

00100-00410 3-D PRINTOUT OF MAZE

Print out 3-D maze as calculated in next

section.

00530-00720 3-D PRINTOUT CALCULATIONS

Set up direction variables. Calculate colors for ceiling, floor, left, and right walls for five squares in direction player is fac-

ing.

01000-01060 PRINT HELP MAZE

Display top view of maze. Display player and, if in FIND THE TREASURE option,

display treasure.

02000-02110 PAINT FLOOR MAIN LOOP

Select starting point. Print out maze, squares painted, and time. Wait for key to be pressed, while incrementing time.

React to key pressed.

03000-03110 FIND TREASURE MAIN LOOP

Select starting point. Print out maze, distance, and time. Wait for key to be pressed, while incrementing time. React

to key pressed.

20030-20090 CREATE RANDOM MAZE

Create maze, by picking random direction and testing spot two units in that direction. If spot is empty, link it to maze. If filled, pick new direction. If all directions are filled, pick new starting point.

20100-20180 COLOR THE MAZE

Color all walls, ceilings and floors.

21010-21080 PRINT TITLE

Display heading.

22010-22080 PRINT GAMES MENU

Print menu and enter all inputs.

40005-40080 GAME OVER

Sound end of game sound. Calculate score. Wait for key to be pressed, then

resume game.

50000-50010 SOUND INITIALIZATION

55010 SOUND A BEEP

Major Variables

General:

M%() Maze

C\$() Ceiling color

G\$() Ground color

R\$() Right wall color

L\$() Left wall color

D() Directions

CM Complexity

F3 Space taken flag

G\$ Game number

DX() Horizontal component

DY() Vertical component

D1 Check for adjacent corridors (right)
D2 Check for adjacent corridors (left)

Player:

XX Horizontal distance "seen"
 YY Vertical distance "seen"
 X Horizontal position
 Y Vertical position
 D Direction

Treasure game:

A Horizontal position
B Vertical position

Paint game:

NY Number of squares painted yellow

NB Number of squares

P Position of next yellow square

Appendix A

Game Programs



Rocket Lander

Charles Mott Jr.

```
INITIALIZATION
26:
26:
27 REM PRINT TITLE SCREEN
28:
30 PRINT"INNOMM"TAB(13)"ROCKET LANDER"
35 FORT=1T014:PRINT:NEXT
40 PRINT" (C) 1984 FANFARE HOUSE
NC."
BY CHARLES MOTT
                                 1984 FANFARE HOUSE,
                                      BY CHARLES MOTT,
60 FORT=0T0319:READA:POKE12288+T,A:NEXT
67
68 REM
                   PRINT INSTRUCTION SCREEN
70 PRINT": INDICATE ROCKET-RIGHT.
....PRESS F1"
75 PRINT | INDICATE ROCKET-LEFT....
..PRESS F3" NOCKET-UP...
..PRESS F5"
84 PRINT" MANAGEMENT
85 PRINT"
NG PAD "
86 PRINT"
                     # LAND ROCKET ON BLACK LANDI
88 REM
                      LOAD CHARACTER DATA
89
90 POKE52,120:POKE56,120:SK=0
116 FORT=1T01000:NEXT:GOSUB900:GOSUB800
117 PRINT"TI # GET READY - ROCKETS AP
ROACHING "
                                                  ROCKETS APP
117 PRINT"IT # GET READY - ROCKETS HPP
ROACHING "
120 POKESL, (PEEK(SL)AND254)
130 POKE1, PEEK(1)AND251
140 FORI=0TO255:FORJ=0TO7:POKEBA+I*8+J,P
EEK(V+I*8+J):NEXTJ,I:GOSUB900
150 POKE1, PEEK(1)OR4:POKESL,PEEK(SL)OR1
160 PRINT"IT;:POKEV+24,(PEEK(V+24)AND240
      FORT=14552T014552+143:READA:POKET,A:
-.5 FURT=
NEXT:SK=0
190
                          PRINT BACKGROUND
```

ESO+1,T*9 215 POKESC+30+SK+T,160:POKESC+SO+30+SK+T ,7:NEXTT:SK=SK+40:NEXT:POKESO+4,32 220 PRINT"%"TAB(48)"%ROCKET LANDER" 225 PRINT"%M"TAB(33)"%GOOD":PRINTAB(31) ;.... 252 (25) ---- (25) (26 / 24) (26 / 26) 253 PRINT" `;**3**\$# :3E E + =4 อัลดี คลิโฟิร์" 23, 000# (SEC 96 SEC (SE6 SEC) SEC (SE6 C SE 265 PRINT" . 3,00000+ 270 PRINT" 3,000000+ 280 PRINT" 3,0000000+ (3) *****(3) 里 : . C + 3. C ≽3" 320 GOSUB1000 34ē 341 REM INITIALIZE SPRITES 342) PRINTLEFT\$(V\$,22)LEFT\$(H\$,31)"# ##########PO 370 390 : 391 REM MAIN GAME LOOP 392 AA=PEEK(197):IFAA=64THENPOKERS,192:P 400 400 AA=PEEK(197):IFAA=64THENPOKERS,192:POKESO+4,128
403 POKESO+1,2:IFAA<>6THENPOKESO+1,250
405 AN=AN+TU:IFAN>30RAN<1THENTU=-TU
410 IFAA=4THENPOKERS,194:FOKEV,H1:PO=PO100:PU=1:H1=H1+1:IFH1>244THENH1=244
420 IFAA=5THENPOKERS,193:POKEV,H1:PO=PO100:PU=1:H1=H1-1:IFH1<24THENH1=24
430 IFAA=6THENPOKERS,195:K1=K1-.1:FG=1:PO=PO-500:PU=1
433 IFAA=6THENPOE 433 435 IFPO<0THENPO=0 IFFGTHENFG=0:GOTO440 1FF61HENF6=0.6010440 K1=K1+.1 V1=V1+1+K1:IFV1>210THENV1=210:GF=1:I -.1AND(PEEK(V+30)AND1=1)THENGOSUB500 IFGFTHENPOKESO+4,129:FORT=0T015STEP. 440 FKĨ< 448 POKE646,10 450 POKEV+1,V1:PRINTLEFT\$(V\$,12)LEFT\$(H\$ 450 16)X\$(AN) IFPUTHENPRINTLEFT\$(V\$,22)LEFT\$(H\$,31 INDEXEDDENT PO:POKESO+4,129 IFPO<1THENPOKERS,192:POKESO+4,128:GO 460 3"**3** 465 ŤÕ437 PU=0:GOTO400 490 491 492 GOOD LANDING REM 432 ...
500 POKESO+4,129:FORT=1ST00STEP-.1:POKES
0+24,T:NEXT:POKESO+4,128:POKESO+24,15
505 POKESO+4,33:FORT=1T025:POKE53280,T:
POKESO+1,T:NEXT:POKESO+4,32
507 FORW=1T04:GOSUB700:NEXT 307 FORM-1104:0030878878871871 510 LN=LN+1:POKE646,1:PRINTLEFT\$(V\$,8)LE FT\$(H\$,31)LN:POKE53280,1:POKE198,0 520 GOSUB1000 530 K1=0:GF=0:GOTOS50 590 591 592 REM CRASH LANDING

```
600 FORT≒0T06:POKE53265,(PEEK(53265)AND2
48)+T:POKE53281,T:NEXT
605_POKE53265,(PEEK(53265)AND248)+3:POKE
53281,0

53281,0

607 FORQ=1T0128:POKERS,192:POKERS,19:POK

ESO+1,Q:POKERS,196:FORP=1T030:NEXT:NEXTQ

610 POKESO+4,128:CR=CR+1:POKE646,1:PRINT

LEFT$(V$,12)LEFT$(H$,31)CR:POKE198,0

620 GOSUB1000
 630 K1=0:GF=0:GOTO350
 690
691
692
             REM
                                                     SOUND ROUTINES
692:
700 POKESO+4,33:FORT=1T0100:POKESO+1,15+
700 POKESO+4,33:FORT=1T0100:POKESO+1,15+
ABS(45-T):NEXT:POKESO+4,32:RETURN
800 POKESO+4,33:FORT=1T0255STEP 1:POKESO
900 POKESO+4,33:FORT=1T0255:POKESO+1,T:N
EXT:POKESO+4,32:RETURN
 990
991 REM
                                             WAIT FOR KEY PRESSED
992.
1000 PRINTLEFT$(V$,4)LEFT$(H$,4)"#PRESS
ANY KEY TO START"
1010 GETA$:IFA$=""THEN1010
1020 PRINTLEFT$(V$,4)LEFT$(H$,4)"
":RETURN
4991 REM
4992 :
                                                               SPRITE DATA
4999
                                   ROCKET - NO THRUSTER
0,16,0,0,48,0,0,252,0,0,252,0,
                 REM
5000 DATA
0.236.0.0
5010 DATA
                                   220,0,0,220,0,0,220,0,0,220,0,
0,252,0,0,252

5020 DATA 0,16,252,16,16,220,16,18,222,1

6,19,223,16,29,221,208

5030 DATA 29,221,208,29,221,208,28,220,2

08,16,48,16,16,0,16,0

5035 REM ROCKET - RIGHT THRUSTER

5040 DATA 0,16,0,48,0,0,252,0,0,252,0
    ,252,0,0,252
98,16,48,16,16,0,16,0
5035 REM ROCKET - RIGHT THRUSTER
5040 DATA 0,16,0,0,48,0,0,252,0,0,252,0,
0,236,0,0
0,252,0,0,252
5060 DATA 220,0,0,220,0,0,220,0,0,220,0,
0,252,0,0,252
5060 DATA 40,16,252,16,16,220,16,18,222,
16,19,223,16,29,221,208
5070 DATA 29,221,208,29,221,208,28,220,2
08,16,48,16,16,0,16,0
5075 REM ROCKET - LEFT THRUSTER
5080 DATA 0,16,0,44,0,0,252,0,0,252,0,
5080 DATA
                                   0,16,0,0,48,0,0,252,0,0,252,0,
0.236,0,0
5090 DATA 220,0,0,220,0,0,220,0,0,220,0,
0,252,0,160,252
5190 DATA 0,16,252,16,16,220,16,18,222,1
6,19,223,16,29,221,208
5110 DATA 29,221,208
5110 DATA 29,221,208,29,221,208,28,220,2
08,16,48,16,16,0,16,0
5115 REM ROCKET — BOTTOM THRUSTER
5120 DATA 0,16,0,0,48,0,0,252,0,0,252,0,
0.236.0.0
    ,236,0,0
5130
                                 220,0,0,220,0,0,220,0,0,220,0,
                DATA
5130 DATA 220,0,0,220,0,0,220,0,0,220,0,
0,252,0,0,252
5140 DATA 0,16,252,16,16,220,16,18,222,1
6,19,223,16,29,221,208
5150 DATA 29,221,208,29,221,208,28,220,2
08,16,32,16,16,136,16,0
5155 REM ROCKET - SPLAT
5160
                 О
5170
                 DATA
                                 0,0,0,0,0,0,124,0,1,255,128,7,
31/0 DR, 0 9,0,0,0,0,0,124,0,1,233,120,7,
255,192,31,255
5180 DATA 204,31,255,254,127,255,255,255
,255,255,125,255,222,11,255,238
5190 DATA 3.248,0,0,0,0,0,0,0,0,0,0,0,0,0,
0.0
5194
5195
               REM
                                                    CHARACTER DATA
                DATA255,255,255,255,255,255,255,255
DATA255,254,252,248,240,224,192,128
DATA255,127,63,31,15,7,3,1
DATA1,3,7,15,31,63,127,255
DATA1,28,192,224,240,248,252,254,255
DATA0,0,0,0,0,0,0
DATA0,0,0,0,0,0,0
DATA0,0,0,0,0,0,0
DATA0,0,32,240,240,252,254,255
DATA0,0,0,15,31,312,127,255
DATA0,0,0,15,31,31,127,255
DATA0,0,254,254,254,254,254,254
DATA255,129,129,129,129,129,129,1255
DATA3,15,63,0,255,63,15,3
DATA192,240,252,0,255,252,240,192
DATA24,24,24,24,24,60,126,255
5196
5200
5210
5220
5230
5240
5250
5260
5265
5280
5290
5300
53īē
5320
```

5340 DATA255,255,255,0,255,255,255,255 5350 DATA128,192,192,224,252,252,254,255 5360 DATA3,15,15,31,63,127,127,255

Squirrel Away

Frank Mikulastik

```
REM
 ĪØ
          REM
                                                 INITIALIZATION
12 :
30 POKE53248+21,0:DIM SP(2,2),BS(4)
32 POKE56578,PEEK(56578)OR3
34 POKE56576,(PEEK(56576)AND252)OR1
36 POKE56576,(PEEK(56576)AND252)OR0
38 POKE648,128:PRINT"O";
99 SM=32768
105 V=53248:CM=55296:S1=54272:M1=1016+SM
110 SP(1,1)=16:SP(1,2)=17:SP(2,1)=18:SP(2,2)=19:FG=1
120 BS(1)=20:BS(2)=21:BS(3)=22:BS(4)=21
130 GOSUB5000:FORI=16*64+SMT023*64-1+SM:READA:POKEI,A:NEXT
130 GUSUBS000:FORI≈16*64+SMT023*64-1
READA:POKEI.A:NEXT
160 POKEV+28,1:POKEV+29,1:POKEV+23,1
170 POKEV+37,0:POKEV+39,1:POKEV+39,9
180 FORI=1TO7:POKEV+39+I,0:NEXT
250 GOSUBS000
260 GOSUBS200
300 GOSUBS200
            GOSUB700
POKEV+31,
GOSUB4400
 310
330
340
497
             GOSUB2000
498 REM
                                                      MAIN LOOP
499
500
             GOSUB3000
500 GOSUB3000
510 POKES1+4,0:JV=PEEK(56320):FR=JVAND16
:JV=15-(JVAND15):IFFR<>16THENGOSUB2100
:JV=15-(JVAND15):JFR<>16THENGOSUB2100
520 ONJV+1GOSUB550,550,550,1500,1500
,1500,1600,1600,1600
530 IFJV>1THENPOKEV+16,RX:POKEV,X1:POKEV
+1,V1:POKEM1,SP(SD,SN):GOSUB2000
540 POKES1+4,0:GOTO500
550
           RETURN
685
690
           REM
                                                INITIALIZE SPRITES
695
700 RX=0
710 X=16:Y=2:X1=X*8+24:Y1=Y*8+53:SD=2:SN
=1
720 POKEM1,SP(SD,SN):POKEV+16,RX:POKEV,X
1:POKEV+1,Y1
725 FORI=1TO7
730 BX(I)=0:BY(I)=I*12+106:B1(I)=BX(I):B
2(I)=BY(I)
735 BN(I)=I:IFI>4THENBN(I)=I-4
740 POKEM1+I,BS(BN(I)):POKEV+16,RX:POKEV
+I*2,B1(I):POKEV+I*2+1,B2(I)
750 NEXT
750 NEXT
755 W1=104:W2=0
760 T1=32:T2=14:TP=0
790 RETURN
795
796
            REM
                                                PRINT PLAY SCREEN
797
/3/ .
800 PRINT"Dm":POKE53280,0:POKE53281,6
805 GOSUB4300:POKE54276,33
810 FORI=0T05*40-1:POKE56096+I,11:POKESM
+800+I,160:POKE54273,RND(8)*20+50:NEXT
```

PRINT" Seistatakkalalalalalalalalalalalalala PRINT" **PRODUK PRODUKS PR** 820 830 840 PRINT" 850 PRINT" 860 PRINT" 865 PRINT"# ## NUTS LEFT: 104 NUTS SAVED: 0 870 คีลั<u>เหา"**ฆณะสา**ต</u>สา 872 PRINT" 873 PRINT". 875 PRINT 43 T 878 PRINT" 890 RETURN 1495 : 1496 REM MOVE SQUIRREL LEFT 1497 1497 1500 X=X-500 X=X-1:IFX<1ANDSD=1THENX=1:RETURN 510 POKES1+4,33:POKES1+1,RND(8)*10+40:I SD=2THENX=X+1 500 V1=V40104.1504 rsu=2|HENX=X+1 1520 X1=X*8+24:|FX1>255THENRX=(RXOR1):X1 =X1-255:60T01535 1525 RX=(RXAND254) 1535 |FX=(RXAND254) 1535 |FX=1THENSH=2:8D=1:RETURN 1550 |SU=1:8D=1:BETURN 1535 1550 SH=1:SD=1:RETURN 1595 1596 1597 REM MOVE SQUIRREL RIGHT 1600 X=X+1:IFX>36THENX=36:RETURN POKES1+4,33:POKES1+1,RND(8)*10+40:I FSD=1THENX=X-1 1620 X1=X*8+24:1FA1/200.... =X1-255:GOTO1635 1625 RX=(RXAND254) 1635 IFSN=1THENSN=2:SD=2:RETURN 1635 IFSN=1THENSN=2:SD=2:RETURN 1620 X1=X*8+24:IFX1>255THENRX=(RXOR1):X1 1995 1996 1997 REM MOVE THE NUT 1797 2000 POKETP+SM,T1:POKETP+CM,T2:POKES1+11 ,0:CC=0:IFSD=2THENTP=X+3+(Y+2)*40:GOTO20 20 2010 TP=X-1+(Y+2)*40 2095 2096 REM DROP THE NUT 2097 2100 FORCC=0T0190STEP10 2105 IFPEEK(TP+40+SM)=1600RPEEK(TP+40+SM)=81THENCC=190:GOTO2180)=81HeNCC=130.00102103 2110 POKETP+5M,T1:POKETP+CM,T2:POKES1+11 ,33:POKE54280,255-CC:TP=TP+40 2120 T1=PEEK(SM+TP):T2=PEEK(CM+TP):POKET P+CM,9:POKETP+SM,81:GOSUB3000 2172 IF(PEEK(V+31)AND254)<>0THENCC=190:N 2172 IFKPEEK EXT:GOTO2200 2180 NEXT: POKES1+11,32:GOSUB4100:GOTO230 . 200 POKETP+SM,T1:POKETP+CM,T2:GOSUB4100 TP=0:T1=32:T2=14:POKEV+31,0 :230 W1=W1-1:PRINT"**3⇒DDDDDDDDDDDDD** 2230 **181**1" / 1 ์ พา 2240 IFW1<1THENGOSUB5100:GOSUB5200:GOTOS 00 2250 2256 2256 GOSUB2000: RETURN REM NUT DROPS INTO HOLE 2297 2300 POKES1+4,0:IFTP<800THENGOTO2200 2310 R=PEEK<TP+1+SM):IFR=1600RR=81THENTP =0:T1=32:T2=14:GOTO2350 2320 POKEV+21,1:POKETP+SM,32:TP=TP+1:POK 2320 POKEY-21,1:POKETP-SM,32:TP=TP+1:POKETP+SM,81:POKETP+CM,9
2335 POKEY+21,255:GOTO2310
2350 POKEY+21,255:GOTO2310
2350 POKEY+21,255:GOTO2310
2350 POKEY+21,1:POKES1+4,0:IFSN
22THENSN=1:SD=SD+1:IFSD>2THENSD=1
2360 POKEY+21,1:POKEM1,SP(SD,SN):POKES1+
11,0:GOSUB4000:POKEY+21,255:NEXT
2382 W2=W2+1:W1=W1-1
2384 PRINT"
2386 PRINT"
2388 PRINT"
2388 PRINT"
2388 PRINT"
2388 PRINT"
2380 PRINT"

```
2395
       GOSUB2000: RETURN
2995
2996
                        MOVE THE BIRDS
2997
3000 N=N+1:IFN>7THENN=1:U9=FRE(0)
3002 FORI=1TO7:BN(I)=BN(I)+1:IFBN(I)>4TH
ENBN(I)=1
3006
       PÓKĒM1+I,BS(BN(I)):NEXT
BX(N)=BX(N)+N*2+8:IFBX(N)>350THENBX
3010
⟨N>≕0
3020
      <sup>™</sup>B1<N>=BX<N>:RX=(RXAND<255-(211)>)
IFBX<N>>255THENRX=(RXOR<211)>:B1<N>
N>-255
3030 I
=BX(N)
3110
       POKĒV+16,RX:POKEV+N*2,B1(N):RETURN
3995
       REM
               NUT DROPS INTO HOLE SOUND
3996
4000
       POKES1+24,15:POKES1+4,33:POKES1+5,3
*16+3:POKES1+5,3
4005 R9=INT(RND(0)*100+50):POKES1,R9:POK
ESI+1, R9
4010 POK
       POKES1+4,0:RETURN
4095
                    NUT MISSES HOLE SOUND
4096
       MEM
4097
4195
4196
       REM
                   SQUIRREL MOVEMENT SOUND
4197
4200
       POKES1+1, RND(8)*10+40: RETURN
4295
4296
       REM
                       HEW SCREEN SOUND
4297
4300 FORJ=1TO4
4303 FOKES1+24,15:POKES1+4,17:POKES1+5,1
5:POKES1+5,15
4305 FORI=1TO200STEP20:POKES1,I:POKES1+1
+305 FORI=1T020
,I:NEXT
4310 POKES1+4,0
4315 NEXT
4320 RETURN
4390 :
4391
       REM
                      INITIALIZE SOUND
4392
4400
       FORI=54272T054296:POKEI,0:NEXT
4410 POKE54296,15:POKE54277,96:POKE54278
,240:POKE54276,33
4420 POKE54284,64:POKE54285,128:POKE5428
3,0:POKE54280,15:POKE54281,233:POKE5428
4430 POKE54291,64:POKE54292,128:POKE5429
0,0:POKE54287,55:POKE54288,33:POKE54289,
233
4440 RETURN
4995
4996
       REM
                       PRINT TITLE SCREEN
4997
       5000
281,5
5002
       PRINT"
       PRINT"
                                24
5003
                 100
                                         2
                               3
                                 7
                    ***
                        450
                                         3
                            200
                                         2 22 22 22
5004
       PRINT"
                  ä
                                                2
5995
       PRINT"
                               23
                                                550
                                         ã
       PRINT
5006
                                             28
                      3
                        5007
       PRINT"
                           73
                                      렴..
                  -
 -
                 13
                         귤
5008
       PRINT"
        *** ***
                         -
       PRINT"
PRINT"
5009
5010
       PRINT"
5011
                                                  3
       PRINT"
5012
       421
             ٠
52 MM
5013
      PRINT
       SI E
5014
                                2
5015
       PRINT"
    3
5016
       PRINT"
3 =
5024
       PRINT
               2
               "海豚豚豚豚豚豚豚豚豚豚豚豚豚 FRANK MIKULAS
5030
       PRINT
TIK
```

POKES1+11,0 5077 GOSUB4 5078 SN=5 GOSÜB4000 SN=SN+1:IFSN>2THENSN=1:SD=SD+1:IFSD >2THENSD=1 5079 POKEM1 5080 IF(PFF 5079 POKEM1, SP(SD, SN):FORL1=1T0100:NEXT 5080 IF(PEEK(56320)AND16)=160RPEEK(197)< >64THENGOT05077 5085 POKEY+1,0:POKEV+29,0:POKEV+23,0 Sē9ē RETURN 5095 GAME OVER REM 5096 5097 PRINT "溫級和**如為中央中央中央中央中** GAME 2 " 5100 OVER 5110 5120 POKE53281,14:POKES1+11,0 POKEV+23,0:POKEV+29,0 GOTO5077 5130 5195 5196 5197 PRINT INSTRUCTION SCREEN REM FORI=0T07:POKEV+I*2+1,0:NEXT
PRINT"D":POKE53281,7:POKE53280,9
GOSUB4300
FORI=0T07:POKEV+I*2+1,0:NEXT 5200 5201 5205 5210 PRINT PRINT PRINT 5211 5213 PRINT" 5214 PRINT"**២៤២២៧** 5215 PRINT"**៧៧** 3 医多多河沟 5216 PRINT".73 5218 PRINT", 78 5219 PRINT", 71588 PRINT:PRINT" YOU HAVE 104 NUTS WHI 5220 CH YOU WANT TO "
5221 PRINT" STORE IN YOUR BURROW.
UST DROP" YOU M UST US | DRUF" THEM CAREFULLY TO KEEP 5222 PRINT" THEM CAREFULLY TO KEEP BIRDS FROM" 5223 PRINT" EATING THEM." 5229 POKEY+40.0 >223 PUKEY+40,0 5230 POKEM1,16:POKEM1+1,21:POKEY,100:POK EV+2,200:POKEV+1,61:POKEV+3,90 5280 FORI=1TO1000:NEXT 5285 IF(PEEK(56320)AND16)=16THENGOTO5285 5287 POKEV,0:POKEV+1,0:POKEV+2,0:POKEV+4 ,0 5290 RETURN 5995 SPRITE DATA 5996 5997 REM SQUIRREL MOVING LEFT - 1 5998 REM 6000 DATA 0,0,0,0,0 4,5,80,20,21 84,20,85,85,85,85 85,85,85,85,85,85 DATA 6010 6020 6030 DATA 6040 6050 DATA DATA 85,85,85,21,85 85,21,84,25,5 64,5,2,130,169 2,138,169,42,170 170,38,170,170,106 170,170,106,170,170 50,162,170,10,128 170,10,128,40,42 0,168,0 SQUIRREL MOVING LEFT - 2 DATA 6060 DATA DATA DATA ēē7ē 6080 6090 6100 6110 6120 6130 6138 DATA DATA DATA DATA REM SQUIRREL MOVING
1,65,64
21,69,80,85,21
84,85,85,84,85
85,85,85,85,85
21,85,85,21,84
85,2,144,21,2
130,133,42,138,165
38,170,169,107,170,50
170,170,2,170,170
2,170,168,2,138
168,2,130,168,2 6140 6150 6160 6170 6180 6180 DATA DATA DATA DATA DATA DATA DATA 6199 6209 6219 6229 6239 6249 6259 DATA DATA DATA DATA DATA DATA REM 6260 6270 6278 ø SQUIRREL MO 1,65,64,5,81 84,21,84,85,21 85,85,85,85,85 SQUIRREL MOVING RIGHT - 1 DATA 6280 6290 DATA

```
85,85,85,85,85
84,85,21,84,84
6,128,82,130,128
90,162,168,106,170
152,106,170,169,170
170,169,170,170,140
170,170,128,42,170
128,42,162,128,42
130,128,10,2,128
10,130,160,0
SQUIRREL MOVING RIGHT -
6310
6320
6330
6340
6350
           DATA
DATA
DATA
            DATA
            DATA
6360
6370
6380
            DATA
6390
6400
            DATA
                       6408
            REM
6410
6420
6430
            DATA
DATA
DATA
6440
6450
            DATA
            DATA
6460
6470
6480
6490
6500
            DATA
           6510
6520
6530
6540
6545
6550
                               BIRD FLYING - 1
                        ø
6560
                        0.0.0.0.0
                        0,0,0,0,0,0
0,0,0,0,0,127
0,240,0,0,127
6,0,63,227,128
31,241,255,255,255
32,231,255,255,255
255,224,63,255,255
15,254,0,0,0
0,0,0,0,0
0,0,0,0,0
0,0,0,0,0
6570
6580
            DATA
            DATA
6590
            DATA
6600
6610
            DATA
6620
6630
6640
6650
            DATA
DATA
DATA
            DATA
6660
6670
6680
            DATA
                        0,0,0
BIRD FLYING - 2
0,0,0
0,0,0
            DATA
6689
6685
6699
6719
6729
6739
6749
            REM
DATA
DATA
            DATA
                        0.0.0.0.0
                        0,0,0,0,0
0,0,0,0,0
0,7,128,255,241
224,159,253,38,231
255,248,255,255,224
63,255,248,15,254
6,0,0,0,0
0,0,0,0,0
            DATA
            DATA
            DATA
6750
6750
6750
6780
6790
            DATA
            DATA
DATA
DATA
DATA
ēšīē
                        0.0.0.0.0
6820
6825
            DATA
                        ø
                               BIRD
                                          FLYING - 3
6830
            DATA
                        0,0,0,0,0
                        0,0,0,0,0
0,0,0,0,0
0,0,0,0,0
128,0,5,228,128
1,40,225,255,240
127,255,224,31,255
240,63,254,8,192
6840
            DATA
685<u>6</u>
            DATA
6860
6870
6889
            DATA
6890
6900
            DATA
6910
            DATA
                        0,4,0,0,0
                        0,0,0,0,0
6920
6930
            DATA
DATA
DATA
6940
6950
                         0.0.0.0
ട്ര്ള്ളര
20010
20020
              REM
                                         RESTORE MEMORY
              POKE648,4:PRINT"D":STOP
 20030
```

	r de la companya de La companya de la co
	그를 받고 하고 하셨다면 그런 그리고 하는데 되었다.
$\mathcal{E}_{i}(x) = \mathcal{E}_{i}(x) + \mathcal{E}_{i}(x) + \mathcal{E}_{i}(x) + \mathcal{E}_{i}(x) + \mathcal{E}_{i}(x)$	
and the second of the second o	l l
÷	

Ghost Hunt

Igor Tulchinsky

```
X1%=X2%+X3%+Y1%+Y2%+Y3%
                                                             MOVE FIGURE
8 REM MUVE FIGURE
9 :
10 BSX=PEEK(56320)
12 IF(NOTBSXAND16)=16THENN1X=NX:NX=NX+1:
NX=NX+3*(NX=4)
13 IFBSX(>255THENPOKE54283,17:POKE54280,
RND(8)*10
14 POKECL(N1X),9
15 POKE2040,192:POKECL(NX),7
19 J3X=-(NOTBSXAND4)/4+(NOTBSXAND8)/8:E3
X=-(NOTBSXAND1)+(NOTBSXAND2)/2
20 XX(NX)=XX(NX)+J3X*MM*1.5:YX(NX)=YX(NX)+E3X*MM*1.5
21 IFXX(NX)>=30ANDXX(NX)<=320ANDYX(NX)>=
100ANDYX(NX)<=30ANDXX(NX)=30
21 IFXX(NX)<30THENXX(NX)=30
23 IFXX(NX)<320THENYX(NX)=30
24 IFYX(NX)<100THENYX(NX)=30
25 IFYX(NX)<100THENYX(NX)=320
27 POKEPX(NX),XX(NX)AND255:POKEV+16,(PEE
K(V+16)AND(255-P(NX))OR(-(XX(NX)>255)*P
(NX))
30 POKEPY(NX),YX(NX)
 9
           PÓKEPY(NZ),YZ(NZ)
POKE54283,0
POKE2040,193:RETURN
 35
 40
47
 48
                                                                 MOVE GHOST
 50
51
52
           POKES+4,65:POKES+1,RND(8)*50+100
IFCT=40THENX8%=RND(8)*300+20:Y8%=RND(
52 IFCT=40THENX8X=RND(8)*300+20:Y8X=RND(8)*180+40:CT=0:GOTO60
53 CT=CT+1
60 IFX3X>X8XTHENDXX=-1
65 IFX3X>X8XTHENDXX=1
70 IFY3X>Y8XTHENDYX=-1
72 IFY3X>Y8XTHENDYX=-1
75 X9X=X3X+MX(1)*DXX:Y3X=Y3X+NX(1)*DYX:X
3X=X3X+MX(1)*DXX:Y3X=Y3X+NX(1)*DYX:X
3X=X3XX+GN(X3X)
76 IFY3X<100THENY3X=100
80 POKE2040,192:POKE54280,50:POKE54287,1
 5ĕ
85
 35 Y3X=Y3X*SGN(Y3X)
90 POKEY+16,(YPEK(Y+16)AND254)OR-(X3X)2
55):POKEY,X3XAND255
95_POKE2040,193:POKE54280,140:POKE54287,
55),,,
95 POKE2040,,,
220
100 POKEY+1,Y3XAND255
105 POKES+4,0
110 RETURN
INITIA
                                                          INITIALIZATION
  199
 199 :
200 GOSUB12000:GOSUB18000
205 GOSUB11000:GOSUB13000
206 TT=TT+1:TZ=INT(VL-TI/60):PRINT"%00000
"TAB$(18)TZ"!! "
"TAB$(18)TZ"!! "
 208 REM
                                                     MAIN CONTROL LOOP
 209
  210 Q=PEEK(V+30):GOSUB10:GOSUB50:Q1=PEEK
(V+30):IFQ1=0THENQ=PEEK(V+30)
```

