Adventures with the ATARI®

Jack B. Hardy

Adventures with the Atari

Adventures with the Atari

Jack B. Hardy

A Reston Computer Group Book
Reston Publishing Company, Inc.
A Prentice-Hall Company
Reston, Virginia

Library of Congress Cataloging in Publication Data

Hardy, Jack B.
Adventures with the Atari.
1. Computer games. 2. Atari computer—Programming.
I. Title.
GV1469.2.H36 1984 001.64'2 83-11201
ISBN 0-8359-0172-6

© 1984 by Reston Publishing Company, Inc.
A Prentice-Hall Company
Reston, Virginia 22090

10 9 8 7 6 5 4 3 2 1

Printed in the United States of America

Illustrations by Bruce Bolinger

Acknowledgments

To

My wife, **Karen,** without whose assistance and support this book would never have been written

My sons, **Aaron** and **Jason,** who helped devise some of the plots and play test the adventures

Dave Simpson, for introducing me to fantasy role-playing games

Frank Gazda, for sharing his knowledge of the Atari computer

Evalyn Schoppet, for her help in all phases of production

My friends **Joe, Greg, Bill** and **Bob,** who tolerated me during this venture.

Contents

Preface

As children we have all played at fantasy adventures, from cowboys and Indians to sword fights with villains. We explored the ocean, searched in caves, dug for treasure, and discovered great wealth in a crystal rock. We treated our fantasy role-playing adventures as fact, and enjoyed playing in the worlds we created.

In the last few years we have seen fantasy board games become some of the most popular on the market. There are many varieties, from *Dungeons and Dragons* to (my favorite) *Empire of the Petal Throne*. In *Empire*, the player is human on a planet which was once a great technological world, but due to some great upheaval has reverted to an almost primitive state in which humans fight for survival. More than half of the planet is ruled by other types of creatures; some are friendly or tolerant toward humans, and others wish to wipe them from the planet. The player can explore the surface of the planet or visit the great underground temples (or dungeons). In either case, the purpose is to find treasure and magical items that were left by the former inhabitants.

In each fantasy game, a Dungeon Master presides over the game and is sole judge on the results of any particular move. Before beginning the adventure, each player chooses a fantasy role to play, such as Warrior, Magician, or Priest. Dice are rolled to determine the character's strength, intelligence, dexterity, constitution, charisma, wealth, and "hit" points. The player then has a choice of what type of armor, shields, and weapons to use in the adventure, depending on the character's strength.

To begin play, the Dungeon Master verbally sets the opening scene. Part of a typical session could be as follows:

DM: You are just inside the dungeon. The hallway forks ahead, one fork going northeast, the other northwest.
Player: We approach the fork cautiously, trying to see what is northeast.
DM: Northeast the hallway is very short, ending in a door.
Player: We proceed to the door and listen.
DM: From within, you hear the sound of something scratching the door.

The player (or players) would now have to choose to enter the room or leave. The player is encouraged to believe that he or she really exists in this world of magic and mystery, and to react to everything learned about the environment as if it were real. For this type of game the rules are completely flexible, and creativity is strongly encouraged. Before choosing to enter the room, the player may wish to cast a spell of clairvoyance (if a magician or priest) to see through the door, and, depending on the level obtained in the game, a certain percentage of chance (calculated by a roll of the dice) is given for the spell to work.

There can be only one thing more habit-forming than playing adventures, and that is creating adventures for others to play. If you have ever had the pleasure of being Dungeon Master for a game, you can understand this. And, considering the fascination for home computers that has swept the country almost in tandem with the popularity of the fantasy adventure, the progression to the computer adventure is not a surprising one. The "original" computer adventure was developed on a mainframe at Stanford University in the late 70s by Willie Crowther and Don Woods. Since that time, there have been many different adventures created on all types of computers.

Ideally, the computer adventure would allow the player to be totally creative in his exploration of the world depicted. However, due to the computer's limited storage space and memory capacities, the adventure must be restricted to certain rules. The player is permitted to communicate with a type of abbreviated

English, usually two words (verb-noun). The computer processes the input to see if the words are understood, and then acts on any valid words within the limits of the program's ability. Some types of input are easily utilized, such as WALK, RUN, and GO. All would be sent to the same subroutine for processing the information.

Allowing many different solutions in the same situation is sometimes very difficult, such as with the simple command to open a locked door. The player could use a key, pick the lock, chop through it with a sword, or remove the hinges. Since most adventure programs cannot allow for all these alternatives, the player's creativity may be somewhat limited. Consequently, the use of a computer as Dungeon Master will never be as limitless as a human referee, because no matter how many alternatives are allowed, someone will try something that was overlooked. However, through careful planning and design, computer adventures can be developed which are highly challenging and enjoyable to the most sophisticated of players.

This book will show you how to become the ultimate Dungeon Master by creating your own adventures on your Atari microcomputer. Part 1 provides a brief explanation of the basic steps involved in creating computer adventures, followed by six actual adventures you can study, input, and play. Included are three interactive adventures (Part 2) and three graphic adventures (Part 3), one in each of the Atari PILOT, Microsoft BASIC, and BASIC languages. (For the sake of brevity, a rudimentary knowledge of the computer languages used is assumed. There are many fine books that can give you the basics of each of these languages.) Finally, Part 4 offers instructions and programs to guide you in creating your own computer adventure. Appended to the book are complete program listings for the six games presented in the book, plus additional information which will help you get the most out of your Atari computer and enhance your adventure-making skills.

PART 1

The Making of an Adventure

The basic steps involved in creating a computer adventure are developing the scenario, identifying objects to be used in the game, drawing a map, preparing a flowchart, keying in the program, and, finally, play testing the game. A brief explanation of each of these steps follows. Detailed examples accompany the individual games presented later in the book.

THE SCENARIO

It all begins with an idea. The most creative part of adventure programming is building the scenario on which the adventure is based. The setting of the adventure can be anywhere—on land, on sea, under water, under ground, in space, on the surface of a new planet—any locale your mind can conceive. You could travel from one land to another, explore the human body, search the underground labyrinth of Neptune, or fly to the cloud city of Venus. Likewise, the objectives for adventures are virtually limitless. The objective could be to hunt for treasure, find a missing person, solve a mystery, explore an unknown region, rescue a damsel in distress, or just to survive. Each has the potential for excellence. It is you who must add the necessary elements to make the adventure unique, enjoyable, and exciting.

Once you have made decisions on the components of your adventure, you can begin to organize it onto paper. Start by listing all of the puzzles that need to be resolved in order to reach your adventure's ultimate objective. A puzzle can be as simple as reading a note to decipher part of the adventure, or as complex as having to put several small pieces together to unravel just one part of the overall solution.

Additional complexity is introduced into the game in the form of obstacles. Again, the possibilities are limited only by the creator's imagination: hills too steep to climb, acid lakes, scorching deserts, and villains who seek to kill are just a few examples.

THE OBJECTS

Once obstacles are introduced into the game, the next step is to provide the player with means of overcoming them. A rope ladder to climb the hill, a rubber boat to cross the lake, canteens full of water to help survive in the desert, and methods of shielding oneself from malefactors are some objects that could be provided to aid the player in reaching the game's objective. Of course, you can also add items to your adventure which have no use except to tantalize the player.

As you develop the various elements of your scenario, you should keep a list of the objects needed to overcome each obstacle in the adventure, their uses, and their locations.

When designing an adventure game it is best to employ the most straightforward way of using objects, since nothing can be as frustrating as trying for several minutes to use an object and being told, "You can't do that." The adventure quickly loses its appeal if the adventurer is constantly thwarted in his or her attempts to solve puzzles. It is important to design the adventure so the pace of the program will keep the player's interest. So give careful consideration to the input a player may try.