211 TT=TT+1:TZ=INT(VL-TI/60):IFTT>6THENT
T=0:PRINT"SMMMM"TAB(18)TZ*-(TZ>0)"
212 IFTZ<=0ANDTT=0THEN15000
213 IFTZ<10THENPOKE53281,RND(8)*16:POKES
+1,RND(8)*100+100:POKES+4,33
230 IFG1<>29THEN210
231 POKEV+21,30:Z=PEEK(V+30):Z=SIN(8):Z=
PEEK(V+30):POKEV+21,31:IFZ<>0THEN210
238 GOSUB13120
239 POKE54276,0:POKE54290,0
240 POKE54283,33:POKE54280,0:FORI=255TO0
STEP-1:POKE54280,1:NEXT:POKE54283,0
250 NP=1000-VL*2-VAL(TI*):SC=SC+NP:GOSUB
16000:TI*="000000"
270 POKEV+21,0 16000:TI\$="000000"
270 POKEY+21,0
280 XX(1)=40:XX(2)=320:XX(3)=160:YX(1)=1
20:YX(2)=120:YX(3)=175
290 NX=1:GOSUB10:NX=2:GOSUB10:NX=3:GOSUB
10:NX=1:Q=PEEK(Y+30):POKEY+21,31
295 TT=TT+1:TZ=INT(VL-TI/60):PRINT"\$BORDO
"TABK(18)TZ"## "
"200 GOSUB12120:DOKEEY=2001 G:COTOCAC 300 GOSUB13120:POKE53281,0:GOTO210 9996 ÉND 9997 : 9998 REM GHOST 1 9999 7977 1000 DATA 0,56,0,0,124,0,0,254,0 10001 DATA 1,147,0,1,183,0,0,254,0 10002 DATA 1,255,0,3,255,184,3,125,252 10003 DATA 63,131,230,127,255,224,207,25 5,240 10004 DATA 15,255,240,30,239,120,29,239, 156 โตติตร. DATA 63,239,254,123,255,223,255,23 9,206 10006 DATA 249,199,132,112,131,0,32,0,0 10007 10008 REM GHOST 2 10009 DATA0,56,0,0,124,0,0,254,0 DATA1,147,0,1,219,0,128,254,0 DATA193,255,0,127,255,129,63,125,1 10010 10011 10013 DATA15,131,254,15,255,248,31,255,2 24 10014 DATA31,255,224,62,239,112,121,239, 176 10015 DATA255,239,248,251,255,220,31,255 .254 10016 DATA7,119,127,2,34,56,0,0,32 10017 10018 REM FIGURE : DATAØ,124,0,1,255,0,3,17,128 DATA7,255,192,6,186,192,3,255,128 DATA1,239,0,0,254,0,68,0 DATA0,56,0,0,16,0,0,56,0 DATA15,255,224,31,255,240,49,254,2 10020 10021 10022 10024 DATA96,252,12,96,252,12,48, DATA0,204,0,0,204,0,1,134,0 .48.204.24 10026 10997 REM PRINT GAME SCREEN \$: A \$ = A \$ + A \$ 11040 FORI=1T013:PRINT" ##"; A\$;:POKES+1,R ND(8)*255:NEXT:PRINTLEFT\$(A\$,16); 11050 FORI=0T039:POKE2023-I,160:POKE5629 S-I,4:NEXT J-1,4: MEX.| 11067 PRINT"\$":FRINTTAB<17>"\$\$\$CORE" 11068 FORI=1TO30 11069 Q=RND(8)*240:POKE1024+Q,46:POKE552 96+Q,1:POKES+1,RND(8)*255:NEXT:POKES+24, 11070 PRINT"9":PRINTTAB(17)"99SCORE" 11075 PRINT"9":PRINT:PRINT:PRINTTAB(15)" 997IME_LEFT9" 11075 I 11090 RETURN 11090 KE... 11997 KE... 11998 REM PRINT TITLE SUMMEDIA 11999 KE... 1200 POKE53280,11:POKE53281,0:PRINT"3"; 12001 POKE53269,0:FORI=54272T054296:POKE 12001 POKE53269,0:FORI=54272T054290,0 12001 POKE54290,0:POKE54290,0

12010 FORI=1TO24:PRINT"**:::** ";:NEXT 12020 PRINT" ";:POKE2023,160:POKE56295,0 Q%(1)=4:Q%(2)=6:Q%(3)=9 2025 \$1=0:\$2=39:\$3=0:\$4=24:FORI=1TO2:C= 12030 Q%(I) 2040 X=S1:FORY=S4T0S3STEP~2:G0SUB12500: MEXT 12050 Y=S4:FORX=S2TOS1STEP-2:GOSUB12500: HEXT 12060 X=S2:FORY=S3T0S4STEP2:G0SUB12500:N EXT 12070 Y=S3:FORX=S1TOS2STEP2:GOSUB12500:N ÊXŤ 12080 12090 12095 12100 PRINT" 12110 __ "B\$"# "B\$"■ "B\$" " יינדד. 12120 PRINT" "B\$" | " R# " 12130 PRINT" " P:# " "B\$" B # "1R4E" 140 PRINT" 12140 "B\$" "B\$" " Test " - "B\$" "B\$" 170 PRINT" 2170 "B\$" ■ ** TR4E ** ■ 中でませり 12180 PRINT" "B#" Ti 12190 PRINT" ■ "B\$" "B\$" "12210 PRINT:PRINT:PRINT:PRINT 12250 PRINT**"#R:PPPPPP**(C) 1984 USE, INC."; 1984 K |ULCHIMSKT 12260 RESTORE:V=53248 12270 FORI=12288T012350:READJ:POKEI,J:NE 12280 FORI=12352T012414:READJ:POKEI,J:NE $\times T$ 197 12300 POKEV+21,3:POKEV+39,1:POKEV+40,0:P OKEV,50:POKEV+16,0:POKEV+1,170:POKEV+29, 12310 POKE V+23,3:POKEV+28,0:X3%=160:Y3% =100 12320 POKEY+28,0:X3X=50:Y3X=170 S=54272:FORL=STOS+24:POKES,0:NEXT POKE198,0 12340 POKES+5,6:POKES+6,200:POKES+24,15: POKES+4,33:DX=3:FX=1:POKES4273,10:VZ=50 12350 X3X=X3X+DX:IFX3X=2540RX3X=50THENDX =-DX 12351 POKE53281,RND(8)*255 12352 Y3X=Y3X+FY:Y=18 12330 2=-FX 12355 %=-F%
12355 POKE54273, VZ: VZ=VZ+SI: IFVZ>2550RVZ
<13THENSI=-SI: VZ=VZ+SI
12360 POKEV, X3%: POKEV+1, Y3%: POKE2040, 192
-(PEEK(2040)=192)
12365 POKE53281, RND(8)*255
12370 POKE53281, RND(8)*255
12375 Q=PEEK(56320): IFQ=127THEN12390
12380 POKES+4, 33: POKES+24, 15: POKEV+21, 0: POKES+1, 50: Q=SIN(812, 1): RETURN
12390 RFM 12400 GOTO12350 12499 GOTO12499 12500 POKE55296+40*Y+X,Q%(RND(8)*3+1):PO KE54273,RND(8)*100:RETURN 12997 12998 REM SPRITE INITIAL IZATION 12999 13000 RESTORE: V=53248 FORI=12288T012350: READJ: POKEI, J: NE 13010 13020 FORI=12352T012414:READJ:POKEI,J:NE $\times \tau$ 13030 XT FORI=12416T012478:READJ:POKEI,J:NE

13070 POKE2040,192:POKE2041,193:POKE2042,194:POKE2043,194:POKE2044,194
13080 POKEY+21,31:POKEY+39,1:POKEY,160:P
OKEY+16,8:POKEY+1,100:POKEY+29,33
13090 POKE V+23,33:POKEY+28,0:X3X=160:Y3 X=100
13100 POKEV+4,40:XX(1)=40:YX(1)=RND(8)*1
00+100:POKEV+5,YX(1):POKEV+41,7
13105 POKEV+8,160:POKEV+9,175:POKEV+43,7
13110 POKEV+8,54:XX(2)=320:YX(2)=RND(8)*
100+100:POKEV+7,YX(2):POKEV+42,7
13111 PX(1)=Y+4:PX(2):POKEV+42,7
13112 PX(3)=Y+4:PX(2):POKEV+42;Y+5:PY(2)=Y+7:CL(1)=Y+4:PX(2)=Y+6:PY(1)=Y+5:PY(2)=Y+7:CL(1)=Y+41:CL(2)=Y+42:NX=1
13112 PX(3)=Y+8:PY(3)=Y+9:CL(3)=Y+43:P(1)=X+161:Y+160:Y+160:Y+175
13113 MM=5:MX(1)=RND(8)*MM+1:MX(2)=RND(8)*MM+1:MX(2)=RND(8)*MM+1
13114 NX(2)=RND(8)*MM+1
13115 NP=3:SC=0:TI\$="000000":YL=101:X8X=RND(8)*300+20:Y8X=RND(8)*100+100 %=100 RND(8)*300+20:Y8%=RND(8)*100+100 POKECL(1),9:POKECL(2),9:POKECL(3), 13116 9 **13117** 13119 :
13120 S=54272:FORL=STOS+24:POKES,Ø:NEXT
13120 S=54272:FORL=STOS+24:POKES,Ø:NEXT
13130 POKES+5,6:POKES+6,200:POKES+24,15:
POKES+4,17:Q=PEEK(V+31):Q=PEEK(V+30)
13150 POKE54284,223:POKE54285,223:POKE54
283,33:POKE54280,17:POKE54279,Ø
13170 PRINT"#880":PRINTTBK(17)"SCORE"
13180 RETURN
13996 F*** 13996 END 14997 14998 REM GAME OVER 14999 14333 15000 GOSUB13120 15005 FORI=255TO10STEP-5:POKES+1,I:NEXT: POKES+24,0:POKEV+21,0:POKE53281,0:POKE53 15998 REM ADJUST SCORE 15999 16000 SC=INT(SC) 16004 PRINT"399" 16005 MM=MM-(MM<15):M%(1)=RND(8)*MM+1:M% (2)=RND(8)*MM+1 SC\$=MID\$(STR\$(SC),2,LEN(STR\$(SC))-16010 16020 PRINTTAB(21-LEN(SC 54296,3:VL=VL/1.1:RETURN 17997 : LEN(SC\$));"##"SC\$:POKE 17998 REM PRINT INSTRUCTION SCREEN 17999 18000 POKE54296,15:FORI=1TO100:NEXT:POKE 54296,0 POKE53281,1:PRINT"""
PRINTTAB(14)"##GHOST HUNTE":PRINT: 18010 PRINT 3020 PRINTTAB(5)"THE CRAZY, NEIGHBORHOO GHOST" 18020 18030 PRINTTAB(5)"IS ON THE LOOSE, .篇" 18090 PRINT:PRINT 18100 PRINT"等7"TAB(5)"CAPTURE THE GHOST BY TOUCHING" 18105 PRINTTAB(5)"IT WITH ALL THREE OF T HE FIGURES" 37 PRĪNTTAB(5)"AT THE SAME TIME AND W NO" 18109 PRINTTAB(5)"FIGURE TOUCHING ANOTHE R FIGURE." 18110 PRINTTAB(5)"HURRY UP, TIME IS SHOR T." T." 18130 PRINT:PRINT:PRINT"Œ"; 18140 PRINTTAB(5)"YOU CAN SWITCH BETWEEN FIGURES" 10150 PRINTTAR(5)"BY PRESSING THE FIRE B 18150 PRINTTAB(5)"BY PRESSING THE FIRE B UTTON." OTTON. 18180 PRINT:PRINT:PRINT:PRINT" PRES S THE FIRE BUTTON TO BEGIN" 18190 IFPEEK(56320)=127THEN18190 18200 POKE54296,15:FORI=1TO100:NEXT:POKE 54296,0 18210 RETURN

Firecracker Boy

Charles Mott, Jr.

```
REM
REM
REM
REM
REM
1
2
3
4
5
5
7
7
7
                    REM
                                                                                            INITIALIZE VARIABLES
31 E
                             Z$(3)="m
                                                                                                                  20
                              Z$(4)="M
                            140
8 5
150
  a a
                             2 年 ( 7 ) 至 " 加
200
                         C$(2)="$$$$$$$$$$
                         210
 220
                             C#(4)="PBBI
                            C$(5)="##
 240 V (1) = "PPPPPPPPM3

250 V (2) = "PPPPPPPM3

250 V (2) = "PPPPPPPM3

PPP3 (2) (2) = "PPPPPPPPM3

260 V (3) = "PPPPPPPPM3
260 V$(3)="photopological state of the color of the color
                     350
M 1
360
                                                                                                                                                          ...
 364
```

365 REM 366 : 370 PRI 380 FOR PRINT TITLE SCREEN 970 PRINT"∏";:GOSUB480:PRINT"∰"; 380 FORT≖1TO8:PRINTX\$(T):GOSUB670:GOSUB5 00:NEXT 00:NEX| 390 FORL∞1TO60:PRINT"\$888888888999;; 400 FORT∞1TO8:GOSUB670:PRINTLEFT\$<Z\$(T), L):NEXTT:GOSUB500:NEXTL 410 FORL∞1TO50:PRINT"\$88888888888888888888888888888 , 420 FORT=1T05:GOSUB670:PRINTLEFT\$(C\$(T), L):NEXTT:GOSUB500:NEXTL 430 FORL=1T050:PRINT"**#MANAGEMENTALEMENTALE** PRINT MEDICAL SERVICE FOR THE TOTAL SERVICE 48ē 490 494 495 0=255 RETURN REM FALLING SOUND 496 500 510 520 IFO<000R0>255THEN0=1 POKES+1,0:0=0-1.5 RETURN 524 525 526 REM PRINT INSTRUCTION SCREEN 526:
530 POKES+5,10:POKES+1,10:POKES+24,15:PO
KES+4,128:POKES+4,129
540 FORT=1T0188:GOSUB670:NEXT
550 PRINT"COMMUNICATIONS
550 FORT=2T0359:PRINT"SATE";:NEXT
570 PRINT"COMMUNICATIONS
570 PRINT"SAMMUNICATIONS
670 PRINT"SAMMUNICATIONS
670 PRINT"SAMMUNICATION
670 PRINT"SAMM 590 PRINT" PRIS WATER BALLOONS ON THE LITT LE BOY'S ... LE BOY'S BUT SEED OF THE MAYOR BOY'S BUT SEED OF THE MAYOR DOES NOTHE BUT SEED OF THE MAYOR BOY SEED OF THE FOURTH OF JULY BUT SEED OF SEED OF THE FOURTH SEED OF THE FORTH SEED OF THE SEED OF THE FORTH SEED OF THE SEED OF 630 630 FORT=1T0400:GOSUB670:NEXT 640 FORT=1T0650:GOSUB670:POKE1343+JU,105 :JU=1:GOSUB650:NEXT 650 POKES+4,129:POKES+4,128 640 630 660 RETURN 664 665 666 REM POKE 1 BYTE OF SPRITE DATA 670 R ETURN READA: IFA<>256THENPOKELL; A:LL=LL+1:R 694 695 REM LOAD PART 2 <u> 696</u> 596 :
700 POKES+5,31:POKES+4,128:POKES+4,129
710 CLR:POKE52,48:POKE56,75
720 POKE631,19:POKE632,13:POKE633,82:POK
E634,85:POKE635,78:POKE636,13:POKE198,6
721 PRINT"###LORD"CHR*(34)"&& FIRECRACKER
"CHR*(34)",8":NEW:END 722 : 725 REM 726 REM 727 : SPRITE DATA
ONE SPRITE EVERY FOUR LINES 5000 DATA 0,255,0,3,85,192,3,150,192,7,8
5,208,5,85,80,1
5001 DATA 125,64,0,85,0,0,20,0,0,170,128
,3,234,240,12,251
5002 DATA 204,12,251,204,12,255,204,5,25
5,212,1,243,208,0,243,192
5003 DATA 0,243,192,0,243,192,0,162,128,3,243,240,0
5004 DATA 0,255,0,3,85,192,3,150,192,7,8
5004 DATA 0,255,0,1,25,0,0,20,0,0,170,12
5005 DATA 235,64,0,125,0,0,20,0,0,170,12
8,3,234,240,12,251
5006 DATA 204,48,251,195,48,255,195,20,2
55,197,4,243,196,0,243,192,0,243,192,0,162,128,3,243,240,0 5000 DATA 0,255,0,3,85,192,3,150,192,7,8

5008 DATA 0,255,0,3,85,192,3,150,192,7,8
5,208,5,125,80,1
5009 DATA 235,64,0,125,0,0,20,0,16,170,1
5010 DATA 205,240,251,195,0,255,192,0,25
5,192,0,243,192,0,243,192
5011 DATA 0,243,192,0,243,192,0,162,128,3,240,0
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,5,125,80,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,208,50,1
5,20 0 5041 DATA 0.0.0.0.0.63.255.255.48.0.3.33 .17.18.32.0 5042 DATA 2.33.17.18.8.0.8.9.17.24.8.0.8 .2.17.32 5043 DATA 2.0.32.2.17.32.0.128.128.0.34. 0,0,8,0,0 5044 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 5045 DATA 0.0.0.21.0.0.60.0.0.32.0.0.168 ,0.2.170 5046 DATA 0.10.90.0.9.90.128.9,90.128.2, 106.128.2.106.0 5047 DATA 0.170.0.0.40.0.0.0.0.0.0.0.0.0

5054, DATA 64,0,85,0,0,20,0,16,127,68,67, 247,241,76,255,205 5055 DATA 240,247,195,0,255,192,0,247,19 2,0,255,192,0,247,192,0 5056 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0, ø **5**057 0,0,0,0,0,0,0,0,0,0,0,0,0,0,12,0 DATA 3,0,0,12,0,0,12,0 .0.85.0 5059 DATA 5059 0.85.0.0 ,87,0,0,253,0,0,85,0,0 .85.0.0 5060 DATA 0.0.0. 0,0,0,0,0,0,0,0,0,0,0,0, ŠØ61 DATA 0.0.0.0.0.0.0.34.0.0.10.0.0.76 5062 DATA 49,0,0,192,0,3,0,0,12,0,0,12,0 5062 DHTH ,0,85,0 5063 DATH 95,0,0 5064 DATH 0.85.0.0 .85.0.0.85.0.0.85.0.0. ā Š065 DATA 0.0.0.0.0.0.0.0.0.0.0.136,0.0.36 .0.0 DATA 5066 48,0,0,200,0,3,0,0,12,0,0,12,0 .0.65.0 5067 DATA 0,85,0,0,85,0,0,85,0,0,85,0,0, 85.0.0 5068 ĎATA 0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0. 5069 Dete 0,0,0,0,0,0,0,0,0,0,32,0,2,0,0 5070 DATA 168,0,0,192,0,3,16,0,12,0,0,12 .0.0.85.0 5071 DATA 0,85,0,0,85,0,0,85, 8.8.85.8.A. 85,0,0 5072 D ĎATA 2 **5**073 DATA 0,0,0,0,0,0,0,0,0,0,0,0,8,0,8,0,0, 5074 DATA 64,0,6, 0.3.16.0.12.0.0.12.0 .0.85.0 5075 DATA ,0,85,0,0,85,0,0,85,0,0, 85,0,0 5076 D DATA .0.0.0.0.0.0.0.0.0.0.0.0. 5077 DATA .0.0.0.0.0.0.0.0.0.0.0. 0.0.0.0.0 32 5078 DATA 0.0.8, .0.12.0.0.12.0.0 .85.0 5079 DATA 85.0. 0.85,0,0,85,0,0, 85.0.0 5080 DATA 5081 DATA .0.0.0.0.0.0.0.0.0.0.0.0. ø 5082 DATA 0,0,32,8 ,0,8,0,0,26,128,0,12,0 70,85,0 5083 DATA 85,0,0 5084 DATA 0.0.0. 0.0.0.0.0.0.0.0.0.0.0.0.0.0. 9 5085 DATA 0.0.0.0.0 0.0.0.0.0.0.0.0.0.0.0.0. 5086 0,0,32,0,0,132,0,0,17,0,0,0,0, 0,85,0 5087 DATA 85,0,0 5088 DATA 0.85.0 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 9 5989 DATA 0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0. ĕ Š090 DATA 0,0,0,0,0,0,0,0,21,0,0,0,0,0,0,8 0,0 5091 0.0.0. 0.0.0 5092 DATA 0.0. 0.0.0.0.0.0.0.0.0. ø 5093 DATA 0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0. Ø 5094 DATA 0,0,0,0,0,0,0,0,8,0,0,0,0,2,0, ē 5095 0.0 DATA 0,0,32,0,0,0,0 0.0 5096 DATA 255,255,255,0,5,87,0,15,255,0, 15,254,0,15,254,0 5097 DATA 15,255,0,15,255,0,15,255,0,15, 255,0,14,255,0,14 5098 DATA 255,0,15,255,0,15,255,0;15,255,0,15,255,0,15,255,0,15,255,0,15,255,0,5

121

MAIN GAME LOOP

220

REM

240 250 GOSUB570 ONDR(JV)GOTO260,260,270,260,260,260, 280,260 260 270 280 GÖŤ0290 260 GOT0290
270 MX=MX+AP:GOT0290
280 MX=MX+AP:GOT0290
280 MX=MX-AP:GOT0290
290 IFMX<30THENMX=30
300 IFMX>240THENMX=240
310 POKEV,N:N=N+(PH*PU):IFN>240ORN<25THE
NPU=-PU:N=N+(PH*PU)
320 IFFG(3)=5ANDFG(2)=5THEN350
320 POKEV+12,BO:BO=BO+HH:IFBO>244THENBO=
0:POKEV+12,BO:FC=INT(RND(1)*211+30):FG(3)=5)=5
340 FC=(INT(FC/2))*2
350 IFFC(BOANDFG(2)=0THENBA=B0:POKEV+10,
BA:FG(2)=5
360 IFFG(2)=0THENPOKEV+10,B0
370 IFFG(2)=5ANDNU)>ENTHENGOTO770
380 IFFG(4)=0THENPOKES7,217
390 IFFG(4)=5THENPOKES7,218:FG(4)=0:GOTO 416 FG(4)=5:WA=6:GOSUB1440 IFFG(2)=5THENNU=NU+1 IFFG(2)=0THENNU=0 IFFG(3)=5ANDFG(2)=0THENFG(3)=0 4<u>6</u>6 410 430 440 450 GOSUB1280 450 POKEV+6,AU:AU=AU+XX:IFAU>(253-XX)THE NGOSUB1480 460 IFFG(15)=5ANDAU=108ANDPEEK(V+16)=136 THENPR=PR+1:POKES4,PR:IFPR>224THENPR=219 :POKES4,PR ·FUNES4,FR 470 IFFG(1)=5THENPOKEV+2,BX:POKEV+3,81+B Y:BY=BY+AP:GOSUB1430:IFBY>123THENGOSUB61 IFFG(1)=0THENPOKEV+27,PEEK(V+27)OR2
IFFG(1)=0THENBY=0:POKEV+2,MX
POKEV+4,MX:IFFR=16THEN240
POKEV+27,PEEK(V+27)AND(255-2)
IFFG(1)=5THEN240
FG(1)=5:BX=MX:GOTO240 480 490 500 510 52ŏ 520 FG(1)=5:BX=MX:GOTO240 540 : 550 REM READ JOYSTICK 560 : 570 JV=PEEK(V+JS):FR=JVAND16:JV=15~(JVAN D15):RETURN sēē 590 REM NEW BALLOON 600 őÍØ POKEV+2.MX:POKEV+3,81:FG(1)=0:BY=6:G OSUB1410:RETURN 620 : 630 REM 630 REM INITIALIZE SPRIJES
640:
650 POKES1,202:POKES2,203:POKES3,205:POK
ES4,219:MX=130
660 POKES5,216:POKES6,206:POKES7,217:POK
ES8,192:POKEV+29,8:POKEV+23,8
670 POKEV,N:POKEV+1,140:POKEV+2,MX
680 POKEV+8,81:POKEV+4,MX:POKEV+5,81
700 POKEV+6,AU:POKEV+7,184:POKEV+8,237
710 POKEV+9,197:POKEV+10,BO:POKEV+11,195
730 POKEV+12,BO:POKEV+13,197
740 POKEV+16,PEEK(V+16)OR128:POKEV+14,40
750 PRINT"D=":GOSUB1000
760 RETURN INITIALIZE SPRITES 760 770 780 REM GAME END 79ē /30 . 800 POKEY+8C+4,128:FORT=1T099:NEXT 810 FORT=206T0213:POKE86,T:FORY=1T040:NE XTY,T:POKEY+8C+4,129:POKE86,214 820 FORT=1T040:NEXT:POKE86,215:FORT=1T01 820 FORT=1T040:NEXI:FOREOU. 00:NEXT 830 FOKEV+21,223:POKES6,206:JU=0:BY=3:M= 173:POKEV+15,176:FORT=1T060:NEXT 840 FORT=192T0201:POKEV+15,M-JU:BY=BY+5 :GOSUB1430:POKES8,T:JU=JU+5 850 FORY=1T030:NEXTY,T: 860 FORT=201T0192STEP-1:POKEV+15,M-JU:BY =BY-5:GOSUB1430:POKES8,T:JU=JU-5 870 FORY=1T030:NEXTY,T:POKEV+15,173 880 : =BY-870 880 890

```
AU=0:FORT=1T010:FG(T)=0:NEXT:BP=-10:GOSU
Bĭ340
960 GOSUB610:BP=0:GOSUB650:PU=1:POKEY+21
,255:POKE53265,PEEK(53265)OR16:GOTO240
970:
980 REM
                        PRINT GAME SCREEN
998
1000 PRINT" LJS =
   "湖 湖
10 PRINT" 為
                             ٠٠ ;
1010
                                      ■FIRE STATION #1L
                             ; יי
1020
        PRINT"⊾SI
        1030
1040
                                                                22
1050
1060
ESH**PARADE**";
1070 PRINT"LS es es
1090 PRINT" ... 2 ...
1110 PRINT" ... 38 .
1120 PRINT"⊾a
1130 PRINT" L. 21
1150 PRINT" LS =
1160 PRINT" La
1170 PRINT"
     EMILITIA
O PRINT'LE
1180
1190
        PRINT"⊾# =
        1200
     -
1210 PRINT"
                        · .
1220 PRINT" # PEL
                 יינר:
1230
         PRINT" = BBBBBCORE : E
                                                             SETIRECR
1250 PRINT" ####

CKER-POINTSES"

1240 RETURN

1250 :

1260 REM C

1270 C
                         COLLISION DETECTION
1270 .
1280 Q1=PEEK(V+30)
1290 L1=Q1AND1:IFL1=1THENGOSUB610:BP=BP-
2:GOSUB1550:RETURN
1300 L2=Q1AND2:L3=Q1AND4:L6=Q1AND32:L7=Q
IAND64
BP=BP-11:0050B1340
1330 RETURN
1340 POKEV+10,BO:POKES6,206:GOSUB610:GOS
UB1410:FORT=1T010:FG<T)=0:NEXT:BP=BP+10
1350 GOSUB1550:POKES2,203:RETURN
1350
1380
1390
1400
        REM
                           SOUND ROUTINES
1490 :
1490 :
1410 POKEV+SC+4,129:POKEV+SC+1,12:POKEV+
SC+4,128:RETURN
1430 S1=BY:POKEV+SC+4,33:POKEV+SC+1,S1:P
OKEV+SC+4,32:RETURN
1440 S1=WA:POKEV+SC+4,33:POKEV+SC+1,S1:P
OKEV+SC+4,32:RETURN
1450 :
1460 REM MOVE FLOAT ACROSS SEAM
1470 :
1480 IFPEEK(V+16)=128THENPOKES4,13:POKEV
+16,136:AU=0:POKEV+6,AU:POKES4,PR:FG(15)
1480 IFPEEK(V+16)=128THENPOKES4,13:POKEV
+16,136:AU=0:POKEV+6,AU:POKES4,PR:FG(15)
=5:RETURN
-3:REIDRN
1490 IFPEEK(V+16)=136THENPOKES4,13:AU=0:
POKEV+6,AU
1500 POKEV+16,128
1510 POKES4,PR:RETURN
```

Into the Pot

Frank Mikulastik

```
REM
REM
REM
  123
  4
5
10
  112
11 REM INITIALIZATION
12
13 S1=54272:CM=55296:SM=49152:V=53248
19 POKE53280.6:PRINT"D":PRINTCHR$(142):G
OSUB4030
20 POKE56578,PEEK(56576)OR3
21 POKE56576,(PEEK(56576)AND252)OR0
22 POKE53272,(PEEK(56576)AND252)OR0
23 POKE648,192:PRINT"D"
30 POKE56334,PEEK(56334)AND254
40 POKE53281,9:GOSUB3900:POKE54273,20:PO
KE1,PEEK(1)AND251
50 FORI=0TO511:POKEI+14336+49152,PEEK(I+53248):NEXT
60 POKE1,PEEK(1)OR4
70 POKE56334,PEEK(56334)OR1
80 POKE1,PEEK(1)OR4
70 POKE56334,PEEK(556334)OR1
80 POKE56334,PEEK(556334)OR1
80 POKE56334,PEEK(556334)OR1
80 POKE563272,PEEK(55272)AND240)OR14
90 F=0:GOSUB3000:F=1
100 FORI=14336+49152+160*STO14336+49152+
168*8+7:READA:POKEI,A:NEXT
110 FORI=49152+16*64TO49152+18*64-1:READ
A:POKEI,A:NEXT
200 FORI=0TO24:POKE54272+I,0:NEXT
210 POKE53270,PEEK(53270)OR16
220 POKE53282,3:POKE53283,2
300 S(1)=163:S(2)=164:S(3)=163:S(4)=165
305 S2(1)=160:S2(2)=161:S2(3)=160:S2(4)=
162
310 BN(1,1)=0:BN(1,2)=1:BN(2,1)=0:BN(2,2)
                REM
                                                                           INITIALIZATION
  305
162
   310
                    BN(1,1)=0:BN(1,2)=1:BN(2,1)=0:BN(2,2
  320°
                    BN(3,1)=1:BN(3,2)=0:BN(4,1)=-1:BN(4,
  ž5≅ø
   350 GOSUB745:POKEM1,SP:GOSUB3000
  354 :
355 REM
                                                                                     START CAME
   356
   400
                   GOSUB900
                    GOSUB700
TI$="000000"
  410
   420
   490
  491
492
500
505
                                                                                         MAIN LOOP
                  GOSUB1000
GOSUB4000
FOR BM=1TONU
GOSUB1500
   506
  510
515
 515 NEXT

520 FORBH=1TONU: IFP2(BN)<>552THENGOT0529

521 FOKESM+P2(BN),32: POKESM+473+NU,S2(1)

:POKECM+473+NU,9

522 FORBP=BNTONU-1: P2(BP)=P2(BP+1): NEXT

525 NU=NU-1: IFNU=ØTHENGOT02000

527 GOT0530

529 NEXT

530 GETA*: IFA*="B"THENT2*=TI*: POKES1+4,0

:POKES1+11,0: GOT02100

540 PRINT"EXPRENDED BRANDED BRANDED BRANDET ; MID*(TI*,

3,2); "PROPRED RESPONDED BRANDED BRANDET ; MID*(TI*,

550 SP=SP+1: IFSP>17THENSP=16

555 POKEM1,SP
```