THE MAP

In order to program the computer with the necessary details of your adventure, you must first create a map. A well-made map will aid you greatly in the design and later debugging of your adventure program. Begin by drawing a square for each location and a line between any connecting locations, and then number each location. Next, enter each object on the map in its proper location. It is often necessary to work back and forth between your list of objects and the map, as you may need to add or delete rooms or items.

Use the map to test the logic of your adventure by being sure that an object can be obtained before it is required for use elsewhere in the game. For example, if a key is needed to unlock a door, you cannot have the key locked behind that door. Your adventure map will be one thing that you will use constantly during programming of the adventure.

THE FLOWCHART

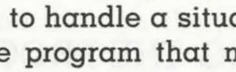
It does not matter how complicated your program appears, the basic form of programming is the same. The easiest way is to carefully think out your program onto paper. The first step is to divide the jobs the program will perform into their smallest parts, or routines. Each routine does a certain job, and together they combine to become the overall program.

The flowchart is the result of careful, step-by-step consideration of the best way for the program to handle a situation. It is an outline showing each part of the program that must be included for the program to work, and is used as a guide in programming the game. Actually, once you have drawn the map of your adventure, you have already designed a type of flowchart of your program.

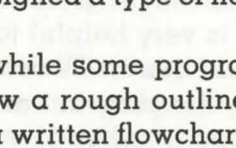
It is worth mentioning here that, while some programmers use a flowchart religiously, some follow a rough outline of the program's execution and do not create a written flowchart. I suggest that you try both, since it is largely a matter of personal preference.

Every programmer develops his own style of flowcharting. Throughout this book, the symbols shown below are used.

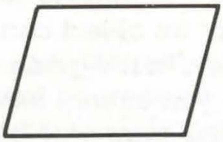
TERMINAL. The terminal symbol indicates the beginning and ending of a program.



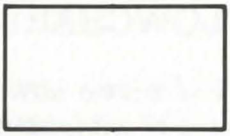
DECISION. A decision symbol indicates the point at which the computer must decide which set of instructions to follow.



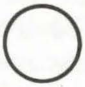
INPUT/OUTPUT. This symbol indicates either that information is being sent to the program by the user, or that the program is supplying information to the user.



PREDEFINED PROCESS. This is the set of instructions the computer will follow regardless of what is entered by the user.



CONNECTOR. The connector is used to show that the program starts, ends, or continues elsewhere.



KEYING THE PROGRAM

Once you have completed the flowchart for your adventure, you can enter the program into your Atari. When keying in your program it is very helpful to be able to see how the text will appear on the monitor or TV screen. This can be accomplished by either running the program or using the GOTO(n) command (where n is the line number of your program to print text). This will aid you greatly in formatting your screen display. A nice feature of the Atari is the ability to use the cursor control characters to format your output. These can be imbedded into your print statements by pressing the escape key, then the control key and an arrow key for the direction in which you would like your text to move.

When keying in programs from magazines or books, remember that each space, comma, colon, semicolon, etc., has a special meaning to the computer and the listing should be followed exactly.

PLAY TESTING

If you think that you are through with your adventure after keying it in, think again! Play testing and debugging are as much a necessity as the actual writing of the adventure. You may have read that a computer program is never actually completed; that goes double for adventure programs. Murphy's Law is well supported–if anything can go wrong, it will, and usually at the worst possible moment. Although you may have worked weeks on a computer adventure, trying every possible combination of survival and death in the game, due to the tremendous amount of user input there is always a chance that something will be overlooked. For example, you may say proudly to a friend, "Take my new game to play and see if you enjoy it." Your friend takes the program and minutes later calls you to ask why your program crashed at line 3000. Frantically, you ask what was entered to cause such a thing, and find that it was an abbreviation of a word in a situation designed to handle only the complete word. Such things can happen, and often do.

The first step in testing your adventure is the initial run of the program. Here you do a straight run through the program, doing everying that should be done in your adventure, at the time it should be done, and at the location it should be done. Make a list of all the errors that you find. If your program "errors out" while running, you can usually continue with the debugging by typing CONT and hitting return.

Mention should be made of the excellent editing features of the Atari. They are a joy to use, because major changes can be effected in the program and the results can be seen immediately. After making the desired changes, type GOTO line (n), and the program will continue from that line without changing any variables. At any point in your program, you can add the STOP command. When your program reaches this point, you can ask for the value of the variable by typing PRINT (variable name or number) and you can see what is in that variable. Then you type CONT (or CON.), hit return, and the program will resume.

It is always a good idea to save your program every 30 minutes or so in case of a power outage or other disaster.

Next comes the major play testing. This is a lengthy process of trying every possible input in each situation to see if the program operates as it should. Rather than starting over at the beginning each time the program crashes, you can use the GOTO feature to return you to the input part where you can continue. Be sure to maintain a list of input that causes errors.

Ideally, you should be trying input that will cause difficulty. It is also a good idea to have someone else try out your program while you just sit and watch the results. Take notes, because you may see things tried that you never thought of. Then it is back to the drawing board to work out the problems you have found so far, and to add features you think would improve the game. Then the cycle starts all over, as you again have to play test all the added features. Remember, no matter how well you have thought out a routine, there will frequently be problems due to the large number of factors involved. Also, you can learn much from debugging programs, since each error made will give you a better understanding of what is required for the computer to accomplish a certain task. Don't worry too much about your programming technique, as this will develop as you learn more about the language you are using and the computer itself. As long as the program does what you want it to do, it is a good program. You can always come back later to make any changes in the program you feel will help.

Enjoy yourself, and program to please yourself. Happy hacking!

PART 2

Interactive Adventures

An interactive adventure game for the computer is written to function as an interpreter. The player, at the computer keyboard, types some form of command, usually in the form of two words (verb-noun). The computer, acting as mediator, accepts the player's input and sends it through the program to be interpreted. The program, finding a message it understands, acts upon the input and returns with some form of output to the player.

The next few chapters deal with the creation of this type of adventure program. There will be adventure programs in Atari PILOT, Atari Microsoft BASIC, and Atari BASIC. In each, I will attempt to show you some of the many ways to handle the information necessary for an adventure game. I have taken for granted that you know most of the commands in each language, such as PRINT, GOTO, FOR/NEXT, etc. But individual commands don't do much; it's the way they are combined that forms the completed working program. Herein the programs are broken down into their smallest working subset of statements to show you the results of each segment and, finally, the completed, playable program.

In each language the program must contain all the information necessary for any command the player might give. Therefore, it is vital that you understand each phase in the construction of an adventure game. For this reason, each program is presented in detail with all the information that you need to make your own adventure programs, and a complete listing of each program is also included in Appendix A. Those of you who would like to play the adventures before you understand how they work can key in the programs from the listings in the appendix, and play them, solve them, and enjoy them right away. You can later refer to the specific section for each adventure for details of how it works.

Each adventure requires the creation of a specific vocabulary, because each adventure will have its own words to help the player solve the puzzles and complete the adventure. Almost all adventure programs use the words TAKE and DROP, but any other words needed can be determined only by the requirements of the adventure program itself. It would be unnecessary to include a word in the program that the player would never need (e.g., there would be little need for the word CLIMB if there is nothing in the adventure to climb). Therefore, a vocabulary is created as the adventure is constructed. You can use as many synonyms for a word as you desire and, indeed, these do help considerably in the user–friendliness of any adventure (e.g., in addition to GO, you could add WALK and RUN; even though each of these sends the program through the same subroutine, they greatly improve the usability of the program).

Each of the three languages, PILOT, Microsoft BASIC, and Atari BASIC, has its own unique way of storing information. This is another thing you will need to know to make your own adventure programs. Each way of handling information will be discussed in detail within the instructions for the program utilizing that language.

IMPORTANT: When keying in each program you will see brackets ([ ]) within the program lines. These are to help you type special graphic characters unavailable by normal keystrokes. In each listing, if there is more than one special character, a number will precede the keystroke. *Example:* [30 ctrl R] indicates that you type the keys control R 30 times.