Into the Pot Program

590 GOTO500 689 : 699 699 700 REM INITIALIZE CHARACTERS : N1=1:P1=549 POKECM+P1,8:POKESM+P1,S(N1) N2(1)=1:P2(1)=82:N2(2)=2:P2(2)=162 POKESM+P2(1),S2(N2(1)):POKECM+P2(1), 710 720 730 740 POKESM+P2(2),S2(N2(2)):POKECM+P2(2), 9
745 M1=49152+1016:SP=16
750 POKEV+21,1:POKEM1,SP:POKEV+29,1:POKE
V+23,1:POKEV+28,1
760 POKEV+37,7:POKEV+38,2:POKEV+39,0:POK
EV+16,1:POKEV,28:POKEV+1,131
770 POKE53282,2:POKE53283,6
780 NU=2:NM=2
790 RETURN
791 :
792 REM PRINT PLAY SCREEN
793 : 900 PRINT"": POKE53281,12: POKE53280,0:GO SUB4030 902 903 FORI=0T039 POKEI+CM, 14: POKEI+SM, 167 904 NEXT 906 907 FOR I = 39T0999STEP40 POKEI+CM, 14 POKEI+SM, 167 908 NEXT 911 912 913 916 POKEI+CM, 14: POKEI+SM, 167 FORI=960TO0STEP~40 917 918 POKEI+CM, 14 POKEI+SM, 167 NEXT 921 922 FORI=3T037STEP2:L=L+1:IFL>1THENL=0 FORJ=L+2T022STEP2 923 P=I+J*40:POKEP+CM,8:POKEP+SM,166:GOS UB4040 924 NE: NEXT:NEXT FORI=439TO431STEP-1 POKEI+CM,6:POKEI+SM,167 926 927 928 NEXT FORI=431T0680STEP40 931 932 POKEI+CM, 6: POKEI+SM, 167 HEXT 933 936 FORI=711TO719 POKEI+CM-40,6:POKEI+SM-40,167 937 938 230 NEA| 940 FORI=1T040 941 X=INT(RND(0)*38+1):Y=INT(RND(0)*23+1):POKESM+X+Y*40,167 942 POKECM+X+Y*40,6 942 943 NEXT 96 ī 962 963 964 968 969 PRINT" SEREED BEEFE SEREED BEEFE BEEFE 970 PRINT"**Side Control Control Control Control Control** Control Contr 980 RETURN 989 990 REM MOVE COOK 991 JV=15-(PEEK(56320)AND15)
IFJV=10RJV=9THENT1=0:T2=-1:GOTO1100
IFJV=80RJV=10THENT1=1:T2=0:GOTO1100
IFJV=20RJV=6THENT1=0:T2=1:GOTO1100
IFJV=40RJV=5THENT1=-1:T2=0:GOTO1100
RETURN 1000 1020 1040 1050 1060 RETURN
T3=P1+T1+T2*40:T4=T3:Q1=PEEK(T3+SM)
IFQ1=32THEN1200
IF(Q1<>166)THENRETURN
FORI=1TONM
T4=T4+T1+T2*40:Q2=PEEK(T4+SM)
IF(Q2=32)THENGOTO1150
IF(Q2=32)THENGOTO1150
IF(Q2<>166)THENRETURN
NEXT
RETURN
POKET4+SM,166:POKET4+CM,8
N1=N1+1:IFN1>4THENN1=1
POKEP1+SM,32:P1=T3:POKEP1+CM,8:POKE 1100 1100 11120 11120 11144 11144 11146 11150 1121 1210 | P1+SM 1290 | MUSCHID RETURN 1490 1491 REM MOVE CHICKEN 1492

Into the Pot **Program**

```
R=INT(RND(0)*4+1)
T3(BM)=P2(BM)+BN(R,1)+BN(R,2)*40
Q1=PEEK(T3(BM)+SM):IFQ1=32THEN1600
1500
1510
1520
1590
        RETURN
N2(BM)=N2(BM)+1:IFN2(BM)>4THENN2(BM
1600
1686 R25M7=R2(BM7+1:1FR2(BM7)24THERR2(BM
)=1:U9=FRE(0)
1610 POKEP2(BM7+8M,32:P2(BM7=T3(BM7:POKE
P2(BM7+CM,9:POKEP2(BM7+SM,S2(N2(BM7)
1690 RETURN
ī 990
1991
1992
         REM
                              WIN SITUATION
2000
         POKE53282,7:POKE53283,7
         GOSUB4020
IF(PEEK(56320)AND16><>16THEN2080
POKE53282,2:POKE53283,2
2002
2010
2015
2020
2030
2035
         GOSUB4020
GOSUB4020
IF(PEEK(56320)AND16)<>16THEN2080
GOTO2000
POKE53282,10:POKE53283,2
2080
         COSUBSIOO
2090
2091
2095
         GOTO400
2096
                            NO WIN SITUATION
         REM
2097
2100 PRINT" MEN PRESS Y TO STOP. PRESS N
TO CONTINUE."
        GETA$:IFA$<>"Y"ANDA$<>"N"THENGOTO21
2105
05
2110
2115
         IFA$="Y"THENGOSUB3100:GOTO400
2115 TI$=T2$
2115 TI$=T2$
2120 PRINT"跨湖2/////////=911N:";MID$(T
I$,3,2);"站///豐SEC:";RIGHT$(TI$,2);
2121 PRINT"站///////
2190
        GOTO590
2191
2192
2193
        REM
                          PRINT TITLE SCREEN
         POKE53280.6:POKE53281.5
PRINT"%"
IFF=@THENPRINT"COM"
3ēēē
3002
3004
3007
         PRINT
PRINT
PRINT
3008
3009
3011 PRINT":
                                      : 1485분 - 1485분 148888885분
1485분 148888분"
                        388.8.8.8.8
                                                                          :3
        M&&&&&
                            218.
& & 🖷
3012
                                               548.55
548.55***
                                                           248.55
                           246.
                                      24S. & 🚃
                                                                       214. 5
38.5 38.5
3013 PRINT"
                               -10. E
                                        24.C. 200
                                      288888
                                                       34S.
                                                                   52 C. .
                           38.8.8.8.
                                        24%. &. 55"
3014 PRINT"
                           22.6. WH
                                      #40.0. BE
                                             24.C.A.
                                                           348. 🕮
                                                                       52.C. 100
3015 PRINT"
                              -
54&.≡
                                        348.EE 348.EE"
                                                24S.
                           248. 💻
                                      348. 33
                                                           248. SS
                                                                      22.C. 1991
        PRINT"
                                        318. E 316. E"
                               344. E
3016
                      38.8.8.8.8.
                                      :10 m
                                                22.C. III
                                                           がおい
                                                                        22.0
& MARINT S018 PRINT"
                           248. E
                                    248.■
                                            #48.6.6.⊞"
                                                                        23.9
                                            製品品品の産業
                                                           調整を必要
8.8.8.8.
3019 PRINT"

3020 PRINT"
                                            32.6. SS
                                                     22A. WE
                                                             32.0 mm
                                                                        240
                                            340.00 SAC = 340.00
                                                                     24.0. 199
3020 FRINT
3025 PRINT"
308题"
                                            348. SS
                                                         388.55
                                                                     24 S. 🚃
3030 PRINT"
                                            529.00
                                                         24.Q. 🗰
                                                                     24.C. 188
    248.EE''
3031
        PRINT"
                                                           288.8.8.5
3032
        PRINT
3032 PRINT

3033 PRINT

3034 PRINT

3034 PRINT BY FRANK MIKULASTI

K":PRINT:PRINT

3038 PRINT" (C) 1984 FANFARE HOUSE

, INC."

3039 IFF=0THENGOSUB4030:RETURN

3040 POKE53282,2:POKE53283,2:FORI=1TO200
NEXT
3041
3042
3043
         GOSUB4000
         SD=SP+1:IFSP>17THENSP=16
POKEM1,SP
IF(PEEK(56320)AND16><>16THENGOTO309
3045
ø
3050
         POKE53282,7:POKE53283,7:FORI=1T0500
: NEXT
3051
3051 GOSUB4000
3055 IF(PEEK(56320)AND16><>16THENGOTO309
3060
         GOTO3040
3089
         REM
                       PRINT INSTRUCTION SCREEN
3091
```

Into the **Program**

3100 POKEV+1,0:POKEV,0
3101 PRINT"D":POKE53281,1:POKE53280,15
3110 PRINT:PRINT" MINTO THE POT":PRINT:PRINT
3111 PRINT" BY FRANK MIKULASTI POT":PRINT:PRINT
3111 PRINT" BY FRANK MIKULASTI
K":PRINT
3112 PRINT: (C) 1984 FANFARE HOUSE,
INC.":PRINT: PRINT
3121 PRINT: BOBJECT: GUIDE THE CHICKE
NS TOWARD"
3122 PRINTTAB(11); "THE COOKING POT BY PU
SHING"
3123 PRINTTAB(11): "BUOCKS TO MAKE A BATH BY PU SHING" 3123 PRINTTAB</11>; "BLOCKS TO MAKE A PATH FOR" 3124 PRINTTAB</11>; "THEM TO FOLLOW. SEE H OW FAST" 3125 PRINTTAB</11>; "YOU CAN GET BOTH CHIC PRINTTAB<11>;"YOU CAN 3125 "
KENS "
\$126 PRINTTH:
\$127 PRINT
\$128 PRINT"
\$100 WIN \$ PRINTTAB(11); "MINTO THE POTE." HINTS: M IF YOU GET INTO 3129 | 3129 | TO" PRINTTAB(11);"SITUATION OR YOU WANT 3130 PRINTTAB(11); "PAUSE, PRESS F7." 3131 PRINT:GOSUB4030 3132 PRINTTAB(11); "@PRESS F1 TO STAR 3132 PRINTTAB(11); "PRESS F7 TO STOP" ƹ tò START" 3133 3140 3142 3144 3146 3900 GÈTA≸ GE! N# IFA\$="#"THENRETURN IFA\$="#"THENPOKE648,4:PRINT"]]":STOP GOT03140 FORI=54272T054296:POKEI,0:NEXT POKE54296,15:POKE54277,13:POKE54278 :POKE54276,33:RETURN 3910 3910 FUN 15 :POK 3997 : 3998 REM 3999 : SOUND ROUTINES 4000 REM SOUND 1 4001 POKES1+4,0:POKES1+24,5:POKES1+4,17: POKES1+5,15:POKES1+6,15 4002 S3=INT(RND(0)*20+5):POKES1+1,S3 4010 REM SOUND 2 4011 POKES1+11,0:POKES1+24,5:POKES1+11,1 7:POKES1+12,15:POKES1+13,15 4012 S3=INT(RND(0)*10+1):POKES1+7,S3:POK ES1+8,S3 4014 FORS4=1TO10:NEXT 4019 RETURN 4020 REM +020 REM SOUND 3 4021 POKES1+4,0:POKES1+11,0:POKES1+24,10 :POKES1+4,17:POKES1+5,15:POKES1+6,15 4022 FORS2=50T0150:POKES1,S2:POKES1+1,S2 :NEXT 4023 POKES1+4,0 4029 RETURN 4020 POKES1+4,0 4023 POKES1+4,0 4029 RETURN 4030 REM SOUND 4 4032 POKES1+4,0:POKES1+11,0:POKES1+24,15 :POKES1+5,15:POKES1+6,15:POKES1+4,17 4034 FORS2=4TO100STEP2:FORS3=S2TOS2-4STE P-1:POKES1,S3:POKES1+1,S3:NEXT:NEXT 4038 POKES1+4,0 4039 RETURN SOUND 5 4037 RETURN SOUND 5 4041 POKES1+24,15:POKES1+4,17:POKES1+5,1 5:POKES1+6,15 4042 POKES1,I*5:POKES1+1,I*6 4044 POKES1+4,0 4043 RETURN 4042 4044 4048 4099 4091 4092 REM CHARACTER DATA CHICKEN 255,190,255,215,255,60,60,255 255,190,255,215,255,60,63,240 255,190,255,215,255,60,252,15 4094 DATA DATA DATA REM COOK DATA DATA 0,255,190,255,215,255,60,255 0,255,190,255,215,255,63,240 0,255,190,255,215,255,252,15 DATA DATA 4200 255,235,215,235,215,235,215,25 4210 DATA 255,255,255,255,255,255,25 4220 DATA 255,255,255,255,255,255,25 รัดดด 5001 5002 5135 REM SPRITE DATA POT AND FIRE -REM

Into the Pot **Program**

```
$140 DATA 0.0.0
$150 DATA 0.0.0.0
$1160 DATA 0.0.0.0
$1170 DATA 0.0.0.0
$1180 DATA 0.0.0.0
$1190 DATA 0.0.0.128.0
$1190 DATA 2.170.166,170,158,107
$1210 DATA 1.74.169,170,158,107
$1210 DATA 1.54.167,1231,128,128
$1220 DATA 1.54.167,1231,123,251
$1230 DATA 234,153,251,238,191
$1240 DATA 234,153,251,238,191
$1240 DATA 234,153,251,15,233
$1250 DATA 116,15,123,251
$1250 DATA 116,15,123,252
$1270 DATA 0.0.0.0
$1280 DATA 0.0.0.0
$1280 DATA 0.0.0.0
$1280 DATA 0.0.0.0
$1290 DATA 0.0.0.0
$1310 DATA 0.0.0.0
$1320 DATA 0.0.0.0
$1330 DATA 1.70,169,169,170,171
$1350 DATA 1.70,169,169,170,171
$1360 DATA 1.86,235,57,186,231
$1370 DATA 1.86,235,57,186,231
$1380 DATA 1.86,235,251,112
$1390 DATA 0.251,192,0
$1300 DATA 0.251,192,0
$1300 DATA 0.251,192,0
$1300 POKE648,4:PRINT"D":STOP
```

Into the Pot Program



Yorick's Revenge

Igor Tulchinsky

```
REM
REM
12345678
       REM
       REM
      REM
                                                  INITIALIZATION
 0 Q=PEEK(53279):GOSUB20000:POKEV+12,160:POKEV+13,60:POKEV+21,64:GOSUB24000:P11GOSUB24000:F=0:PRINT"]
":POKE532000:F=0:POKEV+21,0
 18
        REM
                                                     CONTROL LOOP
19:
20 Q=PEEK(197): IFQ<>64THENPOKE2040,194:POKE2041,195:GOTO40
30 POKE2040,192:POKE2041,193
40 X=X+DX:IFX>255THENX=255
41 Y=Y+DY:POKE53252,X:POKE53253,Y:POKE54
279,(240-Y)/8
42 DY=DY+.4
50 Z=PEEK(53279)ORPEEK(53278)ORPEEK(53279)ORPEEK(53279)ORPEEK(53279): IFZ=0THEN20
51 Z=PEEK(53278)ORPEEK(53278): IFZ<>0THEN
150
57:
58 REM SPRITE-BACKGROUND COLLISION
 19
 58 REM
                              SPRITE-BACKGROUND COLLISION
59: G=SIN(8): Q=PEEK(53279)

100 IFPEEK(1024+40*INT((Y-34)/8)+INT((X-14)/8))=32THEN80

70 DY=-DY: Y=Y+3*DY: POKE53253, Y: POKE54283, 65: G=SIN(8): Q=PEEK(53279): DY=DY+1.7

80 IFPEEK(1024+40*INT((Y-50)/8)+INT((X-14)/8))=32THEN100

90 DY=-DY: Y=Y+3*DY: POKE53253, Y: POKE54283, 65: Q=SIN(8): Q=PEEK(53279)

100 IFPEEK(1024+40*INT((Y-42)/8)+INT((X-22)/8))=32THEN120

110 DX=-DX: X=X+3*DX: POKE53252, X: POKE54283, 65: Q=SIN(8): Q=PEEK(53279)

120 IFPEEK(1024+40*INT((Y-42)/8)+INT((X-7)/8))=32THEN140

130 DX=-DX: X=X+3*DX: POKE53252, X: POKE54283, 65: G=SIN(8): Q=PEEK(53279)

140 POKE54283, 0: GOTO20

147: GEMM
 59
 140 FOR
147 :
148 REM
149 :
150 Q≔PI
152 IFQ
SKULL COLLISION
             POKE54276,17:IFQ<>12ANDQ<>20ANDQ<>36
 THEN200
161 POKE54290,65:POKE54287,RND<8>*100+50
170 SC=SC+100:IFQ=36THENSC=SC+400
```

Yorick's Revenge Program

```
'IFQ=12THENPOKEV+42,RND(8)*14+1
IFQ=20THENPOKEV+43,RND(8)*14+1
IFQ=36THENPOKEV+44,RND(8)*14+1:DY=RN
198 REM
                                GUTTER COLLISION
 199
231
232
 233
        REM
                          FLIPPERS DOWN COLLISION
234
235
        IFQ=50RQ=6THENPOKE54290,65:POKE54287
,25
250
 250 IF(Q=50RQ=6)ANDPEEK(2040)=192THENDY=
-DY+1:Y=Y+2*DY:POKE53253,Y
260 IF(Q<>5ANDQ<>6)ORPEEK(2040)<>194THEN
POKE54290,0:GOT020
267
268
                            FLIPPERS UP COLLISION
         REM
269
270
269 :
270 IFX>110THEN350
270 IFX>110THEN350
280 T=TAN(DX/(DY+1E-3)):L=SQR(DX*DX+DY*D
Y)*2*(50-(115-X))/30:IFLCHL=3
290 T=11**1/8+(105-X)/250+RND(8)/9
300 DX=C08(T)*L:DY=SIN(T)*L
305 IFABS(DY)>10THENDX=10*SGN(DX)
306 IFABS(DY)>10THENDY=10*SGN(DY)
307 X=X+2*DX:Y=Y+2*DY:POKE53252.X:POKE53
10000 DATA 0,0,0,0,0,0,0,0,0,0
10001 DATA 0,0,0,0,0,0,0,0,0
10002 DATA 0,0,0,0,0,0,0,0
                        0.0.0.0.0.0.0.0.0.0
0.0.0.127.255.192.255,255,248
255.255.254.255.255.255.2
 10003
             DATA
 10004
             DATA
 10005
55.254
10006
             DATA
             DATA 255,255,248,127,255,192,0,0,0
 10007
  0008
             REM
                             RIGHT FLIPPER (DOWN)
 10009
             :
DATA 0,0,0,0,0,0,0,0,0
DATA 0,0,0,0,0,0,0,0
DATA 0,0,0,0,0,0,0,0
DATA 0,0,0,0,0,0,0,0
DATA 0,0,0,3,255,254,31,255,255
DATA 127,255,255,255,255,255,127,2
 10009
10010
10011
10012
10013
10014
 55,255
10016
10017
             DATA 31,255,255,3,255,254,0,0,0
 10018
            REM
                            LEFT FLIPPER (UP)
 10020
             DATA 0.0.0.0.0.0.0.1.240
             DATA 127,224,0,127,192,0,1224

DATA 0.63,240,0,15,240,0,31,224

DATA 0.63,224,0,127,192,0,255,192

DATA 1,255,128,3,255,0,7,254,0

DATA 15,252,0,31,248,0,63,240,0

DATA 127,224,0,127,192,0,127,128,0

DATA 63,0,0,30,0,0,0
 10021
10022
10023
 10024
10025
10026
 10027
 10028
                          RIGHT FLIPPER (UP)
             REM
             DATA 0,0,0,0,0,0,15,128,0
DATA 15,192,0,15,240,0,7,248,0
DATA 7,252,0,3,254,0,3,255,0
DATA 1,255,128,0,255,192,0,127,224
DATA 0,63,240,0,31,248,0,15,252
DATA 0,7,254,0,3,254,0,1,254
 10030
 īĕĕŝī
 10032
 10033
```

Yorick's Revenge **Program**

10035

```
10036
            DATA 0.0.252.0.0.120.0.0.0
10038
            REM
                        BALL
                      0.0.0.7.192.0.15.224.0
31.240.0.31.240.0.31.240.0
15.224.0.7.192.0.0.0.0
 10040
            DATA
 10041
            DATA
 10042
            DATA
            DATA
DATA
DATA
10043
 10044
                      0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0
 10045
 10046
            DATA
                      0.0.0.0.0.0.0.0.0
10047
                           100 POINTS
            REM
 10049
10050
           DATA 199,135,128,204,204,192,204,2
04,192
10051
            DATA 204,204,192,204,204,192,204,2
04,192
10052
            DATA
                      199,135,128,0,0,0,0,0,0
                      10053
            DATA
 10054
 10055
10056
            DATA
10057
 10058
            REM
                          500 POINTS
10059
                      63,30,30,48,51,51,48,51,51
62,51,51,3,51,51,3,51,51
62,30,30,0,0,0,0,0
0,0,0,0,0,0,0
10060
            DATA
            DATA
DATA
DATA
DATA
DATA
 10061
10062
10063
10064
 10065
                      0.0.0.0.0.0.0.0.0.0
            DATA
10067
 10068
            REM
                            SKULL
10069
10070
            DATA 0,126,0,3,255,192,14,126,112
DATA 15,60,240,31,153,248,31,255,2
10071
48
10072
                      63,62,124,62,28,60,126,28,62
127,62,126,63,255,252,31,201,
10073
            DATA
248
10074
10075
10076
10076
10078
                      15,201,240,7,255,224,6,219,96
6,0,96,7,108,224,3,255,192
1,255,128,0,255,0,0,126,0
            DATA
            REM
                             GUTTER
10079
 10080
            DATA
                      255,255,255,255,255,0,0,0
            DATA
DATA
DATA
DATA
DATA
                      0.0.0.0.0.0.0.0.0
 10081
10082
                      0.0.0.0.0.0.0.0.0
10084
10085
                      0,0,0,0,0,
                                          0.0.0.0
10086
                      ē,ē,ē,ē,ē,
            DATA
                                          0,0,0
 19998
            REM
                            SPRITE INITIALIZATION
19999
19999 :
20000 POKE53281,0:POKE53280,3:PRINT"D"
20010 V=53248:RESTORE:FORI=0T08:FORJ=0T0
62:READG:POKE12288+I*64+J,Q:NEXT:NEXT
20020 POKEV+39,7:POKEV+40,7:POKEV+41,10:
POKEV+42,14:POKEV+43,14:POKEV+44,8
20030 POKEV+45,1:POKEV+46,10
20040 POKE2040,192:POKE2041,193:POKE2042,196:POKE2043,197:POKE2044,197
20050 POKE2045,198:POKE2046,199:POKE2047
   200
20070 POKEV+29,255:POKEV+23,124:POKEV+16
. 0
20060
            POKEY, 120: POKEY+2, 200: POKEY+1, 210:
20080 POKEY,120:POKEY+2,200:POKEY+1,210:

POKEY+3,210

20090 POKEY+12,162:POKEY+13,140

20100 POKEY+14,161:POKEY+15,232

20110 POKEY+6,120:POKEY+8,215:POKEY+9,13

0:POKEY+7,130

20120 POKEY+10,158:POKEY+11,70

20125 FORI=YTOY+16STEP2:IFPEEK(I)>40THEN

POKEI,PEEK(I)-40

20127 NEXT

20130 RETURN

2097 :
20997
20998
            REM
                        BACKGROUND INITIALIZATION
20999
21000 POKE54273,RND(8)*10+50
21005 FORI=0TO24:FORJ=31TO39:POKE1024+I*
40+J,160:POKE55296+I*40+J,14:NEXT
21010 FORI=0TO24:FORJ=0TO39STEP39:POKE10
24+I*40+J,160:POKE55296+I*40+J,6:NEXT:NE
21000
21005
```

Yorick's Revenge Program

```
21025 POKE54273,RND(8)*10+30
21030 FORI=1TO23:FORJ=1TO30STEP29:POKE10
24+I*40+J,160:POKE55296+I*40+J,4:NEXT:NE
ΧŤ
XT
2104
21060
21070
21080
21090
21100
           21110
21120
21120
21130
21140
21150
21160
21170
21180
21190
21200
21200
21210
21220
21230
21900
21997
21999 : 20000 INITIALIZATION

22000 FORI=54272T054296:POKEI,0:NEXT

22010 POKE54296,15:POKE54277,96:POKE5427

8,240:POKE54276,17

22020 POKE54284,64:POKE54285,128:POKE542

83,0:POKE54280,15:POKE54281,233:POKE5428

2,233

22030 POKE54291,64:POVE5428
22030 POKE54291,64:POKE54292,128:POKE542
90,0:POKE54287,55:POKE54288,33:POKE54289
, <u>2</u>33
22040
            RETURN
22997
22998
           REM
                                     END OF GAME
 22999
           POKE54276.33: POKE54283.17: POKE5429
23000
0,65
23005
           FORI=1T0100STEP25
FORJ=100T01STEP-25
FORK=1T010
POKE54273, (I+J+K)AND255
POKE54280, (I+J-K)AND255
POKE54287, ABS(J-I)+K
NEXT:NEXT:NEXT
IFF=1THENRETURN
POKE53281, 10:POKE54296, 0
IFPEEK(197)=64THENPOKEY+45, RND(8)*
1023080
23010
23020
23030
23040
23050
23050
23060
23065
 23070
23080
 16:GOTO23080
23090
23997
            POKE53281,0:PRINT"3":GOTO 14
                                    PRINT TITLE
23998
            REM
23999
INC. ":FORI=1T05000:NEXT:RETURN
```

Yorick's Revenge Program

Space Crash

Charles Mott, Ir.

```
123
               REM
 10
11
                                      INITIALIZATION
12 .
30 PRINT"%C";:DIMA(40),B(40),C(40)
40 FS(1)=838:FS(2)=839:FS(3)=840:GA=15:A
G=240:LL=0:IN=1944:NA=5
50 V=53248:BR=53280:BK=53281:SC=1024:CL=
55296:SO=54272
60 PA$="!":PI$=""
        ) I Sign (1)
4
260 POKESA, (PEEK(SA)AND254)
270 POKE1,PEEK(1)AND251
280 FORI=0T0160:FORJ=0T07:POKEBA+I*8+J,PEEK(V+I*8+J);PEEK(V+I*8+J);PEEK(V+I*8+J);PEEK(1)OR4:POKESA,PEEK(SA)OR1
290 POKE1,PEEK(1)OR4:POKESA,PEEK(SA)OR1
300 POKEV+24, (PEEK(V+24)AND240)OR14
310 READA:IFA=-1THEN330
320 FORI=0T07:READB:POKEBA+A*S+I,B:NEXT:
GOTOSIO
GUTUS10
330 POKE967,255
331 FORT=1TO255STEP3:POKESO+4,16:POKESO+
4,17:POKEV+4,17:POKEV,T:GOSUB370:NEXT
332 POKEBK,4:PRINT"%CT"):POKEBK,0:U=163
334 GOSUB910:GOSUB680:FORI=1TONA:GOSUB78
0: NEXT
336
          GOTO 510
340
350
        REM
                                          SPIN SAUCER
300 KEN
360 :
360 :
370 POKESO+4,16:IFFGTHENPOKEFS(1),GA:POK
EFS(2),GA:POKEFS(3),GA:FG=0:POKESO+4,17:
RETURN
000 DOKESO+4 :17:POKFFS(1),AG:POKEFS(2),A
G:POKESO+4,17:POKEFS(1),AG:POKEFS(2),A
G:POKEFS(3),AG:FG=1:RETURN
 48ia
490 REM
                                          MAIN LOOP
500
500 :
510 GETA$:POKES0+4,16:POKESO+4,17:GOSUB3
70:I=I+1:IFI>NATHENI=1
520 IFA$="]"THENU=U-8:IFU<51THENU=51
530 IFA$="]"THENU=U+8:IFU>255THENU=255
540 IFA$="M"THENVE=VE+8:IFVE>255THENVE=2
```

Space Crash Program

IFA\$="#"THENVE=VE-8:IFVE<24THENVE=24 GOSUB850:GOSUB370:POKESO+1,I*2 IF<PEEK<V+31>AND1=1>THENGOSUB740 POKEV,VE:POKEV+1,U:GOTO510 550 550 550 570 640 640 SAUCER CRASH REM 650 : 660 POKESO+18,128:POKESO+18,129 665 FORT=UTO222STEP8:POKEV+39,10:POKEV+1 ,T:GOSUB709:POKEV+39,1:NEXT:POKEV+1,222 670 FORT=1TO15:POKEV+39,7:GOSUB709:POKEV +39,2:GOSUB370:NEXT:GOSUB710 680 RESTORE:FORT=0TO14:READA:POKE832+T,8 :NEXT .NEA | 690 POKE2040,13:POKEV,255:POKEV+1,155:PO KEV+39,1:POKEV+21,1:POKEV+16,0 700 VE=255:U=155:XG=PEK(V+31):RETURN 709 FORY=0T014:POKE832+Y,RND(1)*(255-1)+ 1:NEXT:RETURN 710 PRINTLEFT\$(YP\$,9+MO)LEFT\$(XP\$,34)OM\$:MO=MO+1:IFMO=STHENGOSUB680:GOTO1300 RETURN 711 712 SAUCER HIT ALIEN 720 REM 730 730 740 741 750 760 730 : 740 EK=0:FORHT=1TONA 741 IFU>B(HT)-16ANDU<B(HT)+16THENEK=HT 750 NEXT:IFEKTHENI=EK:GOTO770 760 GOSUB709:GOSUB660:RETURN 770 POKEA(I),33:XG=PEEK(V+31):FOKEA(I),3 2:GOSUB780:SR=SR+10:GOSUB775 773 POKESO+18,128:POKESO+18,129:POKEV+31 ,0:RETURN ,0:RETURN 775 PRINTLEFT\$(YP\$,5)LEFT\$(XP\$,33)"77"MID \$(STR\$(SR),2,6):RETURN *(818*(. 777 : 778 REM 779 : 780 RY= NEW ALIEN APPEARS RY=INT(RND(1)*(11)+0):RX=INT(RND(1)* RETURN **838**0 833 ALIEN DESCENDS 834 REM 840 850 POKEA(I),32:A(I)=A(I)+40:IFPEEK(A(I)><>32THEN880 \$55 AC=AC+1:IFAC>4THENAC=1 \$60 POKEA(I),WA(AC):B(I)=B(I)+8:RETURN \$80 IFPEEK(A(I))=160THEN885 \$83 POKEA(I),33:FORT=1T0100:NEXT:POKEA(I),32:GOSUB780:RETURN \$85 BL=BL-1:POKEIN+BL,38:GOSUB1105:IFBL= 0THEN1300 900 0(1)-0/11-40:COTOCOC ><>szTHEN880 A(I)=A(I)-40:GOTO883 898 89<u>9</u> PRINT GAME SCREEN 900 1030 GT#="**R_i** L_**i_i**" _1 _____<=> PRINTGR#:PRINTGT# PRINT"#61 1040 1050 1060 PRINT" #53 1070 PRINT" #13 PRINTLEFT\$(YP\$,3)LEFT\$(XP\$,33)"MSCO 1080 PRINTLEFT\$(YP\$,5)LEFT\$(XP\$,33)"00 1090 FORY=1TO5:PRINTLEFT\$(YP\$,8+Y)LEFT\$(XP\$,34)"=(=>":NEXT 1095 PRINTLEFT\$(YP\$,17)LEFT\$(XP\$,33)"mBL OCKS" OUCKS" 1100 PRINTLEFT\$(YP\$,18)LEFT\$(XP\$,33)"M L EFT:" 1105 PRINTLEFT\$(YP\$,20)LEFT\$(XP\$,34)" 1107 PRINTLEFT\$(YP\$,20)LEFT\$(XP\$,34)"氚"B 1120 1130 1140 1150 RETURN

PRINT TITLE SCREEN

REM

Space Crash **Program**

2011 124 1240 PRINT" E, INC. 麗胡 **===** <C> 1240 FRIM. E: INC. 253 1250 PRINT" 31 "; 1984 FAMFARE HOUS 1260 PRINT" ■ BY CHARLES MOTT, JR. 1270 1280 84" POKESC+999,160:POKECL+999,6 RETURN 1290 1291 REM END OF GAME 1300 FORT⇔VETO0STEP-1:POKEV,7:GOSUB370:P 1300 FORT-VETO05/EF-1.FOREV,1:005083/0:P OKEV+1,U:U=U+.19:NEXT 1304 PRINTLEFT\$(YP\$,7)LEFT\$(XP\$,10)"**=3** G AME_ OVER " 1320 : 1330 DATA 0,255,0,255,255,240,240,24 0,255,255,255,0,255,0 1350 : 1360 REM CHARACTER DATA 1370 : 1370 : 138<u>0</u> DATA 000,255,255,255,255,255,25 1500 DATA **03**0,046,127,239,255,063,247,19 1,239 1510 DATA 031,248,248,173,254,110,254,21 4,124 1520 DATA 033,000,102,060,255,060,102,00 0. 00o 1530 DATA 035,129,090,126,255,201,255,10 2,195 1540 DATA 036,036,090,126,255,201,255,10 DATA 036,036,090,126,255,201,255,10 ຂໍ້, 060 DATA 038,054,008,127,093,042,020,02 0.054 1570 5,102 DATA 043,198,040,016,056,255,153,25 1580 DATA 047,000,000,016,056,255,219,25 5.102 1640 DATA 060,015,031,051,115,063,031,01 5,000 1650 DATA 061,255,255,051,051,255,255,25 5,000 DATA 062,240,248,060,062,252,248,24 1660 ଡ଼ି ଡ଼ିଡ଼ିଡ 1670 DATA .-1