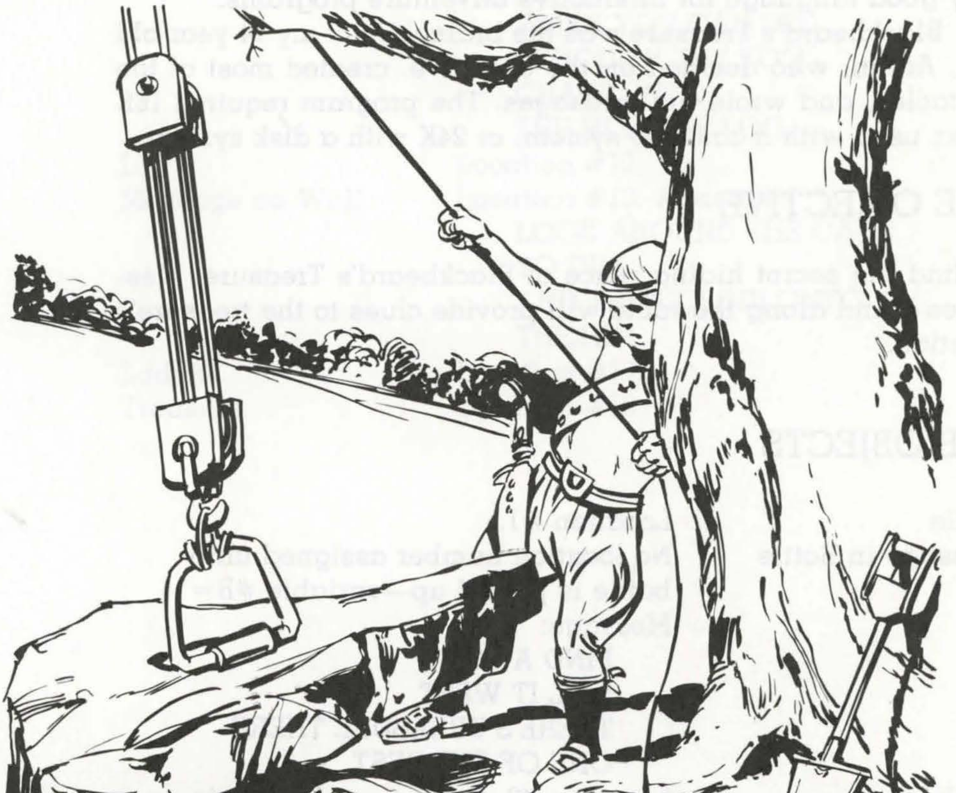
If there are different characters to be used within the listing they will be separated by a slash [/]. *Example:* [esc ctrl −/ctrl E] indicates that you type the keys escape, control, minus, then control, E.

If at any time you are uncertain as to the way your program line should look, refer to the listings in Appendix A. There, each line will be shown as it appears on your monitor or TV screen.

At intervals throughout the listings, there will be run points marked with an asterisk (*). By typing RUN and hitting RETURN, you will be able to see the results of your work up to that point. Many times you will be asked to add, delete, or change routines so you can see the results of a particular action. The intent is to help you understand the results of programming in different ways, such as memory savings, speed of execution, etc.

AN ADVENTURE
IN ATARI PILOT:

Blackbeard's Treasure



Blackbeard's Treasure is written in Atari PILOT, which stands for **P**rogrammed **I**nquiry, **L**earning **O**r **T**eaching. PILOT is the easiest of computer languages to learn and was created as a text-oriented language. It was first conceived as a tool for writing conversational programs, and is designed to handle all types of user input with ease. There is a tremendous amount of user input required to play an interactive adventure; therefore, PILOT is a very good language for interactive adventure programs.

Blackbeard's Treasure was the brainchild of my 14-year-old son, Aaron, who decided on the objective, created most of the obstacles, and wrote the messages. The program requires 16K when used with a cassette system, or 24K with a disk system.

THE OBJECTIVE

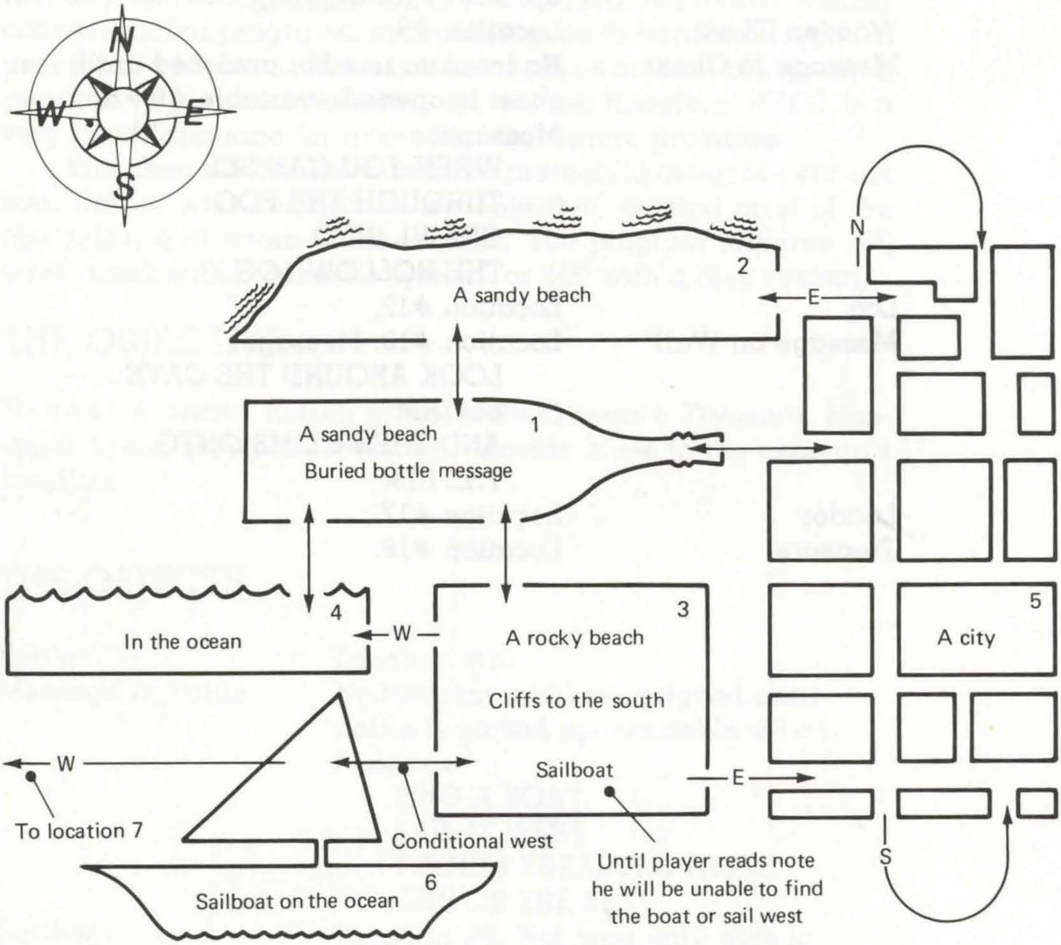
To find the secret hiding place of Blackbeard's Treasure. Messages found along the route will provide clues to the treasure's location.