Space Crash Program



The Blobbis

Frank Mikulastik

```
REM
 12345678
       REM
REM
REM
       REM
                                                INITIALIZATION
8 :
10 POKE52,160:POKE56,160:SM=32768:CM=552
96:V=53248:S1=54272:M1=1016+SM
19 PRINT"M":GOSUB5300:GOSUB6000
20 POKE56578,PEEK(56578)OR3
22 POKE56576,(PEEK(56576)AND252)OR1
24 POKE53272,(PEEK(53272)AND15)OR0
26 POKE648,128
27 F9=0:GOSUB5000:F9=1
100 FORI=20*64T053*64STEP64
102 GOSUB6010
104 FORJ=0T063:READA:POKEI+J+SM,A:NEXT
106 NEXT
104 FORJ=0T063:READA:POKEI+J+SM,A:NEXT
106 NEXT
110 POKEV+21,7:POKEM1,16:POKEM1+1,20:POK
EV+28,255
120 POKEV+29,3:POKEV+23,3:POKEV+37,1:POK
EV+38,10:POKEV+39,6:POKEV+40,11
125 POKEV+41,4
140 DIM S1(2,20),S2(20),S3(20),S4(20)
150 S1(1,1)=49:S1(1,2)=50
155 S1(2,1)=47:S1(2,2)=48
157 B1=2
160 S2(1)=20:S2(2)=21:S2(3)=22:S2(4)=23
165 B2=4
 165
 170 52-4
170 53(1)=27:S3(2)=28:S3(3)=29:S3(4)=28:
53(5)=27:S3(6)=30:S3(7)=31:S3(8)=30
$3(5)=27:$3(6)=30:$3(7)=31:$3(8)=30
175 B3=8
180 $4(1)=37:$4(2)=38:$4(3)=39:$4(4)=40:
$4(5)=39:$4(6)=38
185 B4=6
190 D2(1)=20:D2(2)=24:D2(3)=25:D2(4)=26:
D2(5)=25:D2(6)=24
195 C2=6
200 D3(1)=27:D3(2)=28:D3(3)=29:D3(4)=28:
D3(5)=27:D3(6)=30:D3(7)=31:D3(8)=30
210 D4(1)=37:D4(2)=41:D4(3)=42:D4(4)=43:
D4(5)=40:D4(6)=44:D4(7)=45:D4(8)=46
215 C4=8
220 $5(1)=53:$5(2)=52:$5(3)=51:$5(4)=52
290 GOSUB5000
400 GOSUB5100
491
492
500
              REM
                                        MAIN LOOP - NOT SHOOTING
              GOSUB1000
502
               ONWSGOSUB1000,4000,4100,4200
 505
              K=K+1

IFF=2THENGOSUB6400:GOSUB5200:GOTO400

IFF=1THENGOSUB6030:GOSUB5200:GOTO400

GOSUB2100

K=K+1:IFK>5THENK=0:GOTO590

GOSUB1000
506
507
510
515
               ÕNWSGOSUB1000,4000,4100,4200
```

555 556 560 590 690 IFF=1THENGOSUB6030:GOSUB5200:GOTO400 IFF=2THENGOSUB6400:GOSUB5200:GOTO400 GQSUB2000 GÓT0500 INITIALIZE SPRITES 691 REM 691 6920 7190 7200 7200 7200 . RX=0:POKEV+16,0 POKEV,100:POKEV+1,200:POKEV+2,0:POKE X=100:D=1:N=1 BX=150:BD=1:BS=20 F=0:POKEV+31,0 798 795 796 797 RETURN BUILD FORCE FIELD REM : PRINT"D":POKE53281,0:POKE53280,5 P=240:T1=160 FORI=PTOP-1+5*40 GOSUB6040 POKESM+I,160:POKECM+I,2 800 805 810 830 85ē NEXT 860 861 862 863 865 REM BUILD FLOOR 890 K 990 : 991 R 992 : 1000 PLAYER MOVEMENT DRIVER REM JV=PEEK(56320) JY=FEER(3520) IFJV=127THEN1090 FR=JVAND16 JV=15-(JVAND15) IFFR<>16THENGOSUB6020:GOTO3000 IFJV=40RJV=50RJV=6THEND=1:T=X-9:GOT 1001 1010 1020 1030 1040 01100 1050 I TO1100 1070 N 1090 R IFJV=80RJV=90RJV=10THEND=2:T=X+9:G0 N1=1:POKEM1,S1(D,H1) RETURN 1091 1092 1093 1100 1110 MOVE PLAYER REM . 1=N1+1:IFN1>B1THENN1=1 IF(T)308)OR(T<12)THENT=X:N1=1 X=T:RX=(RXAND254):IFT>255THENT=T-25 5 RX: <RXOR1> POKEM1, S1(D, N1):POKEV+16,RX:POKEV,T GOSUB6050 1140 1190 1990 RETURN MOVE ROW TO THE RIGHT 1991 1992 REM GOSUB6010 FORI=P+K*40~1TOP+(K-1)*40STEP-1 2000 2005 T2=PEEK(SM+I) POKESM+I,T1:T1=T2 NEXT 2010 2030 2040 2040 2090 2091 2092 2093 2105 POKES1+4,0:RETURN MOVE ROW TO THE LEFT REM . GOSUB6010 FORI=P+(K-1)*40TOP+K*40-1 T2=PEEK(SM+I) POKESM+I,T1:T1=T2 2110 2130 2140 2150 2190 NEXT POKES1+4,0 RETURN 2990 2991 2992 REM MAIN LOOP - SHOOTING 3000 T=1783+INT((X+4)/8):POKET+54272,4:P OKET,81:S3=70 3001 BL=BL+1:PRINT"**%%%%%%%%%%%%%%** MANAGEBEEEPEEPEE"; BL; 3005 Q=PEEK(V+31):IF(QAND2)=2THENF=1:GOT osese. 03090 3006 IFF>0THEN3090 3010 POKET+SM-1024,32:FOKET+54272,2:T=T-40:IFT<1024THEN3080 3020 IFPEEK<T+SM-1024>=160THEN3080 3030 S3=S3+1:GOSUB6060 3040 POKET+54272,4:POKET+SM-1024,81 3043 GOSUB 6010 3044 FORIZ=1T0100:NEXT 3050 ONWSGOSUB1000,4000,4100,4200 3070 GOTD3005

```
POKET+SM-1024,32
RETURN
 3080
 3090
 3091
 3092
            REM
                               MOVE HUNCHBACK BLOBBIS
            IF(P2>150)AND(P2<157)THENGOSUB6300
IF(F1=1)AND(F1=2)THENGOTO4090
P2=P2+6:IFP2>320THENP2=0:WS=3:POKEV
 4000
 4005
 4010
4040
P2
4050
            N2=N2+1:IFN2>B2THENN2=1
POKEM1+1;S2(N2)
RETURN
 4060
 4091
4092
4093
            REM
                               MOVE BUG-EYED BLOBBIS
4093 :
4100 IF(P3>150)AND(P3<157)THENGOSUB6100
4120 P3=P3+6:IFP3>320THENP3=0:WS=4:POKEV
+40,7:GOTO4200
4130 IFP3>255THENRX=(RXOR2):POKEV+16,RX:
POKEV+2,P3-255:GOTO4150
4140 RX=RXAND(255-2):POKEV+16,RX:POKEV+2
 , ĒЗ
 415ø
            N3=N3+1: IFN3>B3THENN3=1
POKEM1+1, S3(N3)
 4160
 4190
            RETURN
#193 : #193 : #200 HND(P4<157)THENGOSUB6200 #200 IF(P4>150)AND(P4<157)THENGOSUB6200 #210 P4=P4+6:IFP4>320THENP4=0:WS=2:POKEV+40,11:POKE53281,0:POKE53280,5:GOTO4000 #230 IFP4>255THENRX=(RXOR2):POKEV+16,RX:POKEV+2,P4-255:GOTO4250 #240 RX=RXAND(255-2):POKEV+16,RX:POKEV+2,P4 #250 M4=N4+1:IFN4>B4THENN4=1 #260 POKEM1+1,S4(N4):POKEM1+1 #260,N4+1
4191
4192
4193
4200
280 N4+1
4290 RETURN
4991
4992
4993
            REM
                                 PRINT TITLE SCREEN
           ; IFF9=1THEN5039
PRINT" = 110000": PO
GOSUB60000
PRINT" = 30 = 110000"
5000
5005
                                      :POKE53281,5:POKE53280,0
5010
5011
5012
5013
                                                                                     PR **
                                               200
                                                             200
                                                    ä
                                                                           ...
                                                                  ₩..
                                                                       23
                                                                           <u>...</u>
5014
5015
5016
5017
5018
                                                                       31
31
            P<u>RĪ</u>NŤ"
                                                3
                                                              37
                                                                                -
                                     23
5019
                                 PRINT"
                                   21
                                                              3
                                                                        ₹:3
                                                                             5020
                                      29
                                                                  3 💻
                                                                                3
           PRINT"
                                               의 !
의 !
작의
                                                             調調
5021
                             2
         29
                                 28
          PRINT"
PRINT"
SIES SIES
5022
                             31
                                 23
                                                                  3
                                 158
                                               3
                                                                   72 E
5023
                             쿒
                                               3
                                                    38
                                                                  28
                                                                                 29
                                                                                          PEINT"
         29
                                                   3
5024
                            58
                                               3
                                PRINT"
5025
                            28
                                        929
           5026
TIK"
5027
                                                               FRANK MIKULAS
5050
           5090
ĒV+3
5091
            RETURN
```

5092

5093 REM 5094 : PRINT INSTRUCTION SCREEN 5100 PRINT"Dm";:POKE53281,12:POKE53280,6 POKEV+1,0:POKEV+3,0 PRINT"3 5105 5110 PRINT" # HOW MANY SHOTS WILL IT TAKE YOU TO 5115 PRINT" # BREAK THROUGH THE MOVING FO RCE_FIELD___"; 5120 PRIÑT"# ÁND HIT ONE OF THE BLOBBIS ? 5125 PRINT"# 5130 PRINT" # THE HUNCHBACK BLOBBIS CAN K ILL 5135 JISD MRINT" WHITH HIS PULSAR. THE BUG-EY ED BLOBBIS ";
5140 PRINT" WHILL REPAIR THE TOP ROWS OF THE FORCE ";
5145 PRINT" WHITE FLASHING BLOBBIS WILL TRY ";
5150 PRINT" -- -5150 PRINT" # TO CONFUSE YOU. 5151 PRINT"# 5155 PRINT"# PRESS THE FIRE BUTTON T START O START ";
5160 POKEM1,S1(2,1):POKEV,150:POKEV+1,20
8:N=16:WS=2:POKEV+2,0:POKEV+3,0
5162 N=N+1:IFN>B1THENN=1
5163 GOSUB6050:FORS3=1TO200:NEXT
5164 POKEM1,S1(2,N):IF(PEEK(56320)AND16)
=16THEN5162
5170 POKEV+1,0
5175 GOSUB6000
5180 RETURN
5180 RETURN
5193:
GAME OVER
5193:
5200 POKE53281,15:POKE53280,0 5200 5210 5220 G0SUB6080 IF(PEEK(56320)AND16)=16THENGOT05280 POKEV+39,6 5280 5285 5290 RETURN 5300 5310 REM INITIALIZE SOUND 5320 3320 5330 FORI=54272T054296:POKEI,0:NEXT 5340 POKE54296,15:POKE54277,96:POKE54278 ,240:POKE54276,33 5350 POKE54284,64:POKE54285,128:POKE5428 3,0:POKE54280,15:POKE54281,233:POKE54282 5992 SCREEN CHANGE SOUND 5993 6000 POKES1+24,15:POKES1+5,15:POKES1+6,1 5:POKES1+4,0:POKES1+4,33 6002 FORT=1T0200:POKES1,I:POKES1+1,I:NEX 6003 POKES1+4,0 RETURN 6007 BLOBBIS MOVEMENT SOUND 6008 REM 50007 : 6010 POKES1+4,0:POKES1+24,15:POKES1+5,15 :POKES1+6,15:POKES1+4,17 6012 S2=INT(RND(0)*10+10):POKES1,S2:POKE S1+1,S2 6013 RETURN 6017 : 6018 6019 6020 REM INITIAL BULLET SOUND 6020 FORS2=15TO0STEP-1 6022 POKES1+24,15:POKES1+4,0:POKES1+4,12 9:POKES1+5,15:POKES1+6,15:POKES1,S2*10 6023 POKES1+1,S2*10 6024 6025 6026 6027 6028 REM A BLOBBIS IS HIT 6029 6030 POKES1+24,15:POKES1+5,15:POKES1+6,1 5:POKES1+4,0:POKES1+4,129 6031 FORS2=15TO0STEP-1 6032 S3=S2*10:POKEV+40,S2:POKES1,S3:POKE \$1+1,53 6033 FORS4=1T0100:NEXT

```
6034 NEXT
6035 POKES1+4,0
6036 RETURN
6037 :
    6038 REM
                                                                                                               FORCE FIELD SOUND
    6039
      5039 :
6040 POKES1+24,15:POKES1+5,15:POKES1+6,1
5:POKES1+4,0:POKES1+4,129
6042 POKES1,I-P:POKES1+1,I-P
6044 FORS2=1TO30:NEXT
6045 POKES1+4,0
6046 RETURN
   6040
   6042
6044
   6045
   6046
   6047
6048
                                        REM
                                                                                                       PLAYER WALKING SOUND
   6049
6050
   6050 POKES1+24,15:POKES1+12,15:POKES1+13
,15:POKES1+11,0:POKES1+11,33
6052 POKES1+7,82:POKES1+8,82
6055 FORZ=1T010:NEXT:POKES1+11,0
   6056
6057
                                        RETURN
   6058
                                       REM
                                                                                                      MOVING BULLET SOUND
   6059
     6003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .0003 .
   6060
   6062
    6064
   6068
   6069
                                        RETURN
   6077
6078
6079
6080
                                    REM
                                                                                                                      GAME OVER SOUND
          079 :
080 POKES1+24,15:POKES1+12,15:POKES1+13
15:POKES1+11,0:POKES1+11,17
082 FORS2=180TO1STEP-4:FORS3=82TOS2+10
083 POKES1+7,S3:POKES1+8,S3
084 NEXT:NEXT
086 POKES1+11,0
   6083
   6084
   6086
   6089
6097
                                        RETURN
   6098
                                        REM
                                                                                                                     BUG-EYE ACTION
   6099
                                       FORI=1TO5
FORJ=1TOC3
GOSUB6010:POKEM1+1,D3<J>
PRINT"##:300000000
   6100
   6104
6106
  6107 PRINT" PERSONNELS NO.
  6111 PRINT" AND ADDRESS OF THE PRINT OF THE 
  6112 NEXT
6114 NEXT
6119 RETU
6197
                                       RETURN
   6198
                                       REM
                                                                                                                    FLASHER ACTION
   6199
   6200
                                       FORI=1T05
FORJ=1T0C4
GOSUB6010:POKEM1+1,D4(J)
POKE53280,6+J:POKE53281,7+J
 6200
6210
6225
6225
6250
6250
6250
6250
                                        NEXT
                                        POKE53281.0:POKE53280.5
                                        RETURN
  6298
                                       REM
6298 REM HUNCHBACK ACTION
6299:
6300 N5=1:POKEM1+2,S5(N5):POKEV+4,P2:P5=
53:POKEV+5,P5:POKES1+11,0:POKES1+4,0
6320 FORI=1TO100:NEXT
6321 POKES1+24,15:POKES1+12,15:POKES1+13,15:POKES1+11,0:POKES1+11,0:POKES1+11,0:POKES1+11,0:POKES1+11,0:POKES1+11,0:POKES1+11,0:POKES1+11,0:POKES1+11,0:POKES1+11,0:POKES1+11,0:POKES1+11,0:POKES1+11,0:POKES1+11,0:POKES1+11,0:POKES1+0:POKEM1+1:IFN6>C2THENN6=1
6322 N5=N5+1:IFN6>C2THENN6=1
6330 P5=P5+6:POKEV+5,P5:POKEM1+2,S5(N5):POKEM1+1,102(N6)
                                                                                                            HUNCHBACK ACTION
6330 P5=P5+6:POKEV+5,P5:POKEM1+2,S5(N5):
POKEM1+1,D2(N6)
6340 IF(PEEK(V+30)AND5)=5THENF=2:POKES1+
11,0:GOTO6390
6350 IFP5>250THENGOTO6380
6360 GOTO6320
6380 POKEV+5,0:POKEV+4,0
6385 POKES1+11,0
6385 POKES1+11,0
  639<u>0</u>
                                       RETURN
  6397
 6398
6399
6400
                                   REM
                                                                                                                   PULSAR HITS PLAYER
  6400 FORI=0T015
6410 POKEV+39,I:POKES1+5,15:POKES1+6,15:
POKES1+4,0:POKES1+4,33
```

```
6420
6430
6440
6450
                                     POKES1+1,<16-I):POKES1,<16-I)
FORJ=1TOI*10:NEXT
NEXT
POKEV+4,0:POKEV+5,0:POKES1+4,0
                                       RÉTÜRH
  7000
                                                                                                                             SPRITE DATA
 7001
                                      REM
7002
7545
7550
                                                                            HUNCHBACK BLOBBIS WALKS - 0,40,00
170,0,2,170,128
10,255,160,11,255
224,43,125,224,171
190,224,171,255,232
170,195,170,170,10
170,250,170,170,10
170,170,160,170,170
160,10,170,160,10
170,160,2,130,128
2,130,128,10,130
160,10,130,160,0
HUNCHBACK BLOBBIS WALKS - 3
                                      DATA
DATA
DATA
 7560
 757ē
7580
7590
                                       DATA
DATA
DATA
 7600
7620
7630
                                       DATH
                                       DATA
 7640
7650
                                       DATA
                                       DATA
                                       DATA
DATA
REM
 7660
7670
7675
                                       DATA
DATA
DATA
  768ĕ
                                                                             0,0,0,8,0

0,42,0,0,170

128,2,25,160,11

255,224,11,125,232

43,190,234,171,255

234,170,195,170,170

190,170,170,170,175

250,170,160,10,170

170,160,10,170,160

10,2,128,10,2

160,42,2,160,42

0,0
 7690
7690
7700
7710
7720
7730
7750
7750
                                       DATA
DATA
DATA
DATA
                                       DATA
DATA
DATA
 7770
7780
7790
                                       DATA
DATA
DATA
                                                                             160,42,2,160,42
0.0,0
0.0,0
HUNCHBACK BLOBBIS WALKS - 3
0.0,0
0.0,0,0
0.0,10,0,0
42,128,0,170,168
2,255,170,11,255
234,11,125,234,43
190,234,171,255,239
170,195,160,170,190
160,170,170,160,250
170,160,10,170,160
10,170,160,2,128
168,2,128,168,10
  7800
 7810
7815
                                        DATA
                                        REM
                                       DATA
DATA
  7820
  7830
  7840
                                        DATA
                                       7850
  <u>z</u>ള്ള്
  7870
  7890
  7900
  7910
7920
7930
                                                                            128,0,128,0

0

HUNCHBACK BLOBBIS WALKS - 4

0,0,0,0,0

0,0,0,0,0

128,0,2,170,128

10,255,160,43,255

232,171,125,234,171

190,234,171,255,234

170,195,170,170,190

175,250,170,160,10

170,160,10,170,160

10,170,160,10,170

160,2,128,160,2

128,160,10,128,168

10,128,160,10,128,168

10,128,160,0

HUNCHBACK BLOBBIS DANCES - 0

0,0,170,0,2

170,128,170,125,175

170,128,170,195,175

170,195,175,250,190

170,160,10,170,160

10,170,160,10,170,160

10,170,160,10,170,160

10,170,160,10,170,160

10,170,160,10,170

160,10,170,160

0,0

HUNCHBACK BLOBBIS DANCES - 0

10,130,160,10,128

10,0

HUNCHBACK BLOBBIS DANCES - 0

10,130,160,10,128

10,0

HUNCHBACK BLOBBIS DANCES - 0

10,0

10,0

HUNCHBACK BLOBBIS DANCES - 0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0

10,0
  7940
7950
7955
                                        REM
DATA
DATA
  7960
   7970
                                       DATA
DATA
DATA
DATA
   7980
   799A
  8000
   ēēī ē
                                        DATA
DATA
DATA
  8020
8030
  8040
                                         DATA
DATA
DATA
  8060
8070
    ē080
                                         REM
DATA
DATA
   2025
  8090
 8090
8100
8120
8120
8140
8150
8160
8170
                                         DATA
DATA
DATA
                                         DATA
DATA
DATA
DATA
DATA
  8180
8190
                                         DATA
DATA
DATA
DATA
   8200
  8210
8220
                                                                               0,0

HUNCHBACK BLOBBIS DANCES -

0,40,0,0

170,0,2,170,128

10,255,160,11,255

224,43,125,224,171

190,224,171,255,232

171,0,234,171.0

234,251,0,234,10

255,170,10,170,175

10,170,160,10,170

160,10,170,160,10

170,160,2,130,128

2,130,128,10,130
  8225
8230
                                          REM
DATA
DATA
  8230
8240
8250
8260
8270
8280
                                          DATA
DATA
DATA
                                         DATA
DATA
DATA
DATA
DATA
    8290
    8300
   8310
8320
             330
```

```
835Ø
8355
                    DATA 160,10,130,160,0
REM HUNCHBACK BLOBBIS DANCES - 3
 8360
8370
8380
                                       0

40,0,0,170,0

2.170,128,10,255

160,11,255,224,43

125,224,251,190,224

251,255,232,251,0

234,171,0,234,11

0,234,11,0,234

10,255,175,10,170

160,10,170,160,10

170,160,10,170,160

2,130,128,2,130

160,10,130,160,10

128,0,0

BUG-EYED BLOBBIS WALKS - 1
                    DATA
                    DATA
 8390
                    DATA
                    DATA
DATA
DATA
DATA
DATA
 8400
 8410
8420
8430
8440
8450
                    DATA
DATA
DATA
DATA
DATA
DATA
REM
 8469
8479
8489
 8499
 8500
8510
8520
                    DATA
                   65,64,0,65,
0,65,0,0,65
0,0,65,0,3
 8530
8540
8550
                                       0.0,65,0,3
255,192,170,190,170
149,190,86,149,190
86,128,190,2,128
190,2,170,190,170
3,255,192,3,0
192,3,255,192,0
40,0,40,0
10,170,160,10,170
160,10,0,40,0
BUG-EYED BLOBBIS WALKS - 2
0,164
 8560
 8570
8580
8590
 8600
 8610
8620
8630
8635
8640
8650
                   DATA
REM
                                       BUG-EYED BLOBBIS WI
0/1,64
0/1,6,4,1
0/1,65,0,0
65,0,3,255,192
3,255,192,42,255
168,38,255,152,34
255,136,42,255,168
3,255,192,3,255
192,3,0,192,3
0,192,131,255,192
128,40,0,170,170,0
0,160,0,0,168
                    DATA
DATA
DATA
DATA
                    DATA
DATA
DATA
                   8750
8760
8779
8775
8780
8780
                                       BUG-EYED BLOBBIS WALKS - 3
0,1,64,0,1
0,0,1,0,85
55 0,54 55,0
                    REM
                    DATA
DATA
                                      0.1.64.0.1
0.0.1.0.85
65.0.64.65.0
3.255.192.170.190
170.149.190.86.149
190.86.128.190.2
128.190.2.170.190
170.3.255.192.35
0.192.35.255.192
40.40.0.42.40
0.10.170.160.2
170.168.0
BUG-EYED BLOBBIS WALKS - 4
0.0.64.5.0
64.15.0.64.64
0.65.0.3.255.192.42
255.168.38.255.152
34.255.192.3.255
192.3.255
192.3.255
192.3.255
192.3.255
194.0.40.2.10
170.170.10.170.170
10.0.0.0.0
BUG-EYED BLOBBIS WALKS - 5
 8800
                    DATA
8810
8820
                    DATA
 8830
                    DATA
                   8840
8850
8860
8870
8890
8900
8905
                   DATA
8910
8920
                   DATA
DATA
DATA
DATA
8930
8940
8950
8960
8970
                    DATA
8980
                    DATA
                    DATA
9000
                    DATA
9010
9020
                   DATA
 9030
                    DATA
                                      10,0,0,42,0
0.0
BUG-EYED BLOBBIS WALKS - 5
1,64,0,0
64,0,65,85,0,65
1,3,255,192,170
190,170,149,190,86
149,190,86,128,190
2,128,190,2,170
190,170,3,255,192
3,0,200,3,255
200,0,40,40,0
40,168,10,170,160
10,170,128,10,0
0,42,0,0
BUG-EYED BLOBBIS DANCES - 1
9040
                    DATA
                   REM
DATA
DATA
9050
9060
                   DATA
DATA
DATA
9070
9080
9090
9100
9110
9120
                   DATA
                   DATA
2120
9130
9140
9150
9160
9170
                    DATA
                   DATA
DATA
DATA
                   DATA
REM
                   DATA
9180
9190
                                       65,64,0,65,0
9200
                                      0,65,0,0,65
                    DATA
```

255,192,3,254,170
36,254,86,170,190
36,149,190,2,149
190,2,128,190,170
128,191,170,128
192,3,255,192,128
40,2,128,40,2
170,170,170,170
170,0,0,0
BUG-EYED BLOBBIS DANCES - 2
1,65,64
0,65,64
0,65,64
0,65,0,0,65
0,0,3,255,192
3,255,192,3,255
192,170,190,170,149
190,86,149,190,86
128,190,2,128,190
2,170,130,170,3
255,192,0,40,0
0,40,0,10,170
160,10,170,160,10
0,160,42,0,168
0
BUG-EYED BLOBBIS DANCES - 3 9220 9230 9240 9250 DATA DATA DATA DATA 9260 9270 DATA 9280 9290 DATA 9300 DATA 9310 9315 REM DATA 9320 9330 DATA DATA DATA DATA 9340 9350 9360 9370 9380 DATA DATA DATA DATA DATA 9390 9400 9410 9420 9430 9440 9450 DATA DATA DATA REM BUG-EYED BLOBBIS DANCES - 3
0,0,0,0
0,1,65,64,0
65,0,0,65,0
0,65,0,0,65
0,3,255,192,170
190,170,149,190,86
149,190,86,128,190
2,128,190,2,170
190,170,3,255,192
131,0,194,131,255
194,170,170,170,170
170,170,0,0,0
BUG-EYED BLOBBIS DANCES - 4
0,0 9455 9460 9470 DATA DATA DATA DATA DATA DATA 9480 9490 9500 9510 9520 DATA DATA DATA 9530 9540 9550 9560 9570 DATA 958Ø DATA
DATA
DATA
DATA
DATA 9585 9590 0,0,0,0,0 0,0 9600 9610 9620 9630 9640 9650 DATA DATA 9660 9670 DATA DATA DATA 9680 DATA DATA DATA DATA REM REM DATA DATA DATA DATA DATA DATA DATA DATA DATA 9800 9810 9820 9830 DATA 9840 9850 DATA DATA 9995 REM DATA 10000 10010 10020 10040 10060 ĕĕsĕ 10090 10090 10100 10110 10120 10130 REM FLASHING BLOBBIS WF 0.0,0 0,0,42,170 168,42,170,168,42,170,168 41,85,104,43,215 232,40,20,40,40 41,85,104,41 85,104,42,170,168 10140 10150 10160 10190 10200 10210 10220 DATA