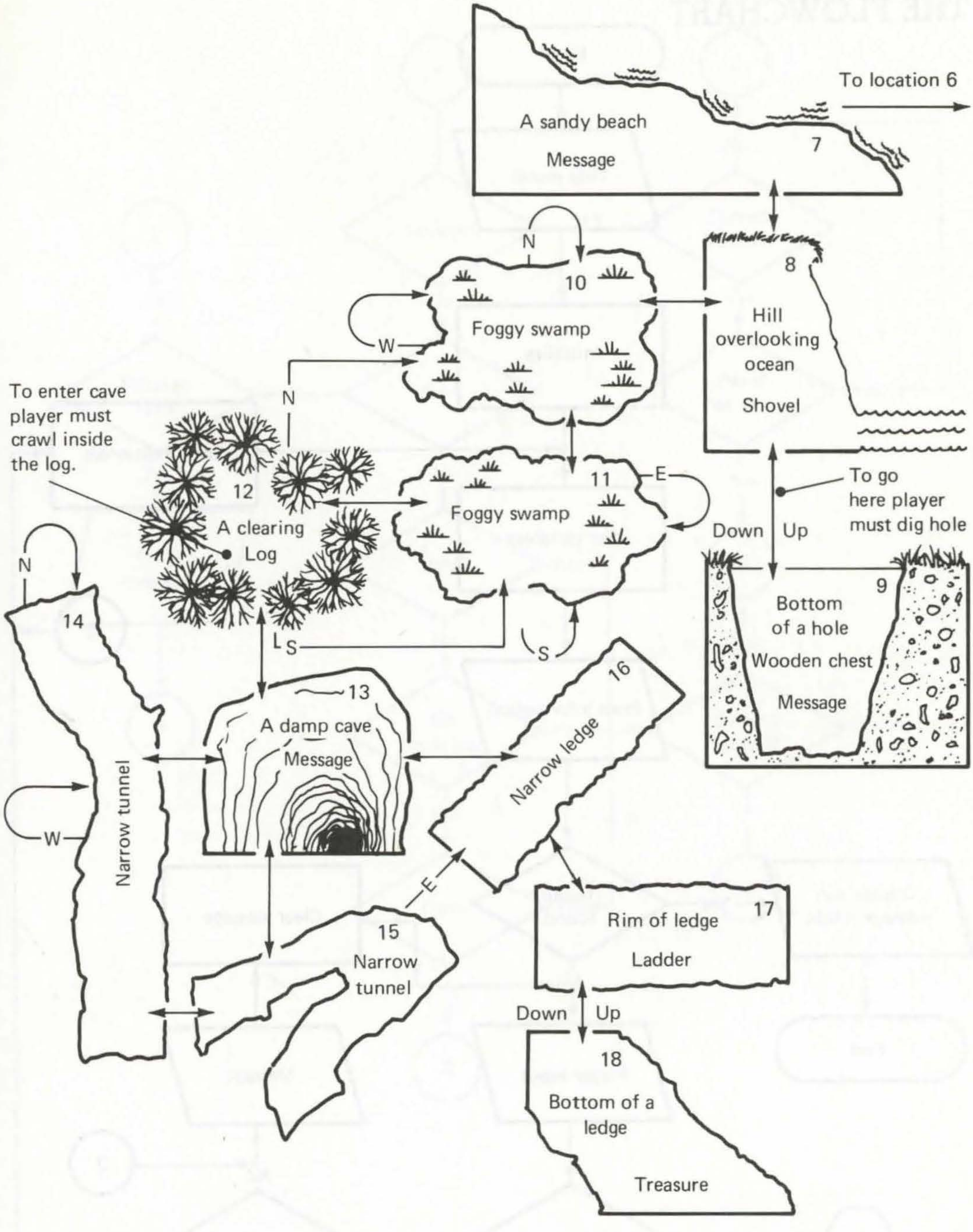
THE OBJECTS

Bottle	Location #1.
Message in Bottle	No location number assigned until bottle is picked up—variable #B = 1. Message: FIND A BOAT SAIL IT WEST THERE'S TREASURE THERE ONE OF THE BEST
Sailboat	Location #3. Not here until note in bottle is read—variable #B more than 1 (#B>1). This keeps the player from going places he is not allowed until after certain conditions have been met.
Rock with Message	Location #7. Message: FIND A SHOVEL DIG IN THE GROUND FOR THERE A CLUE MAY BE FOUND

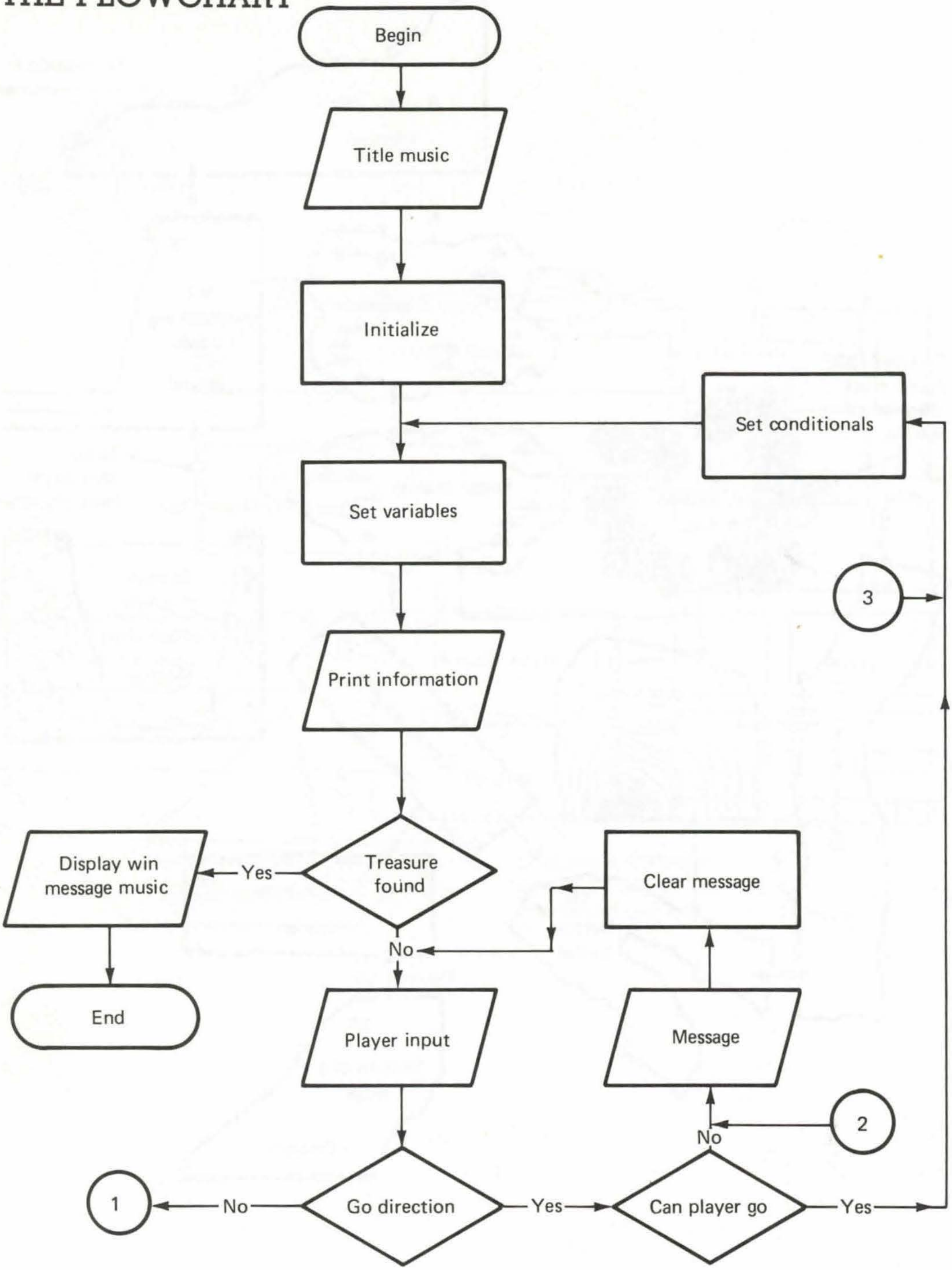
Shovel	Location #8. This item has to be used in order to get to Location #9, which is under ground. #B = 3 for shovel picked up. #B = 4 for digging.
Wooden Chest	Location #9.
Message in Chest	No location number assigned until chest is opened—variable #B = 5. Message: WHEN YOU CAN SEE THROUGH THE FOG CRAWL INTO THE HOLLOW LOG
Log	Location #12.
Message on Wall	Location #13. Message: LOOK AROUND THE CAVE SO DIM AND THEN CLIMB ONTO THE RIM
Ladder	Location #17.
Treasure	Location #18.

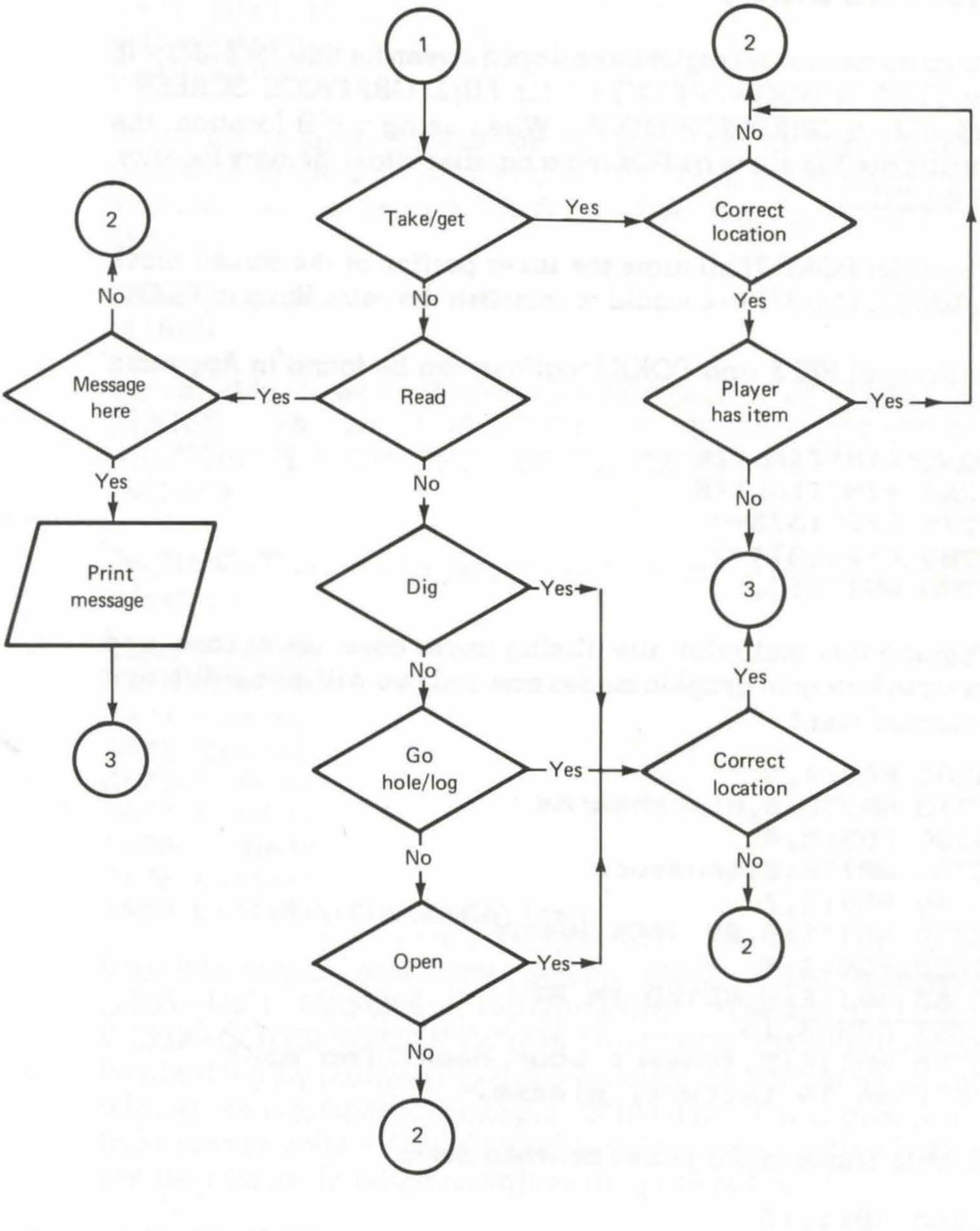
THE MAP


N
W
E
S
2
A sandy beach
E
N
1
A sandy beach
Buried bottle message
4
In the ocean
W
3
A rocky beach
Cliffs to the south
Sailboat
E
5
A city
W
To location 7
Conditional west
Until player reads note he will be unable to find the boat or sail west
6
Sailboat on the ocean
S


To location 6
A sandy beach
Message
7
8
Hill
overlooking
ocean
Shovel
N
10
Foggy swamp
W
N
To enter cave
player must
crawl inside
the log.
12
A clearing
Log
11
Foggy swamp
E
To go
here player
must dig hole
Down
Up
9
Bottom
of a hole
Wooden chest
Message
N
14
S
S
16
13
A damp cave
Message
Narrow ledge
Narrow tunnel
W
E
15
Narrow
tunnel
17
Rim of ledge
Ladder
Down
Up
18
Bottom of a
ledge
Treasure

THE FLOWCHART




1
Take/get
Yes
No
Correct location
No
2
Yes
Player has item
Yes
No
3
Read
Yes
Message here
No
2
Yes
Print message
3
No
Dig
Yes
No
Go hole/log
Yes
Correct location
Yes
3
No
2
No
Open
Yes
No
2

THE PROGRAM

Initialize routine-set registers and open screen for title. [@B1373=16 for TEXT WINDOW/@B1373=0 for FULL GRAPHICS SCREEN / @B1374=n GRAPHICS MODE]. When using a @B location, the results are the same as POKing a number into a memory location in BASIC.

Example: POKE 710,0 turns the inner portion of the screen black in BASIC. C:@B710=0 would accomplish the same thing in PILOT.

Additional PEEK and POKE locations can be found in Appendix B.

```
10 U:*INITIALIZE
3260 *INITIALIZE
3270 C:@B1373=0
3280 C:@B1374=2
3290 WRITE:S,
```

Position text and print title. (Using upper case, lower case, and inverse letters in graphic modes one and two will create different colors of text.)

```
3300 POS:4,3
3310 WRITE:S,blackbeards
3320 POS:5,4
3330 WRITE:S,treasure
3340 POS:3,7
3350 WRITE:S,BY JACK HARDY
3352 POS:4,9
3355 WRITE:S,KEYED IN BY
3357 POS:3,10
3358 WRITE:S,(insert your name) [no mo
re than 14 letters, please.
```

A little music and a pause between notes:

```
3360 SO:1,13
3370 PA:30
3380 SO:2,14
3390 PA:30
3400 SO:4,16
```

```
3410 PA:60
3420 SO:1,13
3430 PA:30
3440 SO:0
```

Pause the program so title can be read. Time in Atari PILOT is measured in units of 1/60th of a second. Thus, PA:60 means stop execution for one second, PA:30 for 1/2 second, PA:180 for 3 seconds, etc.

```
3450 PA:120
(*)RUN
```

Set variables to zero. #B = PROGRESS STATUS / #T = TREASURE STATUS / #N = WHAT IS NORTH OF PRESENT LOCATION / #S = WHAT IS SOUTH OF PRESENT LOCATION, etc. / #D = DOWN / #U = UP.