10240 10250 10260 10270 10275 DATA DATA DATA 42,170,168,42,170 168,42,170,168,1 65,80,5,64,0 0.00.00.00.00
FLASHING BLOBBIS WALKS - 3
0.0.0.00
0.0.00.00
0.0.10.170.160
10.170.160.9.85
96.11.215.224,8
20.32.24,20.36
89.85.101.89.0
101.9.0.96.9
0.96.9.85.96
101.70.160.10.170
160.170.160.10.170
160.165.64.1
65.64.1.65.64
5.65.80.0
FLASHING BLOBBIS WALKS - 4
0.0 DATA Ø REM 10250 10290 10300 DATA DATA DATA DATA DATA DATA 10310 10310 10320 10330 10340 10350 10360 10370 10380 10390 10400 10405 DATA REM DATA DATA DATA DATA DATA DATA FLASHING BLOBBIS WALKS - 4
0.0,0,0,0
0.0,0,0,0
0.0,0,0,0
0.1,85,64,3
215,192,0,20,0
16,20,4,85,85
83,85,0,85,1
0,64,1,85
64,1,65,64
1,0,64,1,85
64,1,65,64,1
65,64,0,1
80,0
FLASHING BLOBBIS SPINS - 1
0,0,0,0
FLASHING BLOBBIS SPINS - 1
0,0,0,0,42,170,169,41
197,105,41,196,105
41,196,105,41,196,105
41,196,105,41,196,105
41,196,105,41,197
105,42,170,169,41
197,105,41,196,105
41,196,105,41,197
105,42,170,169,41
197,105,41,196,105
41,196,105,41,197
105,42,170,169,41
104,41,196,105
41,196,105,41,197
105,42,170,168
42,170,168,0,0
FLASHING BLOBBIS SPINS - 2
55,80,1,65,64 10405 10410 10420 10430 10440 10460 10470 10480 10490 10500 DATA DATA DATA DATA 10520 10530 10540 10545 10550 REM DATA 10560 DATA DATA DATA DATA 10580 10590 DATA DATA DATA 10600 10610 10610 10620 10630 10640 DATA
DATA
DATA
DATA
DATA
DATA
REM
DATA 10650 10660 10670 10675 10680 10680 10670 106710 106710 106710 106710 106710 106710 106710 106810 DATA DATA DATA DATA DATA DATA DATA DATA DATA 10600 10810 10815 REM REM
DATA
DATA
DATA
DATA
DATA 10820 10830 10840 10850 10860 10870 10880 10890 10900 10910 10920 10930 10940 10950 90.00,00,00 FLASHING BLOBBIS SPINS - 4 0,20,0,0,20 0,0,84,0,0 20,0,10,170,160 10,170,161,9,197 101,9,196,101,9 196,101,9,84,96 9,84,96,9,84 96,9,196,101,9 196,101,9,197,101 10,170,161,10,170 160,0,20,0,0 84,0,0,20,0 6,20,0,0 FLASHING BLOBBIS SPINS - 5 5,65 10955 10960 10970 10980 10990 11000 îioio 11020 11030 11040 11050 11060 11070 11090 11100 DATA 5,65 80,1,65,64,42 DATA

```
170,168,42,170,168
42,170,168,42,170
168,41,85,104,41
0,104,41,0,104
41,85,104,41,85
104,41,85,104,40
20,40,43,215,232
41,85,104,42,170
168,42,170,168,42
170,168,42,170,168
11110
11120
11130
11140
11150
                                                                                                                                                                                         DATA
DATA
DATA
                                                                                                                                                                                      11150
11160
11170
11170
111200
11220
11223
11230
                                                                                                                                                                                                                                                                                                                                              0,0,0,0,0

0,0

FLASHING BLOBBIS SP

170,170,170,170

170,170,170,170,170

170,170,170,170,169

83,106,169,19,106

169,19,106,169,21

106,169,21,106,169

21,106,169,19,106

169,19,106,169,83

106,170,170,170,170

170,170,170,170,170

170,170,170,170,170
                                                                                                                                                                                            DATA
                                                                                                                                                                                         DATA
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   SPINS -
   1123450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
20450
2045
                                                                                                                                                                                      REM
DATA
DATA
DATA
DATA
DATA
DATA
DATA
                                                                                                                                                                                            DATA
                                                                                                                                                                                         DATA
DATA
DATA
                                                                                                                                                                                         DATA
DATA
REM
   REM
DATA
DATA
DATA
DATA
DATA
DATA
DATA
                                                                                                                                                                                                                                                                                                                                                  0.170,0,0,170,0
0,170,0,0,85
0,170,0,0,85
0,10,170,160,0
243,0,0,255,192
0,255,0,0,60
0,0,42,0,0
42,128,0,42,128
0,42,128,0,170
128,0,154,128,0
166,0,0,170,0
0,42,0,0,42
0,42,0,0,42
0,0,21,64,0
21,64,0
PLAYER WALKS RIGHT ~
0,170,0
                                                                                                                                                                                                                                                                                                                                                           ø
                                                                                                                                                                                      11495
                                                                                                                                                                                                                                                                                                                                              PLAYER WALKS RIGH 0,170,0 0,170,0 0,170,0 0,0 170 0,0 170 0,0 155,192 0,255,0,0 60 0,0 42,128,0 42,128 0,42,128,0 165,0,0 170,120,0 16,1,74,80,0 81,80,0 $1,80,0,$1,64
                                                                                                                                                                                  REM
DATTA
11495
111552349
111552349
11155789
11155789
1115999
   81.80.0.81.64

Ø FLAYER WALKS LEFT
Ø.0.170.0.0.170

Ø 5.0.10.170.160

Ø 5.0.10.170.55

Ø 0.0.255.0.0

Ø 0.0.255.0.0

Ø 0.0.255.0.0

Ø 0.168.0.2

170.0.168.0

0.154.0.0

1.68.0.1

1.68.0.1

1.70.0

1.68.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

1.70.0

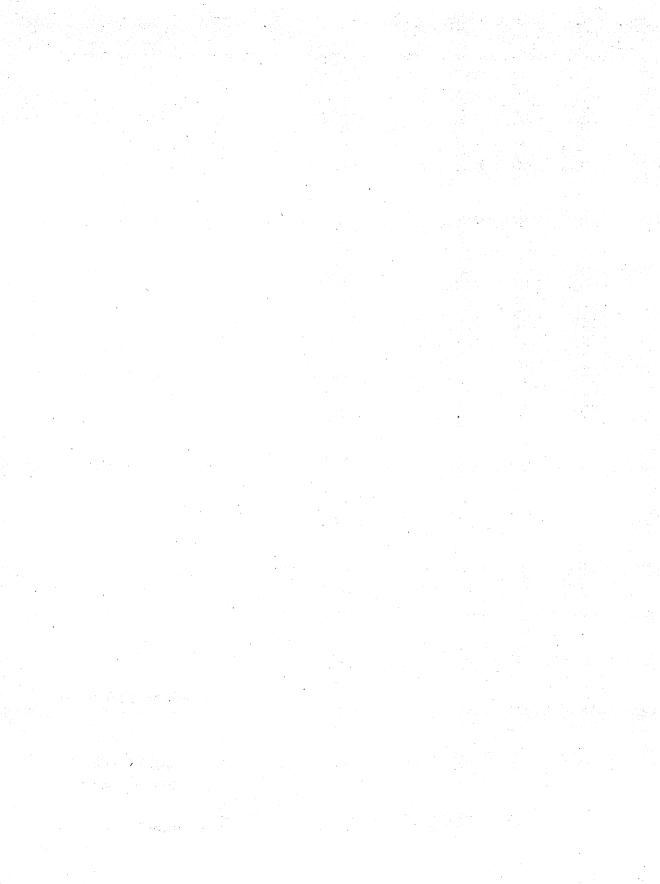
1.70.0

1.70.0

1
                                                                                                                                                                                                                                                                                                                                                        ĕ
                                                                                                                                                                                      DOCTOR DE CONTRA DE CONTRA
             11800
      11810
11820
11830
11830
11840
11840
11860
11880
11890
11990
119930
119930
119960
119960
119960
                                                                                                                                                                                         DATA
REM
DATA
DATA
DATA
DATA
DATA
DATA
                                                                                                                                                                                                                                                                                                                                                           0.0.0.0
                                                                                                                                                                                                                                                                                                                                                        0,0,0,0,0
0,0,0,0
0,0,0,0,0
0,0,0,0,0
0,170,128,0,191
128,0,183,128,0
191,128,0,170,128
```

```
0,0,0,0,0
0,0,0,0,0
0,0,0,0
0,0,0,0
0,0,0,0
0,0,0,0
PULSAR PULSES - 2
11990
12000
12010
12020
                   12030
                                     12035
12040
12050
12060
12070
12090
12010
12010
12011
12011
12011
12011
12011
12011
12011
12010
12010
12010
12010
12010
12010
                    121999
122199
122229
122229
1222229
1222229
1222229
1222229
12222
12299
12299
12299
12299
12299
12299
12299
12299
12299
12299
                    REM
                                                    RESTORE MEMORY
                   POKE648,4:PRINT"D":STOP
```

READY.



Sound the Whistle

Igor Tulchinsky

```
191
         REM
                                     INITIALIZATION
192
          GOSUB12000:GOSUB13000:POKEV+21,255
210
211
212
215
497
498
                                          MAIN LOOP
         REM
         FORI=1T02:GOSUB 500:NEXT:GOT0215
                                   MOVE TEAM MEMBER
499 :
500 JS%=NOT(PEEK(56319+I)):POKES+4,0
510 JF(JS%AND16)<5160RFL%(I)>0THEN540
511 JF(NOTJS%AND127)=111THENRETURN
515 POKES+4,129:POKES+1,(RND(8)*$33)+150
520 CL%(1)=12:CL%(2)=11:GOSUB17000:CL%(1)=1:CL%(2)=0
530 JF(JS%AND1)=1THENN%(I)=1
531 JF(JS%AND2)=2THENN%(I)=2
532 JF(JS%AND2)=2THENN%(I)=3
533 JF(JS%AND4)=4THENN%(I)=4
535 GOSUB17000
540 JFFL%(I)>0THEN553
545 DX%(I)=8*((JS%AND4)=4)-8*((JS%AND8)=8)
2/3 IF(DXX(I)ORDYX(I)><>ØTHENPOKES+4,33:
POKES+1,RND(8)*30+10
555 ZN=((I-1)*4+NX(I)>
560 XX(ZX)=XX(ZX)+DYX(I)>
570 YX(ZX)=YX(ZX)+DYX(I)
580 POKEV+1+2*(ZX-1),YX(ZX)
580 POKEV+1+2*(ZX-1),YX(ZX)
590 IFXX(ZX)=STHENPOKEV+16,PEEK(V+16)O
RPX(ZX-1):GOTO600
595 POKEV+16,PEEK(V+16)PDX(ZX)
600 POVEV-16,PEEK(V+16)PDX(ZX)
 8 >
 630
640
           IFXX(ZX)C=342ANDXX(ZX)>=6THEN670
IFXX(ZX)>342THENSX(1)=SX(1)+1:GOSUB1
 <u> จัดอัด</u>
           IFXX(ZX)<6THENSX(2)=SX(2)+1:G0SUB190
```

Sound the Whistle Program

660 GOSUB16000:POKE17000,S%(1):POKE17001 ,S%(2):CLR:GOSUB13000 665 S%(1)=PEEK(17000):S%(2)=PEEK(17001): GOTO210 IFFLX(I)>0THENFLX(I)=FLX(I)-1:GOTO68 670 ō 0 671 Q1X=PEEK(V+30) 672 IF(Q1XANDPX(ZX-1))C>PX(ZX-1)THEN680 674 DXX(I)=-DXX(I):DYX(I)=-DYX(I) 675 IF(DXX(I)ORDYX(I)>C>OTHENFLX(I)=3:CQ X=0:GOTO678 676 CQX=CQX+1:IFCQX=2THENDXX(I)=INT(RND(676 CQX=CQX+1:IFCQX=2THENDXX(I)=INT(RND(8)*2)*8-4:DYX(I)=INT(RND(8)*2)*8-4:GOTO6 75 678 680 RETURN 997 998 REM SPRITE DATA 999 235. 10000 DATA 0,0,0,0,62,0,0,127,0 10001 DATA 0,201,128,0,201,128, 10002 DATA 0,255,128,15,255,248 0,201,128,0,201,128,0,127,0 0,255,128,15,255,248,27,255,2 36 10003 DATA 51,255,230,51,255,230,27,255, Ž36 10004 DATA 12,247,152,0,99,0,0,99,0 DATA 0,193,128,0,0,0,0,0,0 DATA 0,0,0,0,0,0,0,0 10005 10006 11997 11998 REM PRINT BACKGROUND 11999 12000 12005 POKE53280,9:POKE53281,5:PRINT"; 12010 FORI=1T020:PRINT" I "J:NEXT ı ı 12020 PRINT"ST a" : 12030 PRINTTAB(5)" 12040 12050 GOSUB16000 RETURN 12997 12998 REM INITIALIZE SPRITES 12999 13000 RESTORE:FORI=16000T016062:READJ:PO KEI,J:NEXT 13020 FORI=2040T02047:POKEI,250:NEXT:V=5 3248 3246 13030 FORI=V+39TOV+46:POKEI,-11*(I>V+42) -12*(I<=V+42):NEXT 13040_FORI=VTOV+14STEP2:POKEI,50:NEXT:PO KEV+16,240 13050 FORI=V+1TOV+7STEP2:POKEI,70+(V+7-I NEAT 13110 FORI=1T04: Y%(I+4)=Y%(I): NEXT 13120 NX(1)=1: NX(2)=1 13130 CLX(1)=1: CLX(2)=0: FORI=1T02: GOSUB1 7000: NEXT 13140 FORI=0T08: PX(I)=2*I: NEXT 13145 GOSUB18000: POKES+24, 15: FORI=1T060: POKES+1, 150: POKES+1, 200 13146 FORJ=1T04: NEXT: NEXT: POKES+4, 0 13150 RETURN 14997: 13146 13150 14997 14998 REM END GAME 14999 15000 POKE53280,1:POKE53281,0:POKES+4,33 :TI\$="000000" 15010 FORI=0TO200STEP40 :TI\$="00000"
15010 FORI=0T0200STEP40
15020 FORJ=40T0200STEP-5
15020 FORJ=40T00STEP-5
15030 FORK=1TO5
15040 POKES+1,I+J+K:NEXT:NEXT:NEXT:POKES
+24,0:POKEY+21,0
15050 PRINT"SINDING CONTROL OF THE PROPERTY OF 15070 15997 SX(1)=0:SX(2)=0:GOTO209 15998 REM UPDATE SCORE 15999 16000 PRINT" MANAGAMANA PARAMANA PAR

Sound the Whistle Program

```
<5>"### "$%<1>"##
                    ■"TAB(32)"部間 "S%(2)"Ⅲ
16005
       IFSX(1)>90RSX(2)>9THEN15000
16010
16997
      RETURN
16999
17000
17010
                 SELECT TEAM MEMBER
       REM
       POKEV-(38+N%(1))*(I=1)-(42+N%(2))*
(I=2)
17030
      CLX(I)
RETURN
17997
17998
       REM
                   INITIALIZE SOUND
17999
18000
       S=54272:FORL=STOS+24:POKEL
18055
18010 POKES
POKES+24,15
19020 RETURN
       POKES+5,88:POKES+6,195:POKES+4,33:
18997
18998
       REM
                     SCORE SOUND
18999
18333
19000 POKES+4,17:FORI=200TO0STEP-13:FORI
J=1T010:POKES+1,I+J:NEXT:NEXT
19010 POKES+4,0:RETURN
19997
19998
       REM
                     PRINT TITLE
19999
20000
      POKE53280,0:POKE53281,5:PRINT"#";
       PRINT
20010
* ***
                          ***
                 米米米米
       PRINT"
20020
                                 *
                                         -
* *
20030
       PRINT"
       * "
20040
       PŘINT"
20050
       PRINT"
20060
                                      *
                                         366
20070
       PRINT"
* ***
20075
       PRINT
                                         ****
20080
       PRINT"
                         ****
                                      *
      PRINT"
20090
20100
                           *
20110 PRINT"
      PRINT"
20120
20130 PRINT"
                                      44
20140 PRINT"
                                         ****
20145
20150
       PRINT PRINT"
* *
20160
      PRINT"
20170
       PRINT"
20180
       PRINT
20190
       PRINT
       PŘINT
20200
---
*
20210
PRINT INSTRUCTION SCREEN
       REM
20999
       21000
21030 PRINT"
                           #OUND
                                  THE
                                       OHISTLE
21035 PRINT"
21040 |
21040 |
15 TO
                     THE OBJECT OF
                                     THE GAME
IS TO *"
21050 PRINT"
TEAM'S *"
                 PASS
                       THROUGH THE OPPOSING
               *
TEAM'S *"
21060 PRINT"
                                      THE GOA
                  PLAYERS AND CROSS
L-LINE,
```

Sound the Whistle Program

```
21070 PRINT"
A. *"
21080 PRINT"
                          OUTSIDE OF THE SCREEN ARE
21095 PRINŤ"
21100 PRINT"
21110 PRINT"
H YOUR **
21110 PRINT"
H YOUR **
21120 PRINT"
ENT TERM **
21130 PRINT"
RE BUTTON **
21131 PRINT"
IN THE **
21132 PRINT"
T PLAYER. **
21133 PRINT"
                               NOVE A TEAM MEMBER WIT
                          JOYSTICK.
                                           ♥ELECT A DIFFER
                          MEMBER BY PRESSING THE FI
                          AND POINTING THE JOYSTICK
                          DIRECTION ASSIGNED TO THA
21155 |
21170 |
OTHER
21180 |
                                TLAYERS MAY BLOCK EACH
TEN *"
21190 PRINT"
                          POINTS WINS THE GAME.
21210
                                     SATRESS THE FIRE BU
21230 PRINT"
*"
21240 PRINT"
**********
21250 RETURN
                       ***********
```

Sound the Whistle Program

Bumper Ball

Charles Mott, Jr.

```
* (C) 1984 FANFARE HOUSE, INC. *
* BOX 3062 HOUSTON, TX 77253 *
***************
    REM
9 GOTO170
10 GOTO170
11 :
12 REM
13 :
20 JV=PEEK
                                  MAIN CONTROL LOOP
      :
JV=PEEK(JS):IFJV<>126ANDJV<>125THEN50
IFJV=125THENMY=MY+20:GOTO50
MY=MY-20
IFMY<60THENMY=60
IFMY>215THENMY=215
IFBX>236ANDBY>124ANDBY<146THEN1040
BX=BX+BM:IFBX>239THENBM=-BM:BX=240:GO
50
60
70
00 BA-BA-BH-1FBX/239THENBM=-BM-BX=240.GU
SUB810
90 IFBX<0THENPOKEV+21,6:GOSUB740:GOSUB81
0:GOTO840
0:GOTO840
100 BY=BY+CM:IFBY>222THENCM=-CM:BY=221:G
OSUB810
110 IFBY<57THENCM=-CM:BY=58:GOSUB810
120 IFFF=1THENF=F+1:IFF=9THENFF=0:F=0
130 CX=CX+MC:CY=CY+MC:IFCX>230ORCX<70ORC
Y>200ORCY<60THENMC=-MC
140 IF(PEEK(V+30)AND2=2)ANDFF=0THENBM=-B
M:FF=1:BX=BX+BM:GOSUB680:GOSUB810
150 POKEV+3,MY:POKEV,BX:POKEV+1,BY:POKEV
+4,CX:POKEV+5,CY:GOTO20
160 :
160 :
161 REM
                                PRINT TITLE SCREEN
162
170
       POKE646,7
CHARLES M
        JR."
JR."
PRINT:PRINTTAB(6)"(C)
200 FRINC.
210 :
211 REM
INITIALIZATION
 381 REM
                                  PRINT GAME SCREEN
```

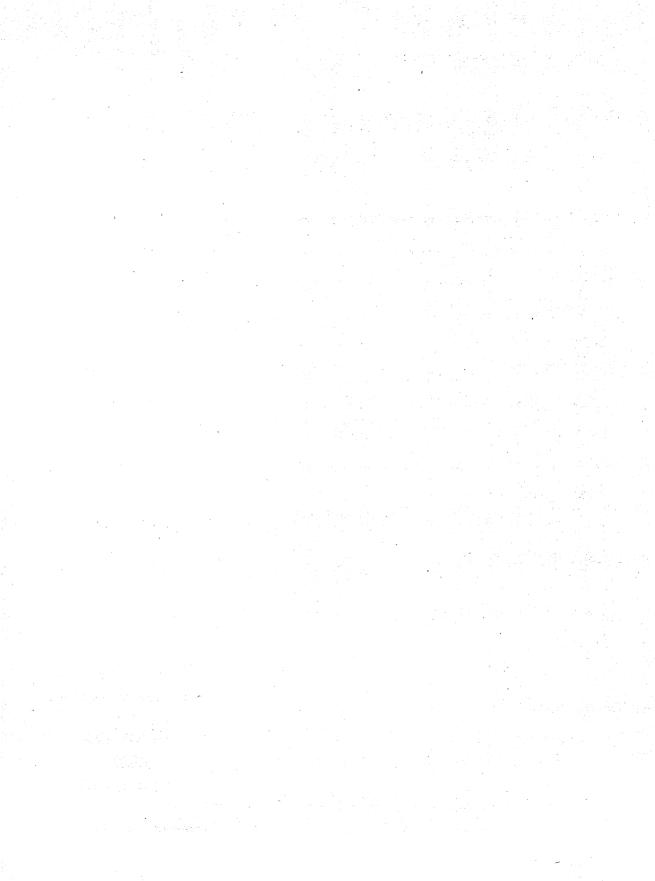
Bumper Ball Program

440 _=BUMPERXBERREREE ISS ALL 450 ਹੁ≉=**"**≱ ISTEL HIBUMPERXER BERNERS BERNE ISSET SENT ALL< 460 Z\$="= 1:22 BALL "470 BL\$(4)="38 0000 "490 BL\$(3)="38 000 "490 BL\$(3)="38 000 "490 BL\$(3)="38 00 "510 BL\$(3)="38 00 "510 BL\$(3)="38 00 "510 BL\$(4)="38 00 "510 BL\$(5)="38 00 "510 BL\$(6)="38 LASTBALL" "520 POKEBK,5:POKE646,11:FORI=1TO31:PRINT "38 "; GOSUB810:NEXT:PRINT"38" 530 FORI=1TO23:PRINTTAB(30)"38 28":GOSUB810:NEXT:PRINT"38" 540 FORI=30TO0STEP-1:POKESC+40*24+I,160:POKECL+40*24+I,11:GOSUB810:NEXT 550 POKEV+21,6:POKE646,0 550 PRINT"300" 550 PRINTTAB(31) W\$ 580 GOSUB810 70 PRINTTAB(31) "30 ":GOSUB8 10 10 "500 PRINT:PRINTTAB(31)"30 ":GOSUB8 10 10 " BALL 470 10 PRINT"XXXXXX"
PRINTTAB(29)X\$
GOSUB810:POKE646,0
PRINT"XXXXXXXXXXTAB(31)V\$
GOSUB810
PRINT:PRINTTAB(31)"X ēĕø 610 620 630 640 650 题":GOSUB a 810 660 670 671 672 GOTO840 REM BALL HIT PADDLE 690 GG=GG+1:PRINTLEFT\$(PY\$,6);LEFT\$(PX\$, 32);CHR\$(18);GG 690 IFJV=1260RJV=125THENGOSUB710:RETURN 700 RETURN 710 IFABS(CM><30THENBM=BM+SGN(BM):CM=CM+ SGN(CM):BM=BM-(BM=0):CM=CM-(CM=0):RETURN 720 730 731 732 740 BALL MISSED PADDLE REM . 740 FORI=1TOPO:MS=MS+1:PRINTLEFT\$(PY\$,23);LEFT\$(PX\$,32);CHR\$(18);MS 750 GOSUB310:NEXT:PO=1 POKESO+7,128:POKESO+12,17:POKESO+13, 140 FORI=9T01STEP-1:POKESO+8,I*9:POKESO+ 11,33 780 F 780°SORY=1T020:NEXT:POKESO+11,32:FORY=1T 0450:NEXT:GOSUB790:NEXT:I=0:GOSUB810 790 POKE646,(IANDNOT5):PRINTLEFT\$(PY\$,20 }LEFT\$(PX\$,31)V\$:RETURN 800 801 REM RANDOM NOTE 802 802 . 810 POKESO+4,16:POKESO+1,RND(0)*40+10:FO RT=1TO5:NEXT:POKESO+4,17:RETURN 820 830 831 REM PREPARE FOR NEXT BALL 832 840 850 868 890 GU19320 870 JV-PEEK(JS):GETA\$:IFA\$="**||**|"THEN920 880 POKE646,Y:PRINTLEFT\$(PY\$,3);LEFT\$(PX \$,31);W\$:Y=Y+1:IFY=3THENY=0 890 GOSUB910:FORI=1TO250:NEXT:IF(JVAND16 890 GOSUB810:FORI=1T0250:NEXT:IF(JVAND16)=16THEN870
900 POKE646,0:PRINTLEFT\$(PY\$,3);LEFT\$(PX\$,31);W\$:Y=Y+1:IFY=3THENY=0
910 GOT01290
920 POKE646,0:PRINTLEFT\$(PY\$,3);LEFT\$(PX\$,31);W\$:GOT01200
930 REM SPRITE DATA
932 :
938 REM PADDLE DATA 0,0,0,0,0,0,0,56,0,0,56,0,0,56, 940 950 DATA 0.0.56.0.0.56.0.0.56.0.0.56.0.0 950 DATA 56.0.0.56.0.0.56.0.0.56.0.0.56.0.0.56.0 0.0.56 960 DATA 0.0.56.0.0.56.0.0.56.0.0.56.0.0

Bumper Ball Program

980 REM BALL AND BUMBER 990 DATH 3,255,192,7,255,224,15,255,240, 31,255,248,63,255,255,255,255,255,25 1000 DATH 255,254,255,255,255,255,25 5,255,255,255,255,255,255,255,25 1010 DATH 255,255,255,255,255,255,25 5,255,255,255,255,255,255,255,255,25 1020 DATH 63,255,255,127,255,254 1020 DATH 63,255,252,31,255,248,15,255,2 1030 PFM 970 0,0 980 DATA 0.56.0.0.56.0.0.56.0.0.0.0.0.0. 1031 BALL HIT PLUNGER 1032 : 1040 GOSUB710:GOSUB1170:POKEV+1,136:POKE V,247:FORY=1T025 1050 PRINTLEFT\$(PY\$,12);LEFT\$(PX\$,29);Y\$ 1060 GOSUB810 1070 PRINTLEFT\$(PY\$,12);LEFT\$(PX\$,29);Z\$ USUBB10 PRINTLEFT\$(PY\$,12);LEFT\$(PX\$,29);Z\$ GOSUB810:NEXT:POKE646,0:JV=0 FORI=1T040:GOSUB680:GOSUB810:NEXT POKEY,239:BX=239:JV=0 PRINTLEFT\$(PY\$,12)LEFT\$(PX\$,29)X\$ GOSUB710 1080 1090 1100 1110 1120 1130 1130 POKEV, BX:BX=BX-4:IFBX<0THENPO=30:PO KEY+21,6:GOSUB740:GOSUB810:GOTO840 140 IFPEEK(V+30)AND1=1ANDPEEK(V+30)AND2 =2THENBY=136:GOTO20 #2|HENBY#136:GU|U20 1150 GOTO1130 1160: 1161 REM SPEED UP BUMPER.BALL 1161: 1170 IFABS<MC><30THENMC=MC+SGN<MC>:MC=MC ~(MC=0):RETURN 1180 RETURN 1190 : 1191 REM 1192 : 1282: 1282: 1289 POKEV+1,136:POKEV,238:POKEV+5,136:POKEV+3,136:POKEV+21,6 1300 CX=INT(RND(1)*150+70):POKEV+4,CX:GO SUB810:FORT=1T01000:NEXT:GOSUB810 1310 POKEV+21,7:FORI=1T01000:NEXT 1320 PRINTLEFT*(PY*,12);LEFT*(PX*,29);Z* :POKEV,247:GOSUB810:FORT=1T01000:NEXT 1330 PRINTLEFT*(PY*,12)LEFT*(PX*,29)X* 1340 GOSUB810:BX=243:CY=136 1350 POKEV+21,7 1360 POKEV,BX:BX=BX-1 1370 IFPEEK(V+30)AND1=1ANDPEEK(V+30)AND2 =2THENBY=136:CX=150:MY=136:GOTO20

Bumper Ball Program



Ride the Wind

Frank Mikulastik

```
*************
        REM
        REM
REM
REM
                            RIDE THE WIND

BY FRANK MIKULASTIK

(C) 1984 FANFARE HOUSE, INC.

BOX 3062 HOUSTON, TX 77253
  12345678
        REM
        REM
                   *************
        REM
                                             INITIALIZATION
  8 ::

9 PRINT"D";

10 POKE56578, PEEK(56578) OR3

11 POKE56576, (PEEK(56576) AND252) OR1

12 POKE53272, (PEEK(53272) AND252) OR1

13 POKE648, 128: FX=1: GOSUB4000: FX=0

15 DIM N(10,2), FI(15): SM=32768

16 FI(1)=19: FI(2)=20: FI(3)=21: FI(4)=22: FI(5)=23: FI(6)=22: FI(7)=21: FI(8)=20

17 FI(9)=19: FI(10)=20: FI(11)=21: FI(12)=2

2: FI(13)=23: FI(14)=22: FI(15)=21

19 N(1,1)=33: N(1,2)=135: N(2,1)=42: N(2,2)

=62: N(3,1)=50: N(3,2)=60

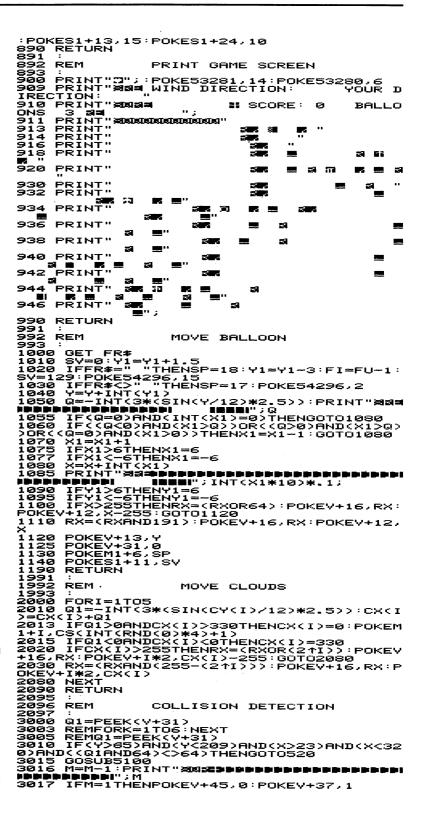
20 S1=54272: V=53248: POKEV+21,126: POKEV+2

8,64: POKEV+29,62: POKEV+23,64: M1=1016+SM

21 CS(1)=16: CS(2)=24: CS(3)=25: CS(4)=26

100 FORI=16*64+SMTO27*64-1+SM: READA: POKE

I,A: NEXT
   I,A:NEXT
300 GOSUB
                              4120
4200
              GOSUB
GOSUB
GOSUB
   310
400
                              900
   410
              GÖSÜB
                              5200
   42ŏ
              GÖSÜB
                              800
   430
              FORI=631T0640: POKEI, 0: NEXT
   497
498
                                                CONTROL LOOP
   499
500
                              1000
2000
3100
              GOSUB
   šĕš
              GOSUB
              GOSUB 310
GOTO 3000
GOTO 500
   507
   510
520
FORK=0T024:POKES1+K,0:NEXT
POKES1+8,51:POKES1+7,51:POKES1+12,15
```



```
3019 IFM=2THENPOKEV+45,5:POKEV+37,13
3030 X=300:Y=70:X1=0:Y1=0:RX=(RXOR64)
3040 POKEV+16,RX:POKEV+12,X-255:POKEV+13,
Y:POKEV+31,0
3053 GOSUB800
            3054
 3055
 3060
                                                           3066
3068
3070
3071
 3072
3080
             GOSUB4200
             GÖTÖ400
 3090
 3091
 3ē92
                                      LANDING DETECTION
3093 :
3100 IF(X)=126)AND(X<=138)AND(Y)=96)AND(Y<=102)THENL=1:K1=1:K2=4:GOTO3200
3110 IF(X)=262)AND(X<=274)AND(Y)=176)AND(Y<=136)AND(X<=274)AND(Y)=176)AND(Y)=190)AND(Y)=190)AND(Y)=200 IF(X)=309AND(X<=42)AND(Y)=192)AND(Y<=200)THENL=4:K1=3:K2=12:GOTO3200
3130 IF(X)=86)AND(X<=98)AND(Y)=152)AND(Y<=160)THENL=8:K1=4:K2=11:GOTO3200
3140 IF(X)=158)AND(X<=170)AND(Y)=128)AND(Y<=136)THENL=16:K1=5:K2=7:GOTO3200
3150 IF(X)=198)AND(X<=170)AND(Y)=120)AND(Y<=120)AND(Y<=120)AND(Y)=120)AND(Y<=120)AND(Y)=120)AND(Y<=120)AND(Y<=120)THENL=6:K1=5:K2=7:GOTO3200
3150 IF(X)=222)AND(X<=210)AND(Y)=192)AND(Y<=200)THENL=64:K1=7:K2=13:GOTO3200
3170 IF(X)=302)AND(X<=314)AND(Y)=194)AND(Y<=200)THENL=64:K1=7:K2=13:GOTO3200
 3093
 (YC=152)THENL=128:K1=8:K2=8:GOT03200
3190 RETURN
 3190
 3190
3191
3192
3193
3200
             REM
                                                 LANDED
 3200 IF(LDANDL)=LTHENGOSUB5020:X=300:Y=7
0:X1=0:Y1=0:RX=(RXOR64):GOSUB5500:RETURN
3230 LD=(LDORL):SC=SC+10:PRINT"XXEDDDDDD
3240 POKE1063+K1,160:POKE55335+K1,K2
3240 POKE1063+K1,160:POKE55335+K1,K2
3241 GOSUB500
3242 X=300:Y=70:X1=0:Y1=0:RX=(RXOR64)
 3240
3241
3242
3243
             X=300:Y=70.X1=0.T1-0.R0-\n
GOSUB800
FORI=631TO640:POKEI,0:NEXT
IFLD<>255THENRETURN
PRINT"900
PETIDM
 3248
3250
 3260
3290
             RETURN
 3990
 3991
3992
             REM
                                      PRINT TITLE SCREEN
 4000
            4005
 4010
            PRINT"
                               22
 4011
 22
                                                            a
            PRINT"
 4013
                                                            22
            PRINT
 4014
                               -3
 4015 PRINT"
                                                         2
                                                            3
                                                                           23
            PRINT
                                       3
 4016
                               -9
                                          71.50
                                                            3
                                     _
                                  23
                                                         23
 4017
            PRINT"
                                            22
                                                      a
            PRINT
 4020
 4030
            PRINT
                                               -2
                                                                -24
4031 PRINT"
                                                                                               28
          PRINT"
                                4:2
 4032
 -2
           734 P
4034 P
                                         28
                                                                                     -3
# 78 # 8
4035 PRINT"
78 # 8 #
4036 PRINT"
4037 PRINT"
                                              24
                                               3
                                                                                               -
                                              3
                                                               4 22
                                                                     23
                                                                                               22
 4040 PŘINT
4045
4050 PRINT"
                                                        BY FRANK MIKULASTI
```

```
(C) 1984 FANFARE HOUSE
         GOSOB/80
POKEY+13,155
GOSUB5200
FORI=631TO640:POKEI,0:NEXT
GOSUB2000
4126
4127
4130
4140
         GETA*:IFA$=""THEN4130
FORI=1TO7:POKEV+I*2+1,0:NEXT
POKE53275,64
4150
4155
4190
         RETURN
4191
4192
4193
4200
                      PRINT INSTRUCTION SCREEN
         REM
         FORI=OTO7:POKEV+1*2+1,0:NEXT
PRINT"D":POKE53280,0:POKE53281,14
PRINT"DME
4205
4212 PRINT": DEE YOU MUST LAND THE BALLOON ON ALL OF "4213 PRINT": DEE
4214 PRINT" THE LANDING PADS (10 POINTS EACH). THE "
4215 PRINT" THE
4216 PRINT" TIME SPACE BAR CONTROLS THE BALLOON'S HEATER"
4217 PRINT" TIME
4218 PRINT": SEE
TO RISE. THE
4219 PRINT": DEE
         PRINT" DWG WHICH WILL MAKE IT START
4220
FT 8
         PRINT " INT BLOWS THE BALLOON LE
FT AND RIGHT.
4229 PRINT"3
4231 PRINT": 138
4232 PRINT":38
        PRINT"XXXXXXI"
PRINT" XXX
PRINT" XX
9456789895
93333333460
944444444
                        2007 BI
...
- P2235000
- P2285000
- P2285000
- P2285000
4991
4992
          REM
                              SOUND ROUTINES
4992 :
5000 POKES1+24,15
5005 FORJ=1TO3
5010 POKES1+6,15:POKES1+5,8*16+9:POKES1+
4,17:POKES1,N(J,2):POKES1+1,N(J,1)
5011 POKES1+12,15:POKES1+13,15:POKES1+11
,33:POKES1+7,N(J,2):POKES1+8,N(J,1)
5015 POKEY+29,0:FORI=1TO100:NEXT:POKEY+2
 .33
5015
5015 PUREY+29,0 PORT=110100 MEAT FOREY+2
3,127:FORT=1T0100:NEXT
5016 POKEY+29,63:FORT=1T0100:NEXT:POKEY+
23,64:FORT=1T0100:NEXT
5017 POKES1+4,16:POKES1+5,0:POKES1+6,0
5018 POKES1+11,32:POKES1+12,0:POKES1+13,
NEXT
POKEY+31,0
 5060
5090
          RETURN
 5099
 5100
          FORJ=0T024:POKES1+J,0:NEXT:POKES1+2
          POKES1+5,15: POKES1+6,15
```

```
5102 IFFL<>1THENPOKEV+37,8:POKEV+38,2:PO

KEV+45,7

5110 FORJ=1TO15

5113 IFFL<>1THENPOKEM1+6,FI<J>

5115 FORI=0TO5:POKEV+39+I,0:NEXT

5120 POKES3281,J:POKES3280,J-1

5130 POKES1+24,16-J:POKES1+4,129:POKES1+

5,15:POKES1+1,15-J:POKES1,75-J*5

5135 FORI=0TO5:POKEV+39+I,1:NEXT

5140 POKES1+4,0

5150 NEXT

5160 FORJ=0TO24:POKES1+J,0:NEXT

5170 POKE53281,14:POKE53280,6:POKEV+38,2

:POKEM1+6,17

5190 RETURN

5199 :
 5190
5199
  5199 :
5200 FORI=0T024:POKES1+I,0:NEXT
5210 POKES1+24,15
5210 POKES1+24,15
5220 FORI=1T0200STEP?
5225 POKES1+4,17:POKES1+5,16:POKES1+6,15
:POKES1,I:POKES1+1,I
5230 FORJ=1T05:NEXT
5240 POKES1+4,16
5250 NEXT
 5220
5225
  5230
5240
5250
 5260 POK
5290 RET
5497 :
5498 REM
                        POKES1+4,0
RETURN
                                                                                     GOOD LANDING
5499:

5500 FORK=1TO3

5505 POKES1+24,15

5510 POKES1+6,15:POKES1+5,8*16+9:POKES1+

4,17:POKES1,N(4-K,2):POKES1+1,N(4-K,1)

5513 POKES1+12,15:POKES1+13,15:POKES1+11

,33:POKES1+7,N(4-K,2)

5514 POKES1+8,N(4-K,1)

5514 POKES1+8,N(4-K,1)

5520 POKEY+23,0:FORI=1TO100:NEXT:POKEY+2

9,127:FORI=1TO100:NEXT

5530 POKEY+23,64:FORI=1TO100:NEXT:POKEY+29,63:FORI=1TO100:NEXT

5535 POKES1+4,16:POKES1+5,0:POKES1+6,0

5537 POKES1+11,32:POKES1+12,0:POKES1+13,0
 5499
5500
 Ø
  5540 NEXT
 5545
5550
5560
                         FORK=ØT024:POKES1+K,Ø:NEXT
GOSUBSØØ
FORK=631T064Ø:POKEK,Ø:NEXT
  5590
5990
                         RETURN
  5991
                         REM
                                                                                    SPRITE DATA
 5992
6291
                         REM
                                                             CLOUD 1
                        DATA
DATA
DATA
DATA
  6300
                                                0,0,0,0,0
0,0,0,0,0
0,0,0,0,0
0,1,252,0,3
254,0,7,255,0
7,255,128,15,255
128,15,255,192,31
255,192,63,255,192
127,255,224,255,255
248,255,255,255,255
127,255,255,127,255
127,255,255,127,255
127,255,256,3
254,0,0
BALLOON
 6310
6320
  6330
 6340
6350
                         DATA
                        6360
 6370
6380
 6390
 6400
 6410
6420
6430
6572
6580
                         DATA
REM
                                               BALLOON

150,0,10,150,160

42,105,168,169,105

106,165,105,90,165

170,90,165,170,90

165,170,90,37,170

88,37,105,88,41

105,104,9,105,96

10,105,160,2,85

128,2,150,128,0

150,0,0,0

0,0,20,0

BALLOON WITH HEATER

10,150,160,42,105

168,169,105,106,165

105,90,165,170,90

165,170,90,165,170

90,38,41,105,104

9,105,88,41,105,104

9,102,85,128,2

150,128,0,100

0,150,0,0

0,150,0,0

0,0,0,0
                         DATA
 6590
                         DATA
 6620
6620
                         DATA
DATA
                         DATA
 6630
6640
6650
                         DATA
                         DATA
DATA
                        DATA
DATA
DATA
DATA
DATA
DATA
DATA
REM
DATA
DATA
                         DATA
                         DATA
                         DATA
                          DATA
                         DATA
                         DATA
  6790
 š800
 ēšīē
                         DATA
 6820
6830
                         DATA
```

DATA

```
6840
6850
6891
             DATA 20,0,0,20,0
             DATA Ø
                                BALLOON EXPLOSION - 1
6900
6910
6920
             DATA
                           0,0,0,0,0
             DATA
                          0.0.0.0.0
6930
             DATA
6940
6950
             DATA
                           0.0.0.0.0
                          0.0.0.0.0
0.0.0.0.0
0.0.112.0
0.0.112.0
0.0.0.0
0.0.0.0
0.0.0.0
0.0.0.0
0.0.0
0.0.0
             DATA
6960
             DATA
             DATA
DATA
DATA
DATA
DATA
6970
6980
6990
7000
7000
7010
7020
7035
7050
7050
7080
             DATA
DATA
REM
                          0.0.0.0.0
0.0.0
_ BALLOON EXPLOSION - 2
                          BALLOON EXPLOS
0,0,0
0,0,0,0
0,0,0,0
0,0,0,0
2,170,128,10,102
96,9,85,160,10
125,96,9,125,160
10,85,96,9,153
160,2,170,128,0
0,0,0,0,0
             DATA
DATA
DATA
             7ēsē
7090
7090
7100
7110
7120
7130
7140
                          8.8.8.8.8
8.8.8.8.8
8.8.8.8
             DATA
7150
7150
7160
7175
7180
7190
             DATA
DATA
DATA
                          ø
             REM
                          BALLOON EXPLOSION - 3
             7190
7200
7210
7220
7230
7240
7250
7260
7270
7280
7290
             DATA
DATA
DATA
7290
7300
7305
7310
7320
7330
7340
7350
                          REM
             DATA
DATA
DATA
DATA
             DATA
7360
7370
7380
7380
7390
             DATA
             DATA
DATA
DATA
DATA
7400
7410
7420
7430
             DATA
7440
7445
7450
             DATA
                          0,0

BALLOON EXPLOSION - 5

0,136,0,34

34,32,8,136,136

34,34,34,170,170

168,38,102,106,169

153,152,37,149,234

169,223,88,38,125

106,169,255,152,39

117,106,169,87,88

38,102,106,169,153

152,42,170,170,136

136,136,34,34,34

8,136,136,0,34

0,0,0,0
             REM
DATA
DATA
DATA
7460
748Ö
             DATA
             7490
7500
7510
7520
7530
7540
7550
7560
7570
7598
             DATA
             REM
                                CLOUD
7600
7610
7620
7630
             DATA
DATA
DATA
DATA
                          7640
7650
7660
7670
7680
7680
7690
7790
7720
7730
7735
7750
             DATA
             REM
             DATA
  750
760
                          0,126,0,0,255
0,0,255,128,1
```

DATA

```
7770 DATA 255.192,1,255,192
7780 DATA 3,255,224,3,255
7790 DATA 224,7,255,240,7
7800 DATA 224,7,255,252
7810 DATA 255,252,127,255
7820 DATA 255,255,255,255
7830 DATA 255,255,255,255
7840 DATA 255,255,255,255,255
7850 DATA 255,255,255,255,257
7860 DATA 255,255,255,255,257
7860 DATA 255,255,255,255,257
7860 DATA 255,254,31,128,248
7870 DATA 256,254,31,128,248
7870 DATA 256,254,31,128,248
7870 DATA 256,254,31,128
7890 DATA 254,0,0,53
7990 DATA 254,0,125,255,255
7950 DATA 255,255,255,255,255
7950 DATA 255,255,255,255,255
7950 DATA 255,255,255,255,255
7950 DATA 254,255,255,255,255
7950 DATA 255,255,255,255,255
7950 DATA 255,255,255,255,255
7950 DATA 255,255,255,255,255
7950 DATA 255,256,255,255,255
7950 DATA 255,256,255,255,255
7950 DATA 255,256,255,255
7950 DATA 255,256,255,255
7950 DATA 255,256,255,255
7950 DATA 254,255,255,255
7990 DATA 254,255,256
7990 DATA 254,255,256
7990 DATA 252,0,33,240
7970 DATA 252,0,33,240
7970 DATA 252,0,33,240
7970 DATA 252,0,33,240
7970 DATA 254,0,0
```



Death Valley Patrol

Igor Tulchinsky

```
REM
    REM
12345678
    REM
    REM
    REM
                           INITIALIZATION
9 POKE46,60:POKE52,60:CLR:GOSUB36000
10 GOSUB30000:GOSUB32000:GOSUB31000:POKE
54276,129:Q=PEEK(53278)+PEEK(53279)
18
     REM
                               MAIN LOOP
19
20
     PK=PEEK(56320):POKE54273,9:GS=GS-.015
*NM-.02
21 IFGS-
22 IF(C1
   MM-.02

IFGS-INT(GS)<.1THENGOSUB1000

!IF(C1AND3)=2THENPOKE2041+INT(RND(8)*3

T(C1/900)+RND(8)*2:IFC1>3150THEN40000

!IF(PEEK(53279)AND1)=1THENGOSUB34000

!IF(PEEK(53278)AND1)=1THENGOSUB34000

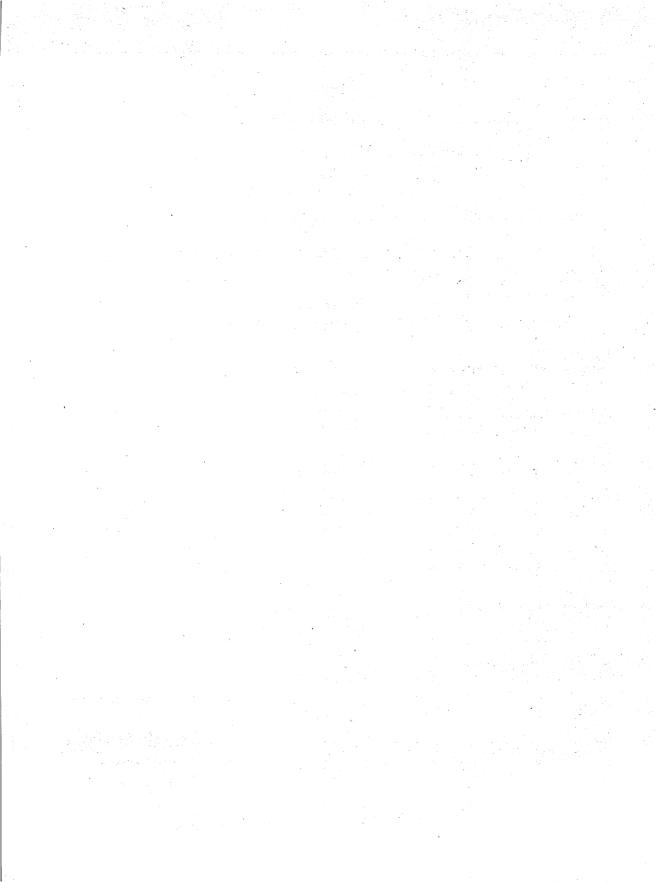
!IFFK=127THEN200
23
24
29
30
      IF(PKAND1)=0THENDY=DY-.7:IFDY<-10THEN
ĎŸ≈-1ø
     IF(PKAND2)=0THENDY=DY+.7:IFDY>10THEND
Y=10
50 I
     IF(PKAND4)=0THENDX=DX-.7:IFDX<-10THEN
     IF(PKANDS)=0THENDX=DX+.7:IFDX>10THEND
240
245
256 IFTI<36000THENPOKE54276,33:FORI=1T01
00:POKE54273,RND<8)*100+150:NEXT:POKE542
00 FORE34273,KND(8)#100+150:NEXT:PUKE542
76,129
257 X=40:Y=63:GOSUB31000:POKEV,X:POKEV+1
,Y:G=PEEK(53278)+PEEK(53279)
260 GOTO20
997:
998 REM ADJUST FUEL GAUGE
999
999 :
1000 POKE55296+35+40*(20-INT(G8)),32:IFG
$<1THEN40000
1010 C2=C2+1:IF(C2AND3)<>2THEN1030
1020 AA=RND(8)*10:BB=RND(8)*10:DX(AA,BB)
=DX(AA,BB)*2:DY(AA,BB)=DY(AA,BB)*2
1030 RETURN
9997 :
9998 REM
                    HELICOPTER - 1
9999
ĨÕÕÕO DATA 0.15,255,0,55,254,0,48,0
```

```
240,48,0,48,62,0,63,227,128
63,224,192,1,255,224,0,255,19
10001
            DATA
10002
            DATA
                        0,127,128,0,60,0,0,0,0
            DATA
                       0.0.0.0.0.0.0.0.0.0
0.0.0.0.0.0.0
0.0.0.0.0.0
10004
            DATA
10006
10007
                          HELICOPTER -
            REM
10008
10009
                        255,192,0,127,176,0,0,48,0
60,48,0,48,62,0,63,227,128
63;224,192,1,255,224,0,255,19
10010
10011
             DATA
            DETE
10012
                        10013
10014
10014
10015
10016
            DATA
             ĎAŤA
10017
10018
            REM
                          VICTIM -
10019
                       0.0,0,0,0,0,0,0,0
0,28,0,0,62,0,62,0
0,28,0,0,28,0,0,127,128
0,255,192,1,190,96,3,62,48
6,62,24,4,28,8,0,28,0
0,54,0,0,99,0,193,128
0,193,128,0,193,128,1,193,192
            DATA
DATA
DATA
DATA
10021
10022
10024
10025
             DATA
             DATA
10026
10027
                          VICTIM -
10028
             REM
10029
10030
                        0,28,0,12,62,12,6,62,24
3,28,48,1,156,96,0,255,192
0,127,128,0,62,0,0,62,0
0,62,0,0,28,0,0,28,0
0,119,0,17,193,196,31,0,124
12,0,24,0,0,0,0,0
             DATA
             DATA
DATA
DATA
DATA
10031
10032
10034
10035
             DATA
             DATA
10037
10038
             REM
                          VICTIM -
10039
                        0,0,0,0,0,0,0,0,0
0,28,0,0,62,0,0,62,0
0,28,0,0,28,0,0,127,128
0,255,192,1,190,96,3,62,48
1,190,24,0,252,8,0,28,0
0,54,0,0,99,0,0,193,128
0,193,128,0,193,128,1,193,192
10040
             DATA
10041
             DATA
DATA
10042
             DATA
10043
10044
             DATA
10046
             DATA
10047
10048
                           VICTIM -
             REM
10049
                        0,0,0,0,0,0,0,0,0
0,28,0,0,62,0,0,62,12
0,28,24,0,28,48,0,127,224
0,255,128,1,190,0,3,62,0
1,190,0,0,252,0,0,28,0
             DATA
10051
             DATA
10052
10053
             DATA
                        0,54,0,0,99,0,0,193,128
0,193,128,0,193,128,1,193,192
             DATA
10055
10056
 10058
             REM
                           VICTIM -
 10059
                        0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0
0,28,0,0,62,0,62,0
2,28,0,3,156,0,8,255,0
12,254,0,7,190,0,3,62,0
0,30,0,0,59,0,0,113,128
0,198,192,0,223,64,0,115,0
             DATA
DATA
DATA
 10060
 10061
 10062
10063
             DATA
             DATA
 10065
 í ööss
             DATA
 10067
                           VICTIM -
 10068
10069
             :
DATA 0.0.0.0.0.0.0.0.0.0
DATA 0.0.0.0.0.0.0.0
DATA 0.96.0.0.240.0.6.248.0
DATA 12.120.0.12.28.0.7.255.0
DATA 3.254.0.0.62.0.0.62.0
DATA 0.30.0.0.59.0.0.115.0
DATA 0.199.128.0.223.128.0.115.0
10070
10071
10073
10074
10075
10076
10076
 10078
             REM
                           VICTIM
 10079
             DATA
DATA
DATA
DATA
                         0.0.0.0.0.0.0.0
 10080
                         0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0
0,0,0,15,0,0,63,192,0
255,224,0,63,255,128,31,131,1
 10081
10082
10083
 10084
              DATA
 10085
             DATA
                         6,1,128,6,0,0,198,0,6
198,63,207,255,255,255,255,25
 10086
5,255
              REM
                           VICTIM ~ 8
 10088
 10089
              DATA 0,0,0,0,0,0,0,0,0
 10090
```

```
0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0
0,0,0,15,0,0,63,192,0
255,224,0,63,248,0,31,140,0
6,0,6,7,0,198,7,6
198,63,207,255,255,255,25
              DATA
DATA
DATA
DATA
DATA
DATA
 10091
 10093
 10094
 10095
 10096
 10097
 10098
              REM
                           LANDING DETECTOR
 10099
 10100
10101
10102
              DATA
DATA
DATA
                         0,0,0,0,0,0,0,0,0,
                         0.0.0.0.0.0.0.0.0.0
0.0.0.0.0.0.0.0.0
 10103
10104
10105
10106
              DATA
DATA
DATA
                         0,0,0,0,0,0,0,0,0
0,0,0,192,0,3,192,0,3
0,0,0,0,0,0,0,0
                         0.0.0.0.0.0.0.0.0
 10107
10108
              REM
                           EXPLOSION CLOUD
 10109
10150
10151
10152
                         0,126,0,1,255,192,3,255,240
3,255,240,7,254,248,30,49,248
123,199,30,255,126,127,254,11
              DATA
              DATA
 5,63
10153
5,230
10154
              DATA 252,243,159,126,127,124,55,12
              DATA 27,255,254,60,115,56,125,255,
 10155
              DATA 249,255,252,124,255,158,63,35
 , ī ē 4
10156
             DATA 31,254,120,3,135,224,0,0,0
 29997
 29998 REM
                             SPRITE INITIALIZATION
 29999
29999 :
30000 V=53248:SM=0:FORI=12288T013030STEP
4:SM=SM+PEEK(I):NEXT:IFSM=9481THEN30010
30005 RESTORE:FORI=0T011:FORJ=0T062:READ
Q:POKE12288+I*64+J,Q:NEXT:NEXT
30010 POKE2040,192
30020 POKE2041,195:POKE2042,195:POKE2043
,195
30030 T(0)=194:T(1)=196:T(2)=198:T(3)≈20
0
30040 POKE2044,202:POKE2045,202:POKE2046
,202:POKE2047,203
30050 POKEY+39,1:POKEV+40,1:POKEY+41,0:P
OKEY+42,9:POKEY+43,8:POKEY+44,8:POKEY+45
30060 X=40:Y=63:P0KEV+46,11
 30080 POKEY+29,16+32+64+128
30090 POKEY+23,128:POKEY+16,0:POKE53275,
 16+32+64
30095 GS=10.9:DF=-.095:DX=2:DY=-1:S<0>=1
92:5(1)=193
30100 POŘÉV+21,14:RETURN
30997
 30998 REM
                                  CREATE BACKGROUND
FORI=1T015
PRINT"#
";:POKE54273,255-I*10:NEXT
31010
31020
31030 PRINT"#
             POKE2023,160:POKE56295,9
             C=6:B=6
FORA=1T033STEP6
GOSUB31900:E=INT(R*10):IFABS(E-D)>
31050
31060
31070
10THEN31070
31080 D=E:GOSUB31700:NEXT
31080 D==:0030631700:NEX|
31081 GOSUB31900:A=INT(R*4)*6+1:B=5:C=6:
D=19:GOSUB31700
31082 FORI=A+5TO30:Q=SIN(I/4)*2+3:FORJ=B
31082 FORI=A+5T030:Q=SIN(I/4)*2+3:FORJ=B
+D-4-QT0B+D-Q:POKE1024+I+40*INT(J),32
31083 POKE54273,ABS(Q)*2
31084 POKE55296+I+INT(J)*40,2:NEXT:NEXT
31085 PRINT"3"TAB(31);:FORI=0T020:PRINT"
31086 FORJ=20T024:FORI=31T039:POKE1024+I
31086 FORJ=20T024:FORI=31T039:PUKE1024+1
+J*40,160
31087 POKE55296+I+J*40,2:NEXT:NEXT
31090 GOSUB31900:FORI=0T01:POKEY+7,0
31100 GOSUB31900:FORI=0T01:POKEY+7,0
1140:POKEY+3+2*I,0:POKEY+2+2*I,P<I)
31105 POKE2041+I,0:POKEY+2+2*I,P<I)
31106 IFABS(P<0)-P<1)><20THEN31090
31107 POKEY+21,14
31110 FORI=0T01:Q=PEEK<53279>:C=0
31120 POKEY+3+2*I,C:C=C+3:POKE54273,255-
```

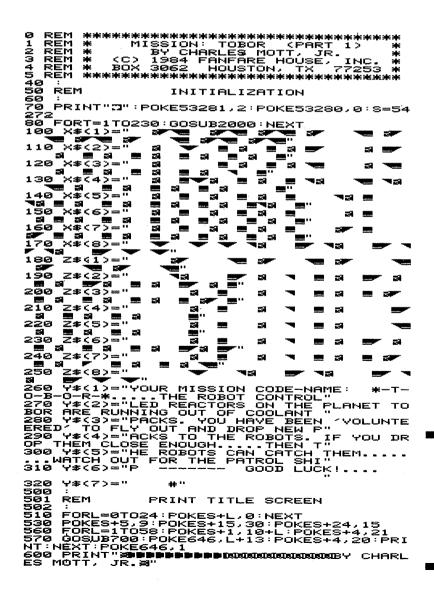
31130 Q=PEEK(53279):Z=SIN(8)*0+PEEK(5327 9):IFZ=0THEN31120 31140 POKEY+3+2*I,C-11:POKE2041+I,194 31150_FORZ=1TO30:Q=PEEK(53279):NEXT:NEXT :POKE2043, 194 POKÉV+6,240:POKEV+7,188 POKEV+8,P(0)-13:POKEV+9,PEEK(V+3)+ 31170 POKEV+10,P(1)-13:POKEV+11,PEEK(V+5 775 31180 POKEV+12, PEEK(V+6)-8: POKEV+13, PEEK 31180 PUKEYTIE, LE... (Y+7)+15 31190 POKEY+21,254 31200 Q=PEEK(53279)+SIN(812):PS=PEEK(532 79)0RPEEK(53279)ORPEEK(53279) 31210 PRINT" SOUNDERS " 31220 PRINT" SOUNDERS " 31230 PRINT" SOUNDERS " 31230 PRINT" SOUNDERS " 31240 POKEY, X:POKEY+1, Y 31240 POKEY,X:POKEY+1,Y 31240 POKEY,X:POKEY+1,Y 31250 PORI=20T010STEP-1:POKE1024+35+40*I ,160:POKE55296+35+40*I,0:NEXT 31260 PORI=20T020-GSSTEP-1:POKE1024+35+4 0*1,160:POKE55296+35+40*1,7:POKE54273,I* 10:NEXT *8,1 31280 POKE1024+35+40*22,5:POKE55296+35+4 0*22,1 31290 PRINT"%"TAB(33)"XXXXXSCORE" LV=LV+1 LV=EV+1 LV\$=STR\$(LV):LV\$=MID\$(LV\$,1,LEN(LV 31650 X=40:Y=63:DX=4:DY=0:POKE54276,129: POKEV+21,255 RETURN 31697 31698 REM CLEAR A HOLE 31699 31700 FORI = ATOA + C - 1 : POKE 54273, I/3 : FORJ = BTOB+D-31710 POKE1024+.T#40+T.32:NEXT:NEXT:RETUR 31900 RP=RP+1:R=ABS(SIN(RP)+SIN(RP*2)):R =R*100:R=R-INT(R):RETURN 31997: 31998 REM SOUND THITT-32000 -32998 32999 REM SPRITE-BACKGROUND COLLISION IFPEEK(V+1)>740RPEEK(V)>60THEN3307 ø 33006 POKE54276,33:IFABS(DX)>20RABS(DY)> ZTHĒN33080 ZIMEN33888 33010 AG=AG+10.2-GS:DX=1:IFAG>50THENPOKE 54276,129:Y=Y-3:POKEV+1,Y:RETURN 33040 GS=10.9:F=1:GOSUB31250:F=0:Y=Y-3:P OKEV+1,Y:DY=-1:POKE54276,129:POKEV+21,25 4
33045 DX=1:DY=-2
33045 DX=1:DY=-2
33050 POKEV+21,255:Q=PEEK(53278)+PEEK(53
279):PS=PEEK(53279)ORPEEK(53279)
33060 PS=PSAND254;RETURN
33070 ZQ=(PEEK(53278)*0+SIN(8)*0+PEEK(53
278)):IF(ZQAND1)=1THEN33300
33080 POKE54276,129:POKEV+16,0
33090 POKE54276,129:POKEV+16,0
33090 POKEV+14,PEEK(V)-12:POKEV+15,PEEK(V+1)-12:POKEV+21,254
33100 FORI=15TO1STEP-.3:POKEV+46,RND(8)*

```
15:POKE54296,RND(8)*5+I:POKE54273,20+I
33110 POKEV+14,PEEK(V)-RND(8)*3+1.5:POKE
V+15,PEEK(V+15)-RND(8)*3+1.5:Q=8IN(8):NE
XT
33120 POKEY+16,128:POKEY+15,255:POKE5429
23120 POKEV+16,128:POKEV+15,25:POKE 33120 POKEV+16,128:POKEV+15,255:POKE5429 6,15:DX=0:DY=0:POKE54276,33 33130 POKEV+21,255:Q=PEEK(53278)+PEEK(53279):X=40:Y=65:POKEV,X:POKEV+1,Y 33140 NH=NH+1:IFNH=5THEN4000 33297 : 33299 5
 33298
           REM
                                  VICTIM RESCUE
 33299
33200 Q=PEEK(53278)*Ø+SIN(8)*Ø+PEEK(5327
8):ZZ=Q:IF(QAND1)=ØTHEN33400
33301 IFDX)30RDY>3THEN33080
33305 POKE54276,33:POKE54273,70:Z=SIN(8↑
 243)
 33306 Q=PEEK(53278)*0+SIN(8)*0+PEEK(5327
8):ZZ≈Q
33310 I
 33320
 33320 IF(QAND32)=32THENNM=NM+1:POKEY+5,0
:POKEY+11,0:IF(QAND4)<>4THENSC=8C+101(NM
  -NK>
 33330 IF(QAND64)=64THENNM=NM+1:POKEY+7,0
:POKEY+13,0:IF(QAND8)<>8THENSC=SC+101(NM
 -NK)
-nn,
33340 GOSUB35000:Q=PEEK(53278)+PEEK(5327
9):POKE54276,129:PS=SIN(8)*0+PEEK(53278)
33350 Y=Y-DY-3:POKEY+1/Y:Q=PEEK(53278)+P
EEK(53279):DY=-3:RETURN
33400
           RETURN
33997
33998 REM
                          SPRITE-SPRITE COLLISION
33999
           Q=PEEK(53278)
IF(QAND2)=2THENQ=18:NK=NK+1:SC=SC+
34000
34010
5:GOSUB35000:GOT033310
34020 IF(QAND8)=8THENQ=72:NK=NK+1:SC=SC+
34030 IF(QAND8)=8THENQ=72:NK=NK+1:SC=SC+
5:GOSUB35000:GOTO33330
34040 RETURN
34997 :
34998 REM
                                    PRINT SCORE
34999
35000 S$≕STR$(SC):S$≕RIGHT$(S$,LEN(S$)-1
350<u>1</u>0
           IFLEN(S$)<5THENS$="0"+S$:GOT035010
PRINT"知成成成"TAB(33)"="S$:RETURN
35020
35997
35998
35999
            REM
                            PRINT TITLE SCREEN
36000 PRINT"เวลเมมมมมมม":POKE53281,1
36008 PRINTTAB(9)"ม
36010 PRINTTAB(9)"% DEATH VALLEY PATROL
36012 PRINTTAB(9)"#
360060**
36020 PRINTTAB<17>"$62 BY 1000"
36030 PRINTTAB<11>"$1■ IGOR T
                                                      TULCHINSKY X000
"DRIDGEDS
36065 POKEV+40,0:POKEV+41,5:POKEV+42,10
36070 POKEV+21,255
36080 FORI=1TO75:POKE2041+RND(8)*3,194+R
ND(8)*4:Q=SIN(8):POKE54273,RND(8)*20:NEX
.
36090 POKEV+3,0:POKEV+5,0:POKEV+7,0:POKE
54296,0:RETURN
39997 :_...
39998 REM
                                          ENDING
39999
37777
40000 POKE54276,33:FORI=10TD100STEP5:FOR
J=I+10TOISTEP-1:POKE54273,I+J:NEXT:NEXT
40010 POKE54296,0:POKEV+21,0
40015 POKE53281,0
40020 IFPEEK(56320)=127THEN40020
40030 RUN10
```



Mission: Tobor

Charles Mott, Jr.