Set $LOCATION to 1 for player's location at the beginning of the adventure.

```
3460 C:#B=0
3470 C:#T=0
3471 C:#N=0
3472 C:#S=0
3473 C:#E=0
3474 C:#W=0
3475 C:#D=0
3476 C:#U=0
3480 C:$LOCATION=LOCATION1
```

Stop title display and format screen. @B82 = RIGHT MARGIN / @B83 = LEFT MARGIN / @B710 = INNER SCREEN COLOR / @B712 = OUTER SCREEN COLOR / 0 = BLACK. Other colors can be obtained by multiplying the color number (0 to 15) by 16 and adding the luminance. (*Example:* (2*16 + 8 or 40) will give you a light orange color.) Try using other color values in line 3500 to see the effects. Luminance values range from 0 to 14.

```
3490 GR:QUIT
3500 C:@B710=0
3510 C:@B82=0
(*)RUN
```

After running the program, hit SYSTEM RESET to restore default values to screen color and left margin or, in direct mode, you can type C:@B710=128, hit RETURN, C:@B82=2, hit RETURN.

End initialization and return to the beginning of the program.

```
3520 E:
```

First location player will be in:

```
20 *LOCATION1
```

Set variables to the room number player will be in if he goes in that direction. (*Example:* If player goes north from location 1, he or she will be in location 2; if the player goes east, he or she will be in location 5.) Not assigning a value to a direction variable (i.e., leaving it zero) will tell the program there is nothing in that direction. These values will be used later to decide if the player can go in that direction.

```
30 C:#N=2
40 C:#S=3
50 C:#E=5
60 C:#W=4
```

Assign an object to this location. This will be used later to tell the player what items can be seen in this location.

```
70 C:$ITEM=A BOTTLE BURIED IN THE SAND
```

Assign a description to this location. This will be used later to give the player a general description of the area he or she is occupying.

```
100 C:$AREA=ON A SANDY BEACH
```

Now jump to subroutine to print the information onto the screen.

```
110 J:*PRINT
```

The following print routine is used throughout the adventure to relay information to the player. First, it clears the screen and positions the cursor in top left corner.

```
1360 *PRINT
1370 T:[esc ctrl clear]
1380 POS:0,0
```

Then it prints an eye-catching design:

```
1390 T:[40 Inverse %'s / Shift +]
1400 T:
```

Prints a description of the area:

```
1410 T:  YOU ARE $AREA
1420 POS:0,4
1430 T:[20 Inverse ()'s / Shift +]
1440 T:
```

Prints the item variable for this location:

```
1450 T:  You can see:
1460 T:  $ITEM
```

Formats the screen to print directions:

```
1470 POS:0,9
1480 T:[21 Spaces]N
1490 T:[20 Spaces/ctrl FMG/5 spaces/ctrl
Q]U[ctrl E]
1500 T:  POSSIBLE EXITS[3 spaces]W[3 s
paces]E[4 spaces/Shift =/1 space/Shift
 =]
1510 T:[20 spaces/ctrl GMF/5 spaces/ctrl
Z]D[ctrl C]
1520 T:[21 spaces]S
1640 POS:0,15
1650 T:[20 Inverse []'s / Shift +]
1660 T:
```

Prepares for input:

```
1680 T:  WHAT DO YOU WANT TO DO:
1690 T:
1700 T:
(*)RUN
```

The main screen has now been formatted, giving the player some of the necessary information. Add arrows to show the possible directions in which he or she can go.

The commands below are conditional and will only print an arrow if the variable for that direction is more than zero. The equivalent statement in BASIC would be IF/THEN. Example:
IF N>0 THEN PRINT . . .
T(#N>0): . . .

```
1530 POS:21,10
1540 T(#N>0):[esc esc/esc ctrl -]
1550 POS:20,11
1560 T(#W>0):[esc esc/esc ctrl +]
1570 POS:22,11
1580 T(#E>0):[esc esc/esc ctrl *]
1590 POS:21,12
1600 T(#S>0):[esc esc/esc ctrl =]
1610 POS:29,11
1620 T(#U>0):[esc esc/esc ctrl -]
1630 T(#D>0):[esc esc/esc ctrl =]
(*) RUN
```

Jump to input routine, position cursor, accept input, and compare to valid commands.

```
1710 J:*INPUT
1720 *INPUT
1730 T:[esc ctrl - / 2 esc ctrl *'s / shift +]
1740 A:
```

Spaces in the match command (M:) of Atari PILOT are very important. A space before the word means there must be a space before the word on input in order for it to match. *Example:* M: WE matches any word in the input string beginning with the letters WE. Because a space always precedes the beginning of a word in the accept buffer, the combination of a preceding space and the letters WE tell the computer to match any word beginning with the letters WE. *Examples:* WEEK, WEIGHT, WELD, WEST, and even GO WEST would all be considered matches. Likewise, a trailing space would require a space after the word. This way the computer can match an ending letter, word, or fragment of an input string. *Example:* M:EW would match CREW, GREW, FEW, NEW, etc.

```
1750 M: NORTH, SOUTH, EAST, WEST, UP,
DOWN
```

Compare and jump to proper routine on a match:

```
1760 JM:*NORTH,*SOUTH,*EAST,*WEST,*UP,
*DOWN
```

If no match, then compare for other valid commands and jump to proper routine. Note that the commands TAKE and GET will send the program to the same routine.

```
1770 M:TAKE,GET,READ
1780 JM:*TAKE,*TAKE,*READ
```

If no valid commands are matched, ring bell, print message, and jump to a routine called CLEARBOTTOM.

```
1790 T:[esc ctrl 2/2 spaces]I DON'T UNDERSTAND
THAT!
1800 J:*CLEARBOTTOM
```

The CLEARBOTTOM routine does exactly what it says—it pauses for 2 seconds for the message to be read, erases the bottom of the screen, and jumps back to the input routine.

```
3220 *CLEARBOTTOM
3230 PA:120
3240 T:[2 esc ctrl -'s / 7 esc shift inserts]
3250 J:*INPUT
```

```
(*)RUN
```

After running the program, hit RETURN without hitting any other keys. The bell should sound, the I DON'T UNDERSTAND THAT should be displayed, wait to be read, and then disappear. Now the program is waiting for more input.

The direction routines *NORTH, *SOUTH, etc., check to see if direction is equal to zero. If it is (conditional jump), the program jumps to a "can't go in that direction" routine. If more than zero, it sets the value of the direction into a string called $LOCNUM (abbreviation of Location Number) and jumps to a routine called *CHANGELOCATION.

```
1810 *NORTH
1820 J(#N=0):*CANTGO
1830 C:$LOCNUM=#N
1840 J:*CHANGELOCATION
1850 *SOUTH
1860 J(#S=0):*CANTGO
1870 C:$LOCNUM=#S
1880 J:*CHANGELOCATION
1890 *EAST
1900 J(#E=0):*CANTGO
1910 C:$LOCNUM=#E
1920 J:*CHANGELOCATION
1930 *WEST
1940 J(#W=0):*CANTGO
1950 C:$LOCNUM=#W
1960 J:*CHANGELOCATION
1970 *UP
1980 J(#U=0):*CANTGO
1990 C:$LOCNUM=#U
2000 J:*CHANGELOCATION
2010 *DOWN
2020 J(#D=0):*CANTGO
2030 C:$LOCNUM=#D
2040 J:*CHANGELOCATION
```

If nothing in that direction, the routine rings the bell, displays the message, and jumps to the CLEARBOTTOM routine.