605 PRINT"**治療的影響的**International internation INC。 INC. 1984 FANFARE HOUSE, INC. 第" 610 FORT=1T08:X\$(T)="":Z\$(T)="":F=FRE(0) :NEXT____ :NEXT
620 FORT=1T0255:POKES+5,T:POKES+4,21:FOR
620 FORT=1T0255:POKES+5,T:POKES+4,21:FOR
Y=1T07:NEXT:POKES+4,20:NEXT:GOT0740
700 PRINT"M":FORT=1T08:PRINTRIGHT*6(X*(T),L):NEXT:POKE53265,PEEK(53265)OR64
720 PRINT"MUMUMUMUMUMUMUMUMU":GOSUB2000
730 FORY=1T08:YEXT:
POKE53265,PEEK(53265)AND191:RETURN 736 737 738 740 ŔEM SCROLL INSTRUCTIONS MESSAGE GOSUB2000: GOTO850 902 : 903 REM READ SPRITE DATA INTO MEMORY 2000 READA:IFA=256THEN3500 2100 POKE12928+U,A:U=U+1:RETURN 3101 : 3104 4900 4901 4902 SPRITE DATA REM

5037 DHTH 239,239,58,235,175,36,236,239,58,235,175,256,239,239,58,250
5038 DATA 191,58,255,255,53,255,255,53,2
39,239,53,239,239,53,231,231
5039 DATA 53,255,255,63,255,255,63,255,2
55,47,255,254,3,192,60,0
5040 DATA 47,255,254,63,255,255,63,255,2 ,1,8,128,1 5065 DATA 24, 128,0,66,0,4,66,32,2,36,64, 1,36,128,8,129 5066 DATA 16,6 DATA 16,68,18,34,66,8,66,33,96,132, 1,40,2,145,64 DATA 1,8,128,0,0,0,0,0,0,0,0,0,0,0, 20.74 5067 0.0 5068 DATA 0,0,0,0,0,0,129,0,0,66,0,0,9 8,128,1 DATA 8,128,1,24,128,0,66,0,4,66,32, 2,36,64,8,129

5070 DATA 16,68,18,34,66,8,66,33,96,132, 2,145,64,1 5071 DATA ,8,128 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 õ <u>5</u>072 DATA 0,0,0,0,0,0,0,48,0,0,48,0,0,48 ,0,0 5073 DATA 48,0,0,48,0,0,48,0,0,252,0,3,2 255,171 DATA 252,131,255,8,0,252,0,0,0,0,0, 48,0,0,48,0,0,48,0,0,252,0,3,2 55,0, 5074 0,0,0,0,0 5075 DATA 5076 D 12,0,0 5077 D DATA 0,12,0,0,14,0,0,14,0,0,12,0,0, ĎATA 12,0,0,12,0,0,60,0,0,255,0,0,2 51,0,255,251 5078 DATA 0, 0.0,251,0,0,255,0,0,60,0,0,12, 0,0,12,0 5079 DATA 0,12,0,0,12,0,0,14,0,0,14,0,0, 12,0,0 12,0,0 5080 DATA 0.176,0,0,48,0,0,48,0,0,48,0,0 5080 DATA 0.176.0.0.48.0.0,48.0.0.48.0.0 ,48.0.0 5081 DATA 48.0.0,48.0.0.60.0.0.255.0.0.2 39.0.0.239 5082 DATA 255.0.239.0.0.255.0.0.60.0.0.4 8,0,0,48,0 5083 DATA 0,48,0,0,48,0,0,48,0,0,48,0,0, 176,0,0 1/5,0,0 5084 DATA 0,0,12,0,0,12,0,0,56,0,0,56,19 2,0,192,48 5085 DATA 0,192,12,3,0,3,3,0,0,252,0,0,2 52,0,0,236 5086 DATA 0,0,252,0,0,252,0,3,0,0,3,0,0, 12,0,0 5087 DATI 24,0,0,0 ĎATA 12,0,0,48,0,0,48,0,0,224,0,0,2 5088 48,0,0,48,0,0,44,0,0,12,0,0,3, 3066 0,3,3 5089 DATA 0,12,0,192,48,0,192,192,0,63,0 อย90 DATA 0.0.63.0.0.63.0.0.0.192.0.0.19 2.0.0.48 5091 DATA 0.0.48,0.0.12.0.0.12.0.0.11.0. 0.3.0 5092 DATA 0.0.3.0 ก 59 0,0,63,0,0,63,0,0,192,0,0,19 DATA 0.0.3.0.0.11.0.0.12.0.0.12.0.0 ,48,0 5093 DATA 0,48,0,0,192,0,0,192,0,63,0,0, 63,0,0,59 5094 DATA ,48,3,0,12 5095 DATA 0,0,63,0,0,255,0,0,192,192,3,0 12,0,3,44,0,0,48,0,0,48,0,0,0 0,0,0 5<u>0</u>96 DATA 224,0,0,48,0,0,12,0,0,12,0,0,3 ,0,0,3 5097 DATA 0,0,0,192,0,0,192,0,0,63,0,0,6 5097 DHIH 0 3,0,0,59 5098 DATA 0 192,12,0,48 5099 DATA 4 0,0,63,0,0,63,0,0,192,192,3,0, 48,0,48,0,0,12,0,0,14,0,0,3,0, 5099 DATA 48.0,48.0,0,12.0,0,14.0,0,3,0,0,3,0
9,3,0
5100 DATA 0,0,0,0,20,0,2,170,128.2,170,1
28.1,85,64.2
5101 DATA 170,128,1,85,64,2,170,128,14,1
70,176,254,170,191,14,170
5102 DATA 176,2,170,128,1,85,64,2,170,12
8,1,85,64,2,170,128
5103 DATA 2,170,128,0,20,0,0,0,0,0,0,0 5108 DATA 0,0,0,0,0,0,0,0,0,2,170,128,10,170,160,42 5109 DATA 0,168,42,204,168,170,221,170,8 5,221,85,170,221,170,85,221 5110 DATA 85,170,170,170,42,170,168,42,1 70,168,10,170,160,8,170,32 5111 DATA 8,0,32,32,0,8,32,0,8,128,0,2,1 28,0,2,0 5112 DATA 0,0,0,0,0,0,0,0,12,0,192,51,51,48,48 51,48,48 5113 DATA 252,48,15,223,192,0,220,0,0,22 0,0,252 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0, 0,0,0,220. 5114 DATA ĕ Š115 DATA 0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0 ō 116 DATA 0,48,0,0 3,255,0,3,171,0,0,252,0,0,48,0

```
5117 DATA 48,0,0,48,0,0,48,0,0,48,0,0,48
.0.0.48
5118 DA
              DATA 0.0.48.0.0.48.0.0.48.0.0.48.0.
0,48,0
5119 D
              ĎATA
                             0,48,0,3,255,0,3,255,0,3,255,0
5119 DATA 0,48,0,3,255,0,3,255,0,3,255,0,5,255,0,0,136,0,0
5120 DATA 3,255,0,3,171,0,0,252,0,12,48,192,51,51,48,48
5121 DATA 252,48,15,223,192,0,220,0,0,22
0,0,0,220,0,0,252
5122 DATA 0,0,48,0,0,48,0,0,48,0,0,48,0,
0,48,0
5123 D
              ĎATA
                             0,48,0,3,255,0,3,255,0,3,255,0
.0.136.0.0
5124 DATA
3,255,0,3,171,0,0,252,0,4,48,6
100
5129 DATA 0,0,90,168,0,175,255,0,2,170,1
70,0,255,230,1,95
5130 DATA 213,0,255,230,2,170,170,175,25
5,0,90,168,0,160,0,0
5131 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,256
                  0
      REM
      REM
REM
REM
 ī
2
3
4
5
10
20
      REM
         REM
                                           INITIALIZATION
 30
40 DATAYOU'RE FIRED!, JANITOR, CABIN BOY, C
OOK, PRIVATE, SERGEANT, LIEUTENANT, CAPTAIN
45 DATAGENERAL, SUPREME COMMODORE
50 FORT=0709: READA$: G$<7>=A$: NEXT
50 FORT=0T09:READA$:G$(T)=A$:NEXT
90 DIMDR(16),FG(16):R=0:PI=1:PS=202:P1=2
20:RB=231:SI=54272
100 SC=1024:MY=100:MX=100:Y=53248:JS=307
2:S1=2040:S2=2041:S3=2042
103 S4=2043:S5=2044:S6=2045:S7=2046:S8=2
047:BR=53280:BK=53281:PH=9:AP=9:U=12
105 FORK=0T015
107 READDR(K):NEXT
127 POKEY+28,255:FOKEY+37,2:POKEY+38,0:F
ORT=0T07:POKEY+39+T,1:NEXT
130 DATA0,1,5,0,7,8,6,0,3,2,4,0,0,0,5,0
140 GOSUB800:GOSUB2000:POKEY+21,245
 144
145
           REM
                                         MAIN CONTROL LOOP
 146
147
150
146 :
147 GOSUB2300
150 GOSUB500:FD=FD+U:IFFD>255THENFD=MY:F
G(2)=0:POKEY+21,PEEK(Y+21)AND(255-8)
160 POKE53287,-7*(JY<>0):ONDR(JY)GOTO180
,190,200,210,220,230,240,250
170 P1=220:GOTO390
180 MY=MY+AP:P1=220:POKE54273,10:GOTO390
190 MY=MY+AP:MX=MX+AP:P1=225:POKE54273,5
0:GOTO390
200 MX=MX+AP:P1=222:POKE54273,12:GOTO390
210 MY=MY-AP:MX=MX+AP:P1=224:POKE54273,2
0:GOTO390
220 MY=MY-AP:P1=220:POKE54273,15:GOT0390
230 MY=MY-AP:MX=MX-AP:P1=223:POKE54273,1
1 GOT0390
1.6010390
240 MX=MX-AP:P1=221:POKE54273,8:GOTO390
250_MY=MY+AP:MX=MX-AP:P1=226:POKE54273,3
:GOTO390
390 MX=MX+1:MY=MY-1:IFMX>255ANDFG(1)=0TH
ENPOKEV+16,PEEK(V+16)OR1:MX=0:FG(1)=5
391 IFMX<0ANDFG(1)=5THENPOKEV+16,PEEK(V+
16)AND(255-1):MX=255:FG(1)=0
392 POKESI+4,129:IFMY>150THENMY=150
393 IFMY<42THENMY=42
394 IFMX<15ANDFG(1)=0THENMX=15
395 IFMX>65ANDFG(1)=5THENMX=65
400 POKES1,P1:POKEV+4,R:R=R+PH:IFR>255OR
R<0THENPH=-PH:R=R+PH:GOTO420
410 GOTO430
 :GOT0390
410 GOTO430
420 IFFG(3)=5THENPS=PS-PI:FG(9)=5:IFPS(2
02THENPS=202:YL=YL+1:GOSUB3200:PS=203
           POKES8,PS
IFFG(3)=0THENPS=PS+PI:POKES8,PS
```

427 IFFG(3)ANDFG(9)ANDR<PH+1ANDFG(10)THE NPOKES3,231:FG(9)=0:FG(3)=0:FG(10)=0
428 IFFG(3)ANDFG(9)ANDR>(254+PH)THENPOKE \$3,233:FG(10)=5:REM RED ROBOT 430 IFPS>216THENGOSUB900:PS=202:R=0:GOTO 127
435 IFMY>120ANDFG(2)THENFG(8)=5
437 IFFG(8)THENPOKEV+21,PEEK(V+21)OR2:PO
KEV+2,BP:POKEV+3,120
438 IFFG(8)THENBP=BP+U+4:IFBP>255THENBP=
0:POKEV+16,PEEK(V+16)OR2:POKEV+2,BP:FG(7)
>=5:GOTO443
439 IFFG(8)ANDBP>(MX-10)ANDBP<(MX+10)AND
MY>110ANDFG(1)=FG(7)THENPOKES1,219:GOSUB
2000 2000 443 IFFG(8)ANDFG(7)ANDBP>91THENGOSUB1000 443 IFFG(8)ANDFG(7)ANDBP>91THENGOSUB1000
:POKEV+16,PEEK(V+16)AND(255-2)
444 IFMY=150THENPOKES1,218:GOSUB2000
445 IFFG(2)THENBOSUB950
450 POKEV,MX:POKEV+1,MY:POKEV+7,FD:IFFR=
16THEN150
475 IFFG(2)THEN150
480 FG(2)=5:GOSUB600:GOSUB2300:GOTO450 484 485 REM READ JOYSTICK 486 500 JV=PEEK(V+JS):FR=JVAND16:JV=15-(JVAN D15):POKESI+4,129:POKESI+24,3:RETURN 520 525 REM MOVE SHUTTLE 530 ŠÕÕ FD=MY:FX=MX-11:POKEV+7,FD:IFFG<1>THE NPOKEV+16,PEEK<V+16)OR8 610 IFFG<1>=0THENPOKEV+16,PEEK<V+16)AND< 620 TÜRN 650 651 REM INITIALIZE SPRITES 652 :
800 POKES1,P1:POKES2,234:POKES3,RB:POKES
4,230
810 POKES5,227:POKES6,229:POKES7,228:POK
ES8,PS:POKEV+29,254:POKEV+23,254
820 POKEV,MX:POKEV+1,MY
830 POKEV+2,0:POKEV+3,125
840 POKEV+4,R:POKEV+5,207
850 POKEV+6,MX:POKEV+7,MY
860 POKEV+6,MX:POKEV+7,MY
860 POKEV+8,202:POKEV+9,207:POKEV+43,13
870 POKEV+10,155:POKEV+11,207:POKEV+44,7
880 POKEV+12,249:POKEV+13,207:POKEV+45,7
890 POKEV+16,128:POKEV+14,40:POKEV+15,20 . 894 PRINT"3";:POKEBR,2:POKEBK,12:GOSUB11 Ø0 ĕ95 RETURN 896 897 REM END OF GAME 898 936 GOSUB2200 905 POKEV+21,241:POKEV+16,PEEK(V+16)OR1: POKEV+23,255:POKEV+29,255:FORT=1TO30 910 POKEBR,2:POKEBK,1:POKEV,40:POKEV+1,1 90:POKES1,218:POKESI+4,129:FORD=1TO30:NE 20 POKEBR, 1: POKEBK, 2: POKES1, 219: POKEV+3
9,2: POKESI+4, 128: FORD=1TO30: NEXT
939 NEXTT: POKEBR, 2: POKEBK, 6: POKEV+21, 245
: POKEV+39, 1: FG(1)=5: FG(2)=0
940 MY=100: R=0: POKEV+23, 254: POKEV+29, 254
: POKES8, 202: PS=202: R=0
943 LP=YL: IFYL>9THENYL=9
944 POKESI+4, 128: POKEV+21, 0: POKEBK, 12
945 PRINT" = COMMISSION OF THE NUMBER OF THE NUM RETURN 950 951 952 955 REM CATCH COOLANT PACK IFR<<FX+10>ANDR><FX-10>ANDFG(2)THEN9 960 370 IFFD<217ANDFD>197ANDFG<2>THENPOKES3, 232:FG<3>=5:GOTO990 980 RETURN 990 POKEV+21,PEEK<V+21>AND<255-8>:RETURN

```
1000 POKEV+21,PEEK(V+21)AND(255-2):FG(7)
=0:FG(8)=0:BP=0:RETURN
1097 :______
 1098 REM
                                                       PRINT BACKGROUND SCREEN
 īĕĕĕ
1100 FORT=1T015:PRINT" XEE
1100 PRINT" XEENS CONTROL OF THE NEXT
1104 PRINT" XEE
 1105 PRINT" ##
113 PRINT SE TENDER (1) 1120 PRINT SE TENDER (1) 1120 PRINT SE TENDER (1) 1125 PRINT SE TENDER (
                                                                                                                                              # 433
 1140 PRINT"mpmmmmmm
                                                                                                               PRINT"
 1145
                                                                                                                                                                 . SEFT 11 1-
                                                               1150 PRINT" ST: POKESC+999, 160
 1155
1197
1198
1199
                         REM
                                                                                SOUND ROUTINES
1199 :
2000 PS=PS+PI:POKES8,PS:IFPS>216ANDFG(12)
2000 PS=PS+PI:POKES8,PS:IFPS>216ANDFG(12)
2005 FORT=0T024:POKESI+T,0:NEXT
2010 POKESI+5,136
2020 POKESI+6,245
2030 POKESI+24,15
2040 POKESI+1,4
2050 POKESI,85
2060 POKESI,85
2060 POKESI+4,129:FORT=15T00STEP-1:POKES
1+24,T:FORPP=1T05:NEXTPP,T
2070 POKESI+4,129
2080 POKESI+4,129
2080 POKESI+24,3:MY=90:MX=100:GOSUB2300:RETURN
 RETURN
2090
 2000
2210
2220
2230
                        FORL=ØT024:POKESI+L,Ø:NEXT
POKESI+1,130
POKESI+5,9
POKESI+15,30
POKESI+24,15
FORL=1T0255STEP5:POKESI+4,20:POKESI
 2240
2250
 2250
+1,L
2260
2270
2280
                         FORT=1T020:NEXT:POKESI+4,21
FORT=1T050:NEXT:NEXT:FG(12)=5
FG(12)=0:RETURN
 2290
 2300 POKESI+5,85:POKESI+6,85:POKESI+24,1
5:POKESI+1,5:POKESI,185
2310 POKESI+4,33:FORT≃1TO30:NEXT:POKESI+
 PRINT SCORE
 3101
                        REM
  3200 PRINT"==========
 ■YOÙ HAVE
                                                                                                                                                                                 RATIN
 5 PUINIS篇 "
3220 PRINT"阅读"TAB(16)YL
3230 FORT1=1T015:POKEBR,T1:GOSUB2300:FOR
T2=1T050:NEXT:NEXT
3240 POKEBR,2
3250 RETURN
```



Cosmo's Rescue

Frank Mikulastik

```
******************************

* COSMO'S RESCUE *

* BY FRANK MIKULASTIK *

* (C) 1984 FANFARE HOUSE, INC. *

* BOX 3062 HOUSTON, TX 77253 *
      REM
      REM
REM
1234567813
      REM
      REM
                  **********
      REM
                                          INITIALIZATION
      :
S1=54272:FG=1:CT=255:POKE53248+21,0
DIM MS(20),GS(20),MM(20)
POKE56578,PEEK(56578)OR3
POKE56576,(PEEK(56576)AND252)OR1
POKE53272,(PEEK(53272)AND15)OR0
POKE648,128:PRINT"D";:GOSUB3300
37
38
39
40
45 SM=32768:FORI=40*64+SMTO41*64-1+SM:PO
KEI,0:NEXT
46 FORI=16*64+SMTO36*64-1+SM:READA:POKEI
47 FORI=41*64+SMTO47*64-1+SM:READA:POKEI
48 NEXT
76 MS(1)=16:MS(2)=17:MS(3)=16:MS(4)=18:M
S(5)=16:MS(6)=19:MS(7)=20:MS(8)=19
77 MS(9)=16:MS(10)=21:MS(11)=16:MS(12)=2
78 T1=12
78 T1=12
.80 v=53248:CM=55296:M1=1016+SM
81 GS(1)=31:GS(2)=32:GS(3)=31:GS(4)=33:G
S(5)=31:GS(6)=34:GS(7)=31:GS(8)=35
ēż
82 T3=8
83 MM(1)=41:MM(2)=42:MM(3)=43:MM(4)=44:M
M(5)=43:MM(6)=42:MM(7)=41
84 MM(8)=45:MM(9)=43:MM(10)=46:MM(11)=43
:MM(12)=45
85 T4=12
86 A1(1)=26:A1(2)=27:A1(3)=26:A1(4)=28
ēē
89 | =4
95 | POKEV+21,000:POKEV+28,195:POKEV+29,12
8:POKEV+23,128
100 | POKEV+39,5:POKEV+40,5:POKEV+41,7:POK
EV+42,7:POKEV+3,7:POKEV+37,6:POKEV+38,1
300 | FORI=0TO7:POKEM1+1,40:NEXT
303 | GOSUB3300
303 GUSUB3300
305 GOSUB3100
310 GOSUB700
320 POKEV+30,0
330 POKEV+31,0
335 FS=0:ME=3:PRINT"=836_EFT : 3":POKEV+2
335 F
1,255
397 :
 398 REM
                                                   MAIN LOOP
399
400
          :
GOSUB1000
IFFU<=0THENGOTO3700
Q=PEEK(Y+30)
IF(QAND33)=33THENGOTO3600
IF(QAND129)=129THENGOTO3700
IF((QAND65)=65)AND(CT=0)THENGOSUB430
 405
407
410
 415
425
425
436
436
            Q=PEEK(V+31)
IF(QAND1)=1THENGOTO3700
IFCT=1THENGOTO450
GOSUB4000
            GOSUB4100
 490
            GOT0400
 695
 696
            REM
                                    INITIALIZE GREEN ALIEN
```

697 : 700 SP=1:POKEM1,MS(SP):X=32:Y=T5-21:X1=0 :Y1=0:POKEY+16,0 790 RETURN 795 796 REM MOVE GREEN ALIEN iõėe JV=PEEK(56320):FR=(JVAND16):JV=15-(JVAND15) 1005 Y1=Y1+.2:K1=40:K2=40:K3=40:PS=0:WF= O 1010 IFFR<>16THENY2=-.4:GOSUB1200:FU=FU-:WF=129 1030 IFJV=40RJV=50RJV=6THENX2=-.3:GOSUB1 300:FU=FU-1:WF=129 1040 IFJV=80RJV=90RJV=10THENX2=.3:GOSUB1 1040 IFJV=80RJV=90RJV=10THENX2=.3:GOS 400:FU=FU-1:WF=129 1100 X=INT(X+X1):Y=INT(Y+Y1):SP=SP+PS 1105 IFSP>T1THENSP=1 1110 IFX<31THENX=31 1120 IFY>223THENY=223 1130 IFY>223THENY=50 1140 IFX>314THENX=314 1140 IFX>314THENX=314 1141 J1=X-25 1142 IFJ1>255THENRX=RXOR4:POKEV+16,RX:PO KEV+4,J1-255:GOTO1150 1143 RX=RXAND(255-4):POKEV+16,RX:POKEV+4,J1 1150 J2=X+25 1151 IFJ2>255THENRX=RXOR8:POKEV+16,RX:PO KEV+6,J2-255:GOTO1160 1152 RX=RXAND(255-8):POKEV+16,RX:POKEV+6,J2 J2 1160 IFX>255THENRX=RXOR17:POKEV+16,RX:PO KEV,X-255:POKEV+8,X-255:GOTO1180 ,156 1160 IFX>255THENRX=RXOR17:POKEV+16,RX:PO KEV,X-255:POKEV+8,X-255:GOTO1180 1170 RX=RXAND(255-17):POKEV+16,RX:POKEV, X:POKEV+8,X 1180 POKEV+1,Y:POKEV+5,Y:POKEV+7,Y:POKEV 1180 FOKEY+1, T: FOKEY+3, T: FOKEY+7, T: FOKEY +9, Y+22 1182 POKEM1+2, K1: POKEM1+3, K2: POKEM1+4, K3 1183 IFFR=16AND(JV=00RJV=10RJV=20RJV=7)T HENPOKEM1, MS(1): GOTO1186 1185 POKEM1, MS(SP) 1186 POKES1+19, 15: POKES1+20, 15: POKES1+18 , ИБ7 1187 POKES1+14, 4: POKES1+15, 112 1190 PRINT"ЯЗЭВЬВЫВЫВЫВЫВЫВЫ RETURN Y1=Y1+Y2 K3=25 PS=1 Ī 195 1200 1210 1280 1290 1300 RETURN X1=X1+X2 K2=24 1310 1380 PS=1 RÉTÜRN 1390 1400 X1=X1+X2 K1=23 PS=1 1410 RETURN 1490 2095 REM INITIALIZE SPRITES 2096 T5=94:POKEM1+5,30:POKEV+44,8:POKEV+ 3000 10,32:POKEV+11,T5 3005 E1=210:E2=210:F1=310:F2=285:G1=F1:G 2≒F2 3010 2010 A3=1:POKEV+45,4:POKEV+12,G1-255:POK 8010 A3=1:POKEV+45,4:POKEV+16,RX 8020 A4=1:POKEV+46,7:POKEV+14,G2-255:POK 8020 B4=1:POKEV+46,RX 8030 FU=600:CT=0 3040 FORI=0TO24:POKES1+I,0:NEXT:POKES1+2 4,15 3090 RETURN 3095 3096 PRINT PLAY SCREEN REM 3097 : 3100 PRINT"]]"; S100 FRINT"O"; 3105 POKE53280,1:POKE53281,0 3110 PRINT"**d=L**EFT : 3 FUEL ORE : 0 3112 PRINT"**dd** 600 sc 3114 P PRINT" Mai 1200 431 MAR 40 mm W 23 er a 3115 PRINT" 198 2000 72 ... 3116 PRINT"MM 1288 ٠٠ ; Control of 3118 PRINT" 1918 13 : ### (의원 '';

```
3120 PRINT" (48) 2000
                             ानस
3122 PRINT"
        PRINT"##
                      332 31
                             T의의
       PRINT"MS
                           य इस
                                (2) 23
        PRINT"%##
3130
                          ្រាធា
        PRINT"SISI
                          129 29
        PRINT" (48)
                      3 3000
3134
                             [23 74]
        PRINT"
3136
                          F=1 =2
       PRINT" A
3138
                                 120 20
        PRINT"
                      3 (1986)
3140
                                 22
3142
                                    ..
                               ाव हो।
                                 æ1
3144
                                 a
3146
        PRINT"AS
                             152 23
3148
        FRINT"SIS
                                 э
                                                 2059
                               rate
3150 PRINT"MM
                       3 3 3 3 3 3
                                              30 52
                            122:22
3152 PRINT"M#
        PRINT"%
                                              3359
3154
        PRINT"MAN 3000
3156
                            :43
                    II Bels'
3160
3180
3195
3196
3197
3200
        POKESM+999,160:POKE56295,2
        GOSUB3000
        RETURN
        REM
                            GAME OVER
        PRINT"沒那那那那那那那那那那那那那那那那那就看到 GAME OVER
        FORI=0T04:POKEM1+I,40:NEXT:FORI=1T0
3210
3210 FORT=0704:POREM1+1,40:NEX1:FORT=170
23:POKES1+1,0:NEXT
3215 GOSUB3500
3220 FORT=1T0500:NEXT
3240:FORJ=1T0100:NEXT:GOSUB4000:FORJ=1T0
100:NEXT:GOSUB4100
3250 FR=PEEK(56320)AND16:IFFR=16THENGOTO
3250
3240
3290
        PRINT"3"; : RETURN
3295
                    PRINT INSTRUCTION SCREEN
3296
        REM
3297
        PRINT"3";
IFFGTHENGOSUB3500
POKE53281,1:POKE53280,2
PRINT"300 COSMO'S RESCUE
3300
3310
3320
3330
3331
IK"
                                      BY FRANK MIKULAST
        PRINT"X
        PRINT"X
3332
                                <C>
                                      1984
                                             FANFARE HOUS
     ĨNC."
6 PRINT"XXXX
E, INC.
3336 PR
ICK TO"
3338 PR
                           MOVEMENT
                                             USE THE JOYST
        PRINT"
                                        REGIN MOVING THE
3338 PRINT"
GREEN"
3340 PRINT"
IGHT."
3342 PRINT"
UTTON"
3344 PRINT"
UP."
                                        ALIEN LEFT
                                                          AND R
                                        PRESS THE FIRE B
                                            BEGIN MOVING
UP."
3346 PRINT"EM
L THROUGH"
                                           YOU MUST
                            OBJECT
L THROUGH
3348 PRINT"
VERN AND"
3550 PRINT"
5 ALTEN:"
                                        THE DANGEROUS CA
                                                  THE PURPL
3350 PRINT"
E ALIEN."
3352 PRINT"
N COSMO"
3354 PRINT"
LATFORM"
3356 PRINT"
                                                   CAN RETUR
                                             YOU
                                            THE LANDING P
YOU WILL BE SCOR
                                        AMOUNT OF REMAIN
                                       THE STA EE FIRE BU
```

```
4160 RX=RXAND(255-128):POKEV+16,RX:POKEV
+14,G2
4170 A
          04=A4+1:IFA4>T4THENA4=1
POKEM1+7,MM(A4)
POKES1+4,Ø:POKES1+5,5:POKES1+6,5:PO
4184
KES1+4,17
4186 POKES1,1:POKES1+1,MM<A4>
4190
            RETURN
4295
4296
           REM
                          PICK UP PURPLE ALIEN SOUND
4300 CT=1
4310 POKES1+24,15:POKES1+12,15:POKES1+13
4310 FORECT.E...
,15
4320 POKES1+11,32:POKES1+11,33:POKES1+8,
7:POKES1+7,12:FORI=1TO100:NEXT
4330 POKES1+11,0
4335 POKES1+24,15:POKES1+12,15:POKES1+13
            POKES1+24, 15: POKES1+12, 15: POKES1+13
, 15<sup>~</sup>
4337
,10
4337 POKES1+11,32:POKES1+11,33:POKES1+8,
7:POKES1+7,233:FORI=1TO100:NEXT
4339 POKES1+11,0
4341 POKES1+24,15:POKES1+12,15:POKES1+13
 715
,15
4343 POKES1+11,32:POKES1+11,33:POKES1+8,
8:POKES1+7,97:FORI=1TO100:NEXT
4345 POKES1+11,0
4347 POKES1+24,15:POKES1+12,15:POKES1+13
,15
4349
,15
4349 POKES1+11,32:POKES1+11,33:POKES1+8,
9:POKES1+7,104:FORI=1TO500:NEXT
4351 POKES1+11,0
4353 POKES1+24,15
4360 POKEM1+6,40
4390
4495
           RETURN
4496
4497
           REM
                                        FALLING SOUND
4500 J=J+1:IFJ>15THENJ=0
4505 POKE53280,15-J
4510 POKE531+24,15:POKES1+12,15:POKES1+13
,15:POKES1+11,16
4512 POKES1+11,17:POKES1+7,Y:POKES1+8,Y
4512 POKES1+11,17:POKES1+7,Y:POKES1+8,Y
 4595
4596
4597
           REM
                                    SUCCESSFUL RESCUE
4600
4610 POKES1+24,15:POKES1+12,15:POKES1+13
,15:POKES1+11,16
4613 POKES1+11,17:POKES1+8,67:POKES1+7,1
3
4615 FORI=1T0100:NEXT:POKES1+11,0
4617 POKES1+24,15:POKES1+12,15:POKES1+13
,15:POKES1+11,16
4619 POKES1+11,17:POKES1+8,75:POKES1+7,6
9
4621 FORI=1T0100:NEXT:POKES1+11,0
4623 POKES1+24,15:POKES1+12,15:POKES1+13
,15:POKES1+11,16
4625 POKES1+11,17:POKES1+8,84:POKES1+7,1
25
23
4627 FORI=1T0100:NEXT:POKES1+11,0
4629 POKES1+24,15:POKES1+12,15:POKES1+13
,15:POKES1+11,16
4631 POKES1+11,17:POKES1+8,89:POKES1+7,1
a";FS
RETURN
 6000
6010
6020
6995
                                         SPRITE DATA
            REM
            REM
                        GREEN ALIEN -
            DATA
DATA
DATA
DATA
DATA
                        63

6,252,55,0,220

55,0,220,63,130

252,2,130,128,0

170,0,0,170,0

0,130,0,0,170

170,170,0,40,0

0,170,0,0,170

0,0,130,0,2

130,128,5,85,80

21,85,84,21,85

84,5,85,80,0

20,0,0
 7000
                        63
 7010
7020
7030
 7040
7050
            DATA
            DATA
DATA
DATA
DATA
DATA
7060
7070
 7őső
7090
7090
7100
7110
7120
7130
```

ĞŘÉĔŇ ALIEN – 2 63,0,0 55,0,252,55,0

DATA DATA REM

DATA

DATA

7140 7150

220,63,128,220,2 130,252,0,170,128 0,170,0,8,130 0,10,170,0,8 40,8,10,170,170 0,40,0,0,170 0,0,170,0,0 130,0,2,130,128 5,85,80,21,85 84,21,85,84,5 85,80,0,20,0 7160 7170 7180 7190 7200 DATA DATA DATA DATA DATA 7210 7220 7230 DATA DATA DATA 7240 7250 DATA DATA DATA 7260 7270 7275 GREEN ALIEN - 3
0,0,252,63,0
220,55.0,220,55
2,252,63,130,128
2,170,0,0,170
0,0,170,0,120,40,32
170,160,32,40,32
170,170,0,0
170,170,0,0
2,130,128,5,85
80,21,85,84,21
85,84,5,85,80
0,20,0,0
GREEN ALIEN - 4
0,0 DATA 0 REM 7280 7290 7300 DATA 7300 7310 7320 7330 7350 7360 7370 7380 7390 DATA DATA DATA DATA REM DATA DATA 7400 7405 7405 7410 7420 7430 7450 7460 7480 GREEN ALIEN - 4
0.0
0.63,0,252,55
0.220,55,0,220
63,130,252,2,170
129,0,170,0
130,0,0,170,0
170,0,40,0,0
170,0,40,0,0
170,0,0,170,0
0.130,0,2,130
128,5,85,80,21
85,84,21,85,84
5,85,80,0,20
0.0
GREEN ALIEN - 5 DATA DATA DATA DATA DATA DATA DATA 7490 7500 7510 DATA 7520 7520 75545 7555 75560 75560 75560 DATA DATA DATA 0,0 GREEN ALIEN - 5 0,0,0,63 0,252,55,0,220 55,0,220,63,130 170,0,2,130,32 10,170,160,8,40 32,10,170,160,0 40,0,0,170,0 0,170,0,0,130 0,170,0,0,130 0,2,130,128,5 85,80,21,85,84 21,85,84,5,85 80,0,20,0,0 REM 7590 7600 7620 7630 7630 7650 7660 7670 7675 DATA DATA REM GREEN ALIEN - 6
63
3,252,55,0,220
55,0,220,63,130
252,22,130,120,0
170,0,0,170,0
170,0,0,170
170,170,8,40,0
10,170,0,10,170
0,0,2,0,0
10,170,0,10,170
0,0,2,0,0
21,255,84,21,0
21,255,84,21,0
21,255,84,21,0
63,0,0
GREEN ALIEN - 7
655,0,220,55,250,0
0,170,0,0,130
0,170,0,0,130
0,170,0,0,130
0,170,0,170,0,170
0,0,170,0,170,0
1208,0,170,0,170,0
1208,0,170,0,160,0
1208,0,0,170,0,0
1208,0,0,0
1208,0,0,0
1208,0,0,0
1208,0,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208,0,0
1208 DATA DATA DATA DATA 63 DATA DATA DATA REM 78<u>0</u>0 7800 7810 7815 7820 7830 7850 7850 7870 DATA DATA DATA DATA DATA DATA DATA 7870 7880 7890 DATA
DATA
DATA
DATA
DATA 77777777777788 DATA DATA REM DATA LEFT THRUSTER DATA DATA DATA DATA DATA 0.0.0.0.0 8010