Note: The program comes here only if the value of the directional variable is equal to zero.

```
2170 *CANTGO
2180 T:[esc ctrl 2/2 spaces]YOU CAN'T
GO IN THAT DIRECTION!
2190 J:*CLEARBOTTOM
```

```
(*)RUN
```

Type GO UP and hit RETURN. If all is well, a bell will sound, a message will be displayed, and the bottom of the screen will clear. Remember, this is Location 1 and there is no way to go up from here. In order to have some new places to go, more locations will have to be added to the program:

```
120 *LOCATION2
130 C:#S=1
140 C:#E=5
150 C:$ITEM=NOTHING SPECIAL
160 C:$AREA=ON A SANDY BEACH
170 J:*PRINT
200 *LOCATION3
220 C:#N=1
230 C:#E=5
235 C:#W=4
240 C:$ITEM=NOTHING OF INTEREST
260 C:$AREA=ON A ROCKY BEACH. THERE AR
E CLIFFS TO THE SOUTH.
270 J:*PRINT
280 *LOCATION4
290 C:#W=4
300 C:#N=4
310 C:#S=4
320 C:#E=1
330 C:$ITEM=LOTS OF WATER
340 C:$AREA=SWIMMING IN THE OCEAN
350 J:*PRINT
360 *LOCATION5
370 C:#N=5
380 C:#S=5
390 C:#W=1
400 C:$AREA=IN A LARGE CITY
410 C:$ITEM=MANY BUILDINGS
420 J:*PRINT
```

Note: In Location 4, going in any direction other than east will keep the player swimming in the ocean. Likewise, going north or south from Location 5 will keep the player in the city.

Now give the program a way to get to all the new locations. Reset all the directional variables to zero to prepare the program for the new location's directional variables:

```
3000 *CHANGELOCATION
3010 C:#N=0
3020 C:#S=0
3030 C:#E=0
3040 C:#W=0
```

```
3050 C:#U=0
3060 C:#D=0
```

Append the $LOCNUM to LOCATION for a new location:

```
3070 C:$LOCATION=LOCATION$LOCNUM
```

The following part of the program tells the computer to accept the new location, match the last three letters of the new location, and jump to them when it finds a match. Upon arriving at the new location, all new information will be assigned to direction variables and to the strings that hold the description of the area and the items seen there:

```
3080 A:=$LOCATION
3090 M:ON1 ,ON2 ,ON3 ,ON4 ,ON5 ,
3100 JM:*LOCATION1,*LOCATION2,*LOCATIO
N3,*LOCATION4,*LOCATION5
```

The player can move from area to area by typing GO NORTH, etc., or by typing only a direction (i.e., NORTH, SOUTH, etc.).

```
(*)RUN
```

Now a routine to pick up the bottle needs to be created. There are several things the program must check before the player is allowed to do this. The first is if the player is in the same location as the bottle:

```
2200 *TAKE
2210 A:=$LOCATION
2220 M:LOCATION1
2230 JY:*TAKE1
2240 T:[esc ctrl 2 / 2 spaces]THERE IS
 NOTHING HERE TO TAKE!
2250 J:*CLEARBOTTOM
```

Second, did the player already pick up the bottle? *Note:* This is done by checking the #B status variable. If the bottle has been picked up, #B will be equal to 1 and the program will jump to a routine to tell the player so. If not, it will set the #B status variable to 1, tell the player OK (to pick the bottle up), and return to

Location 1 to acquire updated information.

```
2260 *TAKE1
2270 J(#B<>0):*HAVE
2280 C:#B=1
2290 U:*OK
2300 J:*LOCATION1
2930 *HAVE
2940 T:  YOU ALREADY HAVE!
2950 J:*CLEARBOTTOM
2960 *OK
2970 T:  OKAY!
2980 PA:60
2990 E:
```

Since the bottle can now be picked up in Location 1, the message there will have to be changed from A BOTTLE BURIED IN THE SAND to something else. Add the following line:

```
80 C(#B=1):$ITEM=A MESSAGE IN THE BOTTLE
```

Note: The above is another conditional command. Since the program is setting #B to equal 1 when the bottle is picked up, this will replace the old message in line 70. View this part of the program by typing in direct mode, LIST 70,80 and hitting RETURN. Now you can see one more way the powerful conditional command can be used.

```
(*) RUN
```

Now that the bottle can be picked up, it can be seen that there is a message to be read inside. The following routines are needed to read the message. Hit SYSTEM RESET and add the following:

```
2355 *READ
2360 J(#B=1):*MESSAGE1
2420 J:*CANTREAD
```

Note: Again the program will check #B to see if the bottle has been picked up. If so, it will jump to message 1 routine. If not, it will jump to "can't read" routine, which sounds the bell, displays the message, and jumps again to the oft-used CLEARBOTTOM routine:

```
2870 *CANTREAD
2880 T:[esc ctrl 2 / 2 spaces]THERE IS
 NOTHING HERE TO READ!
2890 J:*CLEARBOTTOM
```

The information on the note can be read only once, as the program sets the #B status variable to two, prints the message, and jumps to the reset routine:

```
2430 *MESSAGE1
2440 U:*OK
2450 C:#B=2
2455 T:[2 esc ctrl -'s]
2460 T:   FIND A BOAT
2470 T:    SAIL IT WEST
2480 T:     THERE'S A TREASURE THERE
2490 T:      ONE OF THE BEST
2500 J:*RESET
```

The reset routine pauses so the message can be read, repositions the cursor, and jumps to the CLEARBOTTOM routine:

```
3180 *RESET
3190 PA:240
3200 T:[3 esc ctrl -'s]
3210 J:*CLEARBOTTOM
```

What if the player wanted to read the note again? There will have to be a routine created to explain why the note can no longer be read:

```
2885 T(#B=2):  THE NOTE TURNED TO DUST!
[esc ctrl -]
```

The note must also be disposed of in Location 1:

```
90 C(#B=2):$ITEM=NOTHING UNUSUAL
```

If the message has been read, add the sailboat in Location 3 and set direction so player can go west:

```
250 C(#B=2):$ITEM=A SMALL SAILBOAT
255 C(#B=2):#W=6
```

Since sailing west is now possible, more locations are needed:

```
440 *LOCATION6
450 C:#N=6
460 C:#S=6
470 C:#E=3
480 C:#W=7
490 C:$AREA=IN A SMALL SAILBOAT ON THE OCEAN
500 C:$ITEM=LOTS OF WATER
510 J:*PRINT
520 *LOCATION7
530 C:#S=8
540 C:#E=6
550 C:$AREA=ON A SANDY BEACH
560 C:$ITEM=A LARGE ROCK. THERE IS A MESSAGE
ON THE ROCK.
570 J:*PRINT
```

A way to get to these new locations is needed now. Change lines 3090 and 3100. *Note:* The method of leaving a space at the end of the match statement for checking only the last three letters of the information accepted by the input buffer is again used:

```
3090 M:ON1 ,ON2 ,ON3 ,ON4 ,ON5 ,ON6 ,O
N7 ,
3100 JM:*LOCATION1,*LOCATION2,*LOCATIO
N3,*LOCATION4,*LOCATION5,*LOCATION6,*L
OCATION7
```

Use the read routine in line 2355 to read Message 2. Let the program input the location, match, and jump to the proper message. If the player is in the right location, have the program use the OK routine, position the cursor, and jump to Message 2:

```
2370 A:=$LOCATION
2380 M:ON7 ,
2390 UY:*OK
2400 TY:[3 esc ctrl -'s]
2410 JM:*MESSAGE2
```

The program displays the message then jumps to the reset routine:

```
2510 *MESSAGE2
2520 T:   FIND A SHOVEL
2530 T:    DIG IN THE GROUND
2540 T:     FOR THERE A CLUE
2550 T:      MAY BE FOUND
2560 J:*RESET
```

```
(*)RUN
```

Note: Try reading things in the wrong location, going in the wrong direction, taking things that are not there, etc. All this will help you understand how the program works.