Cosmo's Rescue **Program**

8030

DETE

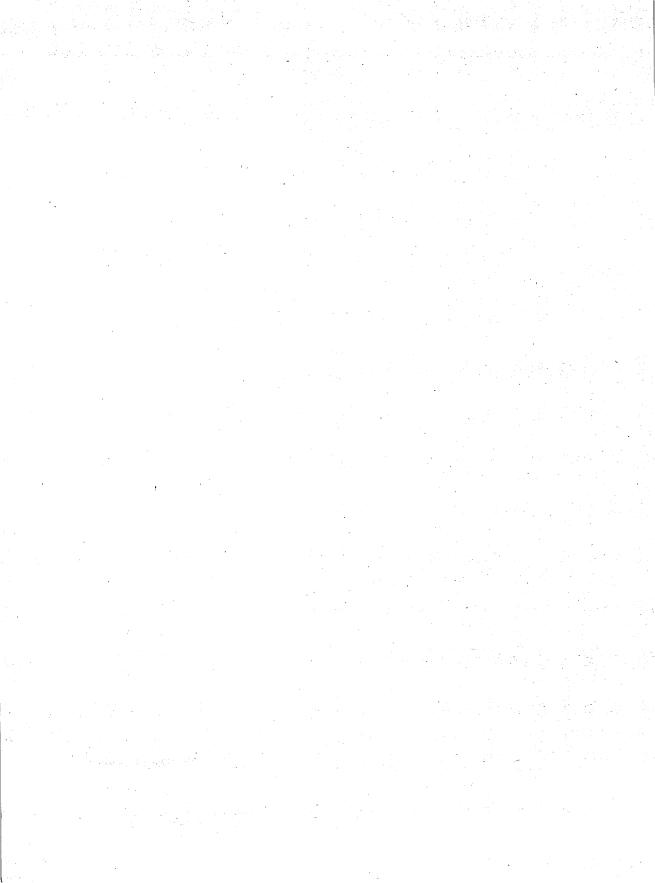
```
0,16,0,0,0
0,0,36,0,0
10,0,0,71,0
0,11,0,0,38
0,0,8,0
RIGHT THRUSTER
8040
8050
8060
           DATA
DATA
           DATA
8070
8080
           DATA
           DATA
šāēŠ
           REM
           DATA
DATA
DATA
8090
                      0,0
8100
8110
                       ອັ.ອັ.ອ.ອ.ອ
                      0,0,0,0,0
0,0,0,0,0
0,0,0,0
8120
8130
8140
8150
           DATA
DATA
DATA
           DATA
DATA
DATA
                      0.0.0.0.0
8160
8170
                      0.0.0.0.0
                      0,0,0,36,0
0,80,0,0,226
0,0,208,0,0
100,0,0,16,0
           DATA
DATA
DATA
ĕī:ĕĕ
8190
8200
8210
           DATA
8220
8225
                      0.0
BOTTOM THRUSTER
           DATA
           REM
                      BOTTOM THRUST
0,50,0,0
125,0,0,235,0
4,93,0,1,34
64,2,73,0,1
36,64,0,18,0
0,0,0,0,0
           REM
DATA
DATA
DATA
DATA
8230
8240
8250
8260
8270
8280
8290
           DATA
           DATA
š300
                       0.0.0.0.0
           DATA
DATA
DATA
8310
                       0.0.0.0.
8320
                       0.0.0.0.
                                         Й
                       อี.อี.อี.อี.อี
8340
8350
8355
                       อ.อ.อ.อ.อ
           DATA
                       0.0.0.0
                                    ALĪEN FALLING - 1
           REM
                       GREEN
8360
8370
8380
           DATA
                      55
0.220,23,0,212
53,0,92,55,130
220,2,130,128,0
170,0,0,170,0
0,130,0,0,170
0,32,40,8,170
170,170,0,40,0
0,170,0,0,170
0,0,130,0,2
130,128,0,0
0,0,0,0,0
           DATA
           DATA
8390
8400
8410
8420
8430
           DATA
           DATA
           DATA
           8440
8450
8460
8470
8480
8490
8495
                       0,0,0
GREEN ALIEN FALLING -
                      GREEN ALIEN FALL
29.0.116
55.0.220.55.0
220.29.130.116.2
130.128.0.170.0
0.130.0.8.130
32.10.170.160.8
40.32.10.170.160
0.40.0.0.170
0.0.170.0.0
0.0.170.0.0
0.0.0.0.0
8500
            DATA
8510
8520
           DATA
8530
           DATA
8540
           DATA
8550
           DATA
8560
8570
8580
           DATA
DATA
DATA
           DATA
DATA
DATA
2590
8600
8610
8620
8630
8635
           DATA
                       ອັ.ອັ.ອັ.ອັ.ອັ
                      DATA
           ŘEM
8640
865<u>0</u>
           DATA
DATA
DATA
8660
8670
           DATA
           DATA
DATA
DATA
8680
DATA
           DATA
           DATA
           DATA
DATA
DATA
                       ŠHĨĖ
           REM
           DATA
                       ĕ,ē
                       0.0.0.0.0
0.0.0.0.0
0.0.0.0
           DATA
ĕŻ
           DATA
šėēē
8810
            DATA
                       0,0
8820
8830
                       0,0,0,0,0
           DATA
            DATA
                       0,0,0,0,0
8840
8850
                       ē.ē
            DATA
                       0.0.0.0.0
                      0,0,0,0,0
0,0,0,0,0
0,5,85,80,21
85,84,21,85,84
5,85,80,0,20
0,0
8860
            DATA
8870
            DATA
8880
8890
           DATA
```

ĽÁŇDING PLATFORM

REM

255,255,255,255,255 255,255,255,255,255,255 255,255,255,255,6 255,134,96,255,6 96,126,6,96 60,6,96,62,6 96,255,6,97,255 134,99,255,198,103 255,230,111,255,246 127,255,254,127,255 254,257,255,254,255 254,257,255,254,255 255,255,0 8910 8920 8930 DATA DATA DATA 8940 DATA 8950 DATA 8960 DATA 8970 DATA 898Ø 8990 DATA 9000 DATA 9010 9020 DATA 9030 9040 9045 DATA DATA 9050 DATA DATA 15 195,240,13,195,112 13,195,112,15,235 240,0,130,0,0 170,0,32,40,8 170,170,170,0,170 0,0,40,0,0 170,0,0,130,0 0,130,0,2,130 128,0,0,0 0,0,0,0 0,0,0,0 15 9060 9070 DATA DATA DATA 9080 9090 9100 9110 9120 DATA DATA DATA 9130 9140 9150 DATA DATA DATA DATA 9160 9170 9180 9185 DATA 0.0,0,0,0 0.0,0 0.0,0 15,195,240 15,195,240 15,67,208,15,107 208,15,195,240,0 130,32,0,170,160 32,40,32,170,170 160,0,170,0,8 40,0,10,170,0 0,2,0,0,2 0,0,2,128,0 0,0,0,0,0 0,0,0,0,0 DATA REM 9190 9200 9210 9220 DATA DATA DATA 9230 9240 9250 DATA DATA DATA DATA DATA DATA 9260 9270 9280 9290 DATA 9300 9310 9320 DATA DATA 0.0.0.0.0 ĕ 0
PURPLE ALIEN - 3
15,195,240,7,193
240,7,233,240,15
195,240,8,130,0
10,170,170,170,0
170,0,0,40,32
0,170,160,0,128
0,0,128,0,2
128,0,0,0
0,0,0,0 REM DATA DATA DATA 9330 9340 9350 DATA DATA DATA 9360 9370 9380 9390 9400 DATA 9410 9420 DATA 9430 DATA 0,0,0,0,0 0,0,0,0,0 0,0,0,0,0 0,0,0,0 FURPLE ALIEN - 4 15,195 240,15,65,240,15 105,240,15,195,240 0,130,0,0,170 0,32,40,8,170 170,170,0,170,0 8,40,32,10,170 9440 9450 9455 DATA REM DATA DATA DATA 9460 9470 948Ø 9490 9500 9510 DATA 9520 DATA 9530 9540 9550 DATA 160,0,0,0,0 DATA 0.0.0.0.0 9560 9570 9580 ē,ē,ē,ē,ē 0,0,0,0,0 0,0,0,0 DATA DATA 9590 DATA 0.0 PURFL 9595 REM PURPLE ALIEN - 5
15,195,240,7
195,208,7,235,208
15,195,240,8,130
32,10,170,160
0,170,0,40
0,0,170,0,0
130,0,130,0
2,130,128,0
0,0,0,0,0 9600 9610 9620 DATA DATA DATA 9630 9640 9650 DATA DATA 9660 DATA 9670 9680 DATA DATA 9690 DATA 9700 9700 9710 9720 9725 DATA 0,0,0,0,0 0,0,0,0,0,0 0,0,0,0,0 0,0,0,0,0 YELLOW MONSTER - 1 170 130,170,128,130,2 143,195,242,13,235 112,13,235,112,13 235,112,15,235,240 2,130,128,34,170 DATA DATA REM DATA DATA DATA DATA 9740 9750 9760 9770 9780 DATA

```
136,170,170,170,170
170,170,170,170
170,170,120,10,170
160,10,170,160,10
170,160,10,170,160
2,130,128,2,130
128,10,130,160,10
130,160,0
YELLOW MONSTER - 2
170,130,170
15,195,240,13,195
112,13,235,112,13
235,112,15,235,240
34,0,136,42,0
168,34,0,136,42
170,168,42,170,168
2,170,128,2,170,168
2,170,128,2,170,160
170,160,10,170,160
120,170,160,10,10
130,160,10,10
9800
9810
                   DATA
982Ø
                    DATA
                    DATA
9830
9840
9850
                    DATA
9860
9865
9870
9880
                    ÃΕΜ
                    REM
DATA
DATA
DATA
DATA
DATA
9890
9900
9910
9920
                     DATA
9930
                    DATA
9940
                    DATA
9950
9960
9970
                    DATA
9980
                     DATA
9990
                    DATA
                                            0
YELLOW MONSTER - 3
15,195,240,13,195
112,13,195,112,13
235,112,15,235,240
2,0,128,2,0
128,2,0,128,34
0,136,170,0,170
170,170,170,2,170
128,2,170,128,10
170,160,10,170,160
10,170,160,10,170,160
10,170,160,10,170
160,2,130,128,2
130,128,10,130,160
10,130,160,0
YELLOW MONSTER - 4
10000
                        DATA
                                            Ø
10005
                        REM
                        DATA
DATA
DATA
10010
10020
10030
10040
10050
10060
                        DATA
                        DATA
DATA
DATA
10070
                        DATA
DATA
DATA
īēēsē
10090
10100
10100
10110
10120
10130
10135
10140
                        DATA
                        DATA
DATA
REM
                                            10,130,160,0
YELLOW MONSTER - 4
0,0
0,15,195,240,13
195,112,15,235,240
2,0,128,2,0
128,34,0,136
42,0,168,42,0
168,2,170,128,2
170,128,10,170,160
10,170,160,10,170
160,10,170,160,2
130,128,10,130,128
10,130,160,0,2
160,0
YELLOW MONSTER - 5
170,130,170,15
195,240,13,195,112
13,235,112,13,235
112,15,235,240,34
0,136,42,170
168,42,170,168
34,0,136,42,170
168,42,170,168,2
170,128,2,170,128
10,170,160,10,170
                        DATA
                        DATA
10160
10170
                         DATA
10150
10190
                        DATA
DATA
DATA
10200
10210
                        DATA
                        DATA
DATA
DATA
DATA
10220
10230
10230
10240
10250
10260
10270
                        DATA
DATA
REM
10275
10280
10290
                        DATA
DATA
DATA
DATA
DATA
DATA
 10300
10310
10320
10330
                        DATA
10340
                         DATA
                                             170,128,2,170,128
10,170,160,10,170
160,10,170,160,10
170,160,2,130,128
10,130,128,10,130
160,0,2,160,0
YELLOW MONSTER -
10360
                        DATA
DATA
DATA
 īÖSSÖ
10390
                        DATA
                        DATA
REM
 10405
                        DATA
DATA
DATA
10410
10420
10430
                                            0,0,15,195,240
13,195,112,15,235
240,2,0,128,2
0,128,34,0,136
42,0,168,34,0
136,42,0,168,42
0,168,2,170,128
2,170,128,10,170
160,10,170,160,10
170,160,10,170,160
2,130,128,2,130
160,10,130,160,10
10440
                         DATA
                        DATA
10460
10470
                        DATA
DATA
DATA
DATA
19479
19489
19499
19599
19519
                        DATA
DATA
10530
10540
                         DATA
                                              128,0,0
20010
                        REM
                                                                  RESTORE MEMORY
                        POKE648,4:PRINT"3":STOP
```



Mazemaster

Igor Tulchinsky

```
0 REM :: 1 REM :: 2 REM :: 2 REM :: 5 REM :: 10 REM :: 10 REM :: 12 GOOG
                                13 GOSUB50000:GOSUB21000:GOSUB22000:GOSU
B20000:ONYAL(G$)GOTO2000,3000
                                                                      3-D PRINTOUT OF MAZE
 100 PRINTC$(1)"%13
110 PRINTR*(1)" 3 "C*(1)"

"L*(1)" "

120 PRINTR*(1)" 3 "C*(2)"

130 PRINTR*(1)" 3 "R*(2)" "C*(2)"

140 PRINTR*(1)" 3 "R*(2)" "C*(2)"

150 PRINTR*(1)" 3 "R*(2)" "C*(2)

"L*(2)" "L*(1)" "

150 PRINTR*(1)" 3 "R*(2)" "C*(2)

"L*(2)" "L*(1)" "C*(2)

160 PRINTR*(1)" 3 "R*(2)" "C*(2)

"L*(2)" "L*(1)" "C*(2)"

170 PRINTR*(1)" 3 "R*(2)" "R*(2)"

170 PRINTR*(1)" 3 "R*(2)" "R*(2)"

"L*(3)" "L*(3)" "L*(2)" "P*(3)" "D*(3)" "D*(3)"
                                                                                                                                                                           "C$(3)"
                                                                                                                                                                          "R$(3)" "C$
" "L$(1)
 180 PRINTR$(1)"# "R$(2)" "R$(3)" "C
$(3)" "L$(3)" "L$(2)" "L$(1)
190 PRINTR$(1)"8
C$(4)" ";
200 PRINTL$(3)"
210 PRINTR$(1)"8
R$(4)" "C$(4)"
                                                                                                          "L$(2)"
"R$(2)"
";
....4)" "C$(4)"
220 PRINTL$(4)"
(1)" "
230 PRINTC*
                                                                                               "L$(Ś)"
                                                                                                                                                         "L$(2)"
236 PRINTR$(1)" # R$(4)" "C$(5)" | 240 PRINTL$(4)" | $(1)" | "
                                                                                                           "R$(2)"
                                                                                                                                                                          "R$(3)"
                                                                                                     ";
"L$<3>"
                                                                                                                                                               "し事(2)"
$(1)" "
250 PRINTR*(1)" "R*(2)"
R*(4)" "R*(5)" "C*(6)" "
260 PRINTR*(5)" "L*(4)"
"L*(1)" "R*(2)"
270 PRINTR*(1)" "R*(2)"
280 PRINTL*(4)" "L*(3)"
*(1)" "
290 PRINTR*(1)" "R*(2)"
                                                                                                                                                   ″L$(3)"
                                                                                                                                                                          "R$(3)"
                                                                                                                                                               "L$(2)"
 Ž90 PRINTR$(1)"#
R$(4)" "G$(4)"
                                                                                                                                                                           "R$(3)"
R$(4)" "G$(4)"

$00 PRINTL$(4)"

(1)" "

310 PRINTR$(1)"

G$(4)"

320 PRINTL$(3)"

325 PRINTR$(1)"

325 PRINTR$
                                                                                               "L$(3)"
                                                                                                                                                         "L$(2)"
                                                                                                                                                                            "R$(3)"
                                                                                   , "L$(2)" "L$(1)" "
"胡 "R$(2)" "R$(3)" "G
"L$(3)" "L$(2)" "L$(1)
```

```
"R$(2)
"L$(2)" "L$(2)"
"L$(2)" "L$(2)"
"L$(2)" "L$(2)"
"L$(2)" "L$(2)"
"L$(2)" "L$(2)"
"L$(2)" "L$(2)
"C$(2)" "L$(2)" "L$(2)
"C$(2)" "L$(2)" "L$(2)"
"C$(2)" "G$(2)"
"L$(1)" "G$(2)"
"C$(1)" "G$(1)"
"G$(1)" "G$(1)"
"G$(1)" "G$(1)"
"G$(1)" "G$(1)"
"G$(1)" "G$(1)"
"G$(1)" "G$(1)"
                                                                                                        "R$(2)"
"L$(1)"
"R$(2)"
"R$(2)"
"R$(2)"
                                                                                                                                                                      "G$(3)"
                                                                                                                                                                "Ģ$(2)"
                                                                                                                                                         "G$(2)"
                                                                                                    、ペチンピタファー・
" "Lまく1)"
"Rまく2)" "(
>" "Lまく1)"
"Gまく2)"
                                                                                                                                                    "G$(2)"
   497
498
                       REM
                                                           3-D PRINTOUT CALCULATIONS
    499
    500
                       REM
DX(0)=0:DY(0)=-1
DX(1)=1:DY(1)=0
DX(2)=0:DY(2)=1
DX(3)=-1:DY(3)=0
DX(3)=-1:DY(3)=0
YY=Y:XX=X:FORI=0TO5
D1=D+1:IFD1>3THEND1=0
D2=D-1:IFD2<0THEND2=3
R$(I+1)=MID$(C$,RX(YY,XX)+1,1):QQ$=R
   530
540
550
    560
    600
   610
611
   620 L$(I+1)=MID$(C$,L%(YY,XX)+1,1)
620 L$(I+1)=MID$(C$,L%(YY,XX)+1,1)
621 IFM%(YY+DY(D1),XX+DX(D1))=ØTHENL$(I+
    622
                         IFMX(YY+DY(D2), XX+DX(D2))=@THENR$(I+
  1)="""
630 C$(I+1)=MID$(C$,C%(YY,XX)+1,1)
640 G$(I+1)=MID$(C$,G%(YY,XX)+1,1)
641 YY=YY+DY(D):XX=XX+DX(D)
648 IFG%(YY,XX)<>>OHTEN656
651 FORZ=I+2TO7:G$(Z)=QQ$:C$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ$:R$(Z)=QQ
   661
670
720
                        RETURN
    997
    998 REM
                                                                                      PRINT HELP MAZE
   999
  999 :
1000 PRINT"INM":D(0)=21:D(1)=18:D(2)=4:D
(3)=12:POKE53281,9:POKE53280,9
1005 PRINTTAB(13)"MAPRESS ANY KEY":PRINT
:PRINT:PRINT
1010 PRINT"MM";:FORII=0T016:FORJ=0T036
1015 IFG%(II,J)=7THENPRINT"#M ";:GOT0103
   ø
    ĭ020
                           PRINTMID#<C*,-R%<II,J>*<G%<II,J><>0
  1020 PRINTMID$(C$,-R%(11,J)*(G%(11,J)</br>
1011)"3 ";
1030 NEXT:PRINT:PRINT"||||";:NEXT</br>
1040 POKE1024+7*40+Y*40**+2,D(D)+128:POK<br/>
E55296+7*40+Y*40*+X+2,T<br/>
1045 IFG$="2"THENPOKE55296+7*40+B*40+A+2,<br/>
7:POKE1024+7*40+B*40+A+2,36+128<br/>
1050 GETA$:IFA$=""THEN1050<br/>
1060 PRINT"D";:POKE53281,9:POKE53280,10:<br/>
EFTIEN"
   ŘĚŤŬRN
   1997 :
1998 REM
                                                                     PAINT FLOOR - MAIN LOOP
 TAğçse>ÑY
  2059 IF(NV=NB)OR((TI/60))TM)THEN40000
2060 GETA$
2061 PRINT:PRINT"$NNNNNNN"TAB(30)"$TIME!"
:PRINTTAB(30)INT(TI/60)
2062 IFTI/60)TMTHEN40000
2063 IFA$=""THEN2060
2070 GOSUB55000:IFA$=""TUCKY"
  7+DY(D)
2075 IFM
GOTO2060
                              IFM%(Y,X)=1THENY=Y-DY(D):X=X-DX(D):
   2089
                              IFA$="R"THEND=D+1:IFD=4THEND=0
```

```
2090
2100
2110
           IFA$="L"THEND=D-1:IFD=-1THEND=3IFA$="H"THENGOSUB1000
           GOTÖ2045
2997
2998
          REM
                       FIND TREASURE - MAIN LOOP
2999
20+50
20+50
3015 POKE55296+(B+7)*40+A+2,1:POKE1024+(
B+7)*40+A+2,36:NEXT:TI$="000000"
3040 POKE54276,0:D=INT(RND(8)*3):C$="■耳3
840 POKE54276;0:D=INT(TND(8)*3):C$="■耳3
840 POKE54276;0:D=INT(TD(8)*3):C$="■耳3
840 POSUB500
3045 GOSUB500
3050 PRINT"到现现即"TAB(30)"而即ISTANCE!":PRI
NTTAB(30)" ■■■"INT(SQR((ABS(A-X))72+(A
"THENX=X+DX<D>:Y=
Y+DY(D)
3075 IFM:
GOTO3060
           IFM%(Y,X)=1THENY=Y-DY(D):X=X-DX(D):
          8060
IFA$="R"THEND=D+1:IFD=4THEND=0
IFA$="L"THEND=D-1:IFD=-1THEND=3
IFA$="H"THENGOSUB1000
GOTO3045
3080
3090
3100
3110
20000
20001
                                CREATE RANDOM MAZE
20020
            POKE53281,3:POKE53280,9:PRINT"";
FORI=55296T056295:POKEI,0:NEXT
PRINT"対域切"TAB(12)"=CREATING THE M
20030
20035
20036
AZE"
           PRINT"SININININI";
FORI=9T016:PRINT"|||||||||||
20040
20045
                                                     HEXT
20050 D(0)=-40:D(1)=1:D(2)=40:D(3)=-1:CR
=7:C9=CM
20060 P=55296+19+40*12
20065 CM=CM-1:F=PEEK(F+D(R)*2)AND15:IF(F
=0ANDCM>0>0R(F=CR)AND(CM>0>)THENF2=F2+1
: GOTO20080
           F2=0:IFCM<0THEN20100
- C7=C7+1:IFC7>INT(C9/120+3.8)THENC7
20066
20067
=0:GOTO20081
=0:G0102081
20070 POKE54276,33:FORI=1T02:POKEP,CR:P=
P+D(R):POKE54273,I*10:NEXT:POKE54276,0
20075 IFF=2ANDCM<0THEN20100
20080 R=INT(RND(8)*4):IFF2<220THEN20088
20081 P=55296+3+INT(RND(8)*16)*2+40*(2+I
NT(RND(8)*10)*2):IF(PEEK(P)AND15)<>CRTHE
N20081
            F3=0:FORI=0TO3:IF(PEEK(P+D(I))AND1
20083
20063 F3-0-1-5-0-1.

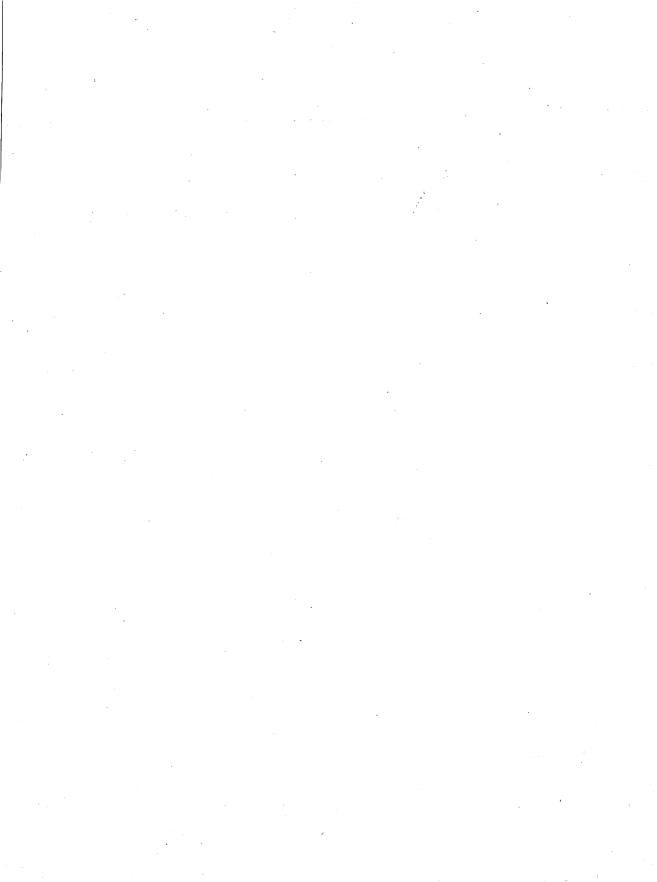
5)<>CRTHENF3=1

20084 NEXT:IFCM<0THEN20100

20085 IFF3<>1THEN20081

20086 G0T020065
            IFF2=0ANDCM<0THEN20100
R=INT(RND(8)*4):CM=CM-1:IFCM<0THEN
20088
20089
20090
            G0T020065
20097
20098
           REM
                                    COLOR THE MAZE
20099
EŽÍŽÍ DIMMX(16,36),CX(16,36),GX(16,36),L
X(16,36),RX(16,36)
X(16,36),RX(16,36)
Z0125 PRINT"≅NNN"TAB(6)" ■COLORING
THE MAZE
THE M
            NB=0:FORJ=7T023:FORI=2T038:P=55296
+J*40+I
20140 IF(PEEK(P)AND15)=2THENM%(J-7,I-2)=
1:POKE54276,0:GOTO20169
20150 IF(PEEK(P)AND15)=7THENM%(J-7,I-2)=
0:POKE54276,33:POKE54273,I*2+J*2
20160 C%(J-7,I-2)=ABS((J+I)AND1)+1
20161 G%(J-7,I-2)=ABS((J+I)AND1)+1
20162 R%(J-7,I-2)=ABS((J/4+I/4)AND11)+3
20163 L%(J-7,I-2)=ABS((J/4+I/4)AND11)+3
20164 IF(PEEK(P)AND15)=7THENNB=NB+1:POKE
  RXCJ
              -7,I-2):GOTO20170
```

```
20169 POKEP,0:G%(J-7,1-2)=0
20170 NEXT:NEXT
20175 PRINT"阅阅阅"TAB(6)"
E HEREE
                                                            STORY OU BR
20180
           POKE53281,9:RETURN
21000
21001
21002
                                 PRINT TITLE
          REM
21010 PRINT"":POKE53281,1:POKE53280,8:P
OKE54276,33
21020 PRINT"SAUGRAGON"TAB<15>"SMAZEMASTER
          PRINT"XXXXXX"TAB(19)"BY"
PRINT"X"TAB(12)"BIGOR TULCHINSKY"
PRINT"XXXXXXX"TAB(5)"X(C) 1984 FAN
21040
                                                                   FAME
          DUSÉ, INC."
POKE54276,0:FORI=1TO1000:NEXT:POKE
       HOUSE
21060
54276,33
21070 FO
21070 FORI=1T0100:POKE53281,RND(8)*15:PO
KE54273,RND(8)*20+100:NEXT
21080 POKE54276,0:POKE54273,20:RETURN
21080
22000
河2>
                                                FIND A TREASUR
22040
           YOUR
22050
           ĞETG$:IFG$<>"1"ANDG$<>"2"THEN22050
GOSUB55000
           JUSUSSSOOO
PRINT:PRINT:PRINT"######";:INPUT"#EN
IZE COMPLEXITY (0-200)";CM:CM=CM*20
IFCM<10THENCM=10
IFCM>4000THENCM=4000
TM=1E9
PETINE
22055
22060
TER MAZE
22061
22062
22070
22080
           RETURN
40000
40001
           REM
                                 GAME OVER
40002
40002 ·· POKE198,0
40010 POKE53281,0:POKE53280,1
40010 POKE53281,0:POKE53280,1
40020 POKE54276,33:FORI=10TO240STEP7:FOR
J=0TO10STEP10:POKE54273,1+J
40021 IFG*<>"2"THEN40029
40025 L=1024+INT(RND(8)*24)*40+INT(RND(8
40005
40010
40020
40021
40025
>*25>
40026
           POKEL,36+128:POKE55296-1024+L,7
L=1024+INT(RND(8)*24)*40+INT(RND(8
40027
)*25)
40028
           POKEL.36+128:POKE55296-1024+L,7
POKE53280,RND(8)*15+1:NEXT:NEXT
PRINT"SUBSISSISSISSISSISS"TAB(30)"#SC
40029
40030
ORE"
40040 IFG$="1"THENSC=INT(NY*C9/20-TI/100
):IFNY=NBTHENSC=SC+C9
40050 IFG$="2"THENSC=C9*3-INT(TI/100)
           PRINTTAB(30)SC
40055
           POKE54276,0
GETA$:IFA$=""THEN40070
40060
40070
40080 CER:GOSUBSS000:GOSUB22000:GOSUB200
00:ONVAL(G$>GOTO2000,3000
49997
49998
           REM
                          SOUND INITIALIZATION
50000 FORI=54272T054296:POKEI,0:NEXT
50010 POKE54296,15:POKE54277,96:POKE5427
8,240:POKE54276,0:POKE54273,20:RETURN
55000 :
55001
           REM
                                SOUND A BEEP
55002
55010 POKE54276,33:POKE54273,20:FORI=1TO
70:NEXT:POKE54276,0:RETURN
```



CBS CBS CBS CBS CBS

Games Disk for

FANFARE HOUSE:

ARCADE GAMES FOR THE C64

ISBN 0-03-003664-X

CBS Educational & Professional Publishing HRW 383 Madison Ave., New York, NY 10017

PAT. PEND: 43-4-1 FD by ROCHESTER 100 ROCHESTER, NY 14603

ARCADE GAMES FOR THE COMMODORE 64

Fanfare House, Inc.

While most other books promise you hours of fun on your Commodore 64, here's a book/disk package that promises you years. . .

Play all 15 games or select just a few — you can change the rules in scores of ways — creating new games for years to come!

Whatever type of game you play — adventure, "space monsters," war games, or swords and sorcery — you can make them harder or easier; change the scoring mechanism; or modify the sprite characters.

All the games are coupled with individual programmer's notes to help beginners and pros learn each game, and programming hints to show quick methods and fast shortcuts.

DISK INSIDE! Includes 15 dynamite arcade games that you can turn into 100's more.