Add some more locations:

```
580 *LOCATION8
590 C:#W=10
610 C:#N=7
620 C:$AREA=ON A HILL OVERLOOKING THE OCEAN
630 C:$ITEM=A SHOVEL
660 J:*PRINT
670 *LOCATION9
680 C:#U=8
690 C:$AREA=IN THE BOTTOM OF A HOLE
700 C:$ITEM=A WOODEN CHEST
720 J:*PRINT
```

and a way to get to them:

```
3120 M:ON8 ,ON9 ,
3130 JM:*LOCATION8,*LOCATION9
```

Note: Location 9 is in the bottom of a hole and there is no way to get to it. Therefore, the command *DIG needs to be added to the program's growing vocabulary. Change line 1770 and 1780 as shown:

```
1770 M:TAKE,GET,READ,DIG
1780 JM:*TAKE,*TAKE,*READ,*DIG
```

Also needed is a way to take the shovel. Change lines 2220 and 2230 to the following:

```
2220 M:LOCATION1,LOCATION8
2230 JM:*TAKE1,*TAKE2
```

Again the program goes through the verification routines. First, it checks to see if player is in the same location as the shovel; second, it makes sure the shovel has not already been picked up. If it has not, it sets the #B status variable to three, gives the shovel to the player, says OK, and jumps back to Location 8 so information can be updated:

```
2310 *TAKE2
2320 J(#B>2):*HAVE
2330 C:#B=3
2340 U:*OK
2350 J:*LOCATION8
```

In order to get the shovel off the ground, a conditional routine must be added to Location 8 to remove the shovel and place it in the hands of the player:

```
640 C(#B=3):$ITEM=A SHOVEL IN YOUR HANDS.
```

Now create a way to use the new *DIG word in the vocabulary. First, the program checks to make sure player is in Location 8 (as this is the only place a hole can be dug); second, it makes sure the shovel has been picked up; and third, it checks to see if the player has already dug a hole. If the player has met these requirements, it sets #B status variable to four, says OK, and jumps back to Location 8 to see the hole:

```
2780 *DIG
2790 A:=$LOCATION
2800 M:LOCATION8
2810 JN:*CANTDO
2820 J(#B<3):*CANTDO
2830 J(#B>=4):*HAVE
2840 C:#B=4
2850 U:*OK
2860 J:*LOCATION8
```

If player did not have the shovel, the program comes here:

```
2900 *CANTDO
2910 T:[esc ctrl 2 / 2 spaces]YOU CAN'T DO
THAT!
2920 J:*CLEARBOTTOM
```

When a hole has been dug in Location 8, the program changes its description to show a hole and sets a directional variable for a way to go into it:

```
650 C(#B>3):$ITEM=A HOLE IN THE GROUND
655 C(#B>3):#D=9

(*)RUN
```

Now there is a way to get to Location 9 by typing GO DOWN (or just DOWN). But what if the player wanted to say GO HOLE instead? That would require the addition of a new word to the program's vocabulary as well as routines to use the new word:

```
1770 M:TAKE,GET,READ,DIG,HOLE
1780 JM:*TAKE,*TAKE,*READ,*DIG,*HOLE

2110 *HOLE
2120 A:=$LOCATION
2130 M:LOCATION8
2140 JN:*CANTDO
2150 J:*DOWN
```

A routine is now needed to open the chest at the bottom of the hole, so another word is added to the program's vocabulary as well as a routine to use the word:

```
1770 M:TAKE,GET,READ,DIG,HOLE,OPEN
1780 JM:*TAKE,*TAKE,*READ,*DIG,*HOLE,*
OPEN

2700 *OPEN
2710 A:=$LOCATION
2720 M:LOCATION9
2730 JN:*CANTDO
2740 J(#B<>4):*HAVE
2750 C:#B=5
2760 U:*OK
2770 J:*LOCATION9
```

Add a conditional command to Location 9 to show what was in the chest:

```
710 C(#B=5):$ITEM=AN OPEN WOODEN CHEST
 WITH A MESSAGE CARVED ON THE BOTTOM
```

Add another message to the read routine:

```
2380 M:ON7 ,ON9
2410 JM:*MESSAGE2,*MESSAGE3

2570 *MESSAGE3
2580 J(#B<5):*CANTREAD
2590 T:   WHEN YOU CAN SEE
2600 T:    THROUGH THE FOG
2610 T:     CRAWL INTO
2620 T:      THE HOLLOW LOG
2630 J:*RESET

(*)RUN
```

Now to add more locations. Note that the player can easily become lost in the foggy swamps (Locations 10 and 11), as they are identical in description and each sets some of the directional variables to its own location number. The player can wander around for quite a while in the swamp if he or she doesn't go in the right directions.

```
730 *LOCATION10
740 C:#N=10
750 C:#S=11
760 C:#E=8
770 C:#W=10
780 C:$AREA=IN A FOGGY SWAMP
790 C:$ITEM=NOTHING
800 J:*PRINT
810 *LOCATION11
820 C:#N=10
830 C:#E=11
840 C:#S=11
850 C:#2=12
860 C:$AREA=IN A FOGGY SWAMP
870 C:$ITEM=NOTHING
```

```
880 J:*PRINT
890 *LOCATION12
900 C:#N=10
910 C:#W=10
920 C:#E=11
930 C:#S=11
940 C:$AREA=IN A CLEARING.
950 C:$ITEM=A LOG
960 J:*PRINT
970 *LOCATION13
980 C:#N=12
990 C:#S=15
1000 C:#E=16
1010 C:#W=14
1020 C:$AREA=IN A DAMP CAVE.
1030 C:$ITEM=A MESSAGE ON THE WALL
1040 J:*PRINT
```

Now a routine must be added to get to the new locations by changing lines 3120 and 3130:

```
3120 M:ON8 ,ON9 ,N10 ,N11 ,N12 ,N13 ,
3130 JM:*LOCATION8,*LOCATION9,*LOCATIO
N10,*LOCATION11,*LOCATION12,*LOCATION1
3
```

The only way to get to Location 13 is through the log, so another word must be added to the program's vocabulary and a routine to use it.

```
1770 M:TAKE,GET,READ,DIG,HOLE,OPEN,LOG
1780 JM:*TAKE,*TAKE,*READ,*DIG,*HOLE,*
OPEN,*LOG

2050 *LOG
2060 A:=$LOCATION
2070 M:LOCATION12
2080 JN:*CANTDO
2090 C:$LOCNUM=13
2100 J:*CHANGELOCATION

(*)RUN
```

Add the message found in Location 13:

```
2380 M:ON7 ,ON9 ,N13 ,
2410 JM:*MESSAGE2,*MESSAGE3,*MESSAGE4

2640 *MESSAGE4
2650 T:   LOOK AROUND THE CAVE
2660 T:    SO DIM
2670 T:     AND THEN CLIMB ONTO
2680 T:      THE RIM
2690 J:*RESET
```

Add the final locations, all in the cave:

```
1050 *LOCATION14
1060 C:#N=14
1070 C:#S=15
1080 C:#E=13
1090 C:$AREA=IN A NARROW TUNNEL
1100 C:$ITEM=NOTHING
1110 J:*PRINT
1120 *LOCATION15
1130 C:#N=13
1140 C:#E=16
1150 C:#W=14
1160 C:$AREA=IN A NARROW TUNNEL
1170 C:$ITEM=NOTHING
1180 J:*PRINT
1190 *LOCATION16
1200 C:#W=13
1210 C:#S=17
1220 C:$AREA=ON A NARROW LEDGE
1230 C:$ITEM=WATER RUNNING DOWN THE WALLS.
1240 J:*PRINT
1250 *LOCATION17
1260 C:#D=18
1270 C:#N=16
1280 C:$AREA=ON THE RIM OF A LEDGE
1290 C:$ITEM=LADDER GOING DOWN
1300 J:*PRINT
1310 *LOCATION18
1320 C:$AREA=AT THE BOTTOM OF A LEDGE
1330 C:$ITEM=[inverse] ** THE TREASURE **
```

```
[inverse off]
1340 C:#T=1
1350 J:*PRINT
```

And a way to get to each one:

```
3150 M:N14 ,N15 ,N16 ,N17 ,N18 ,
3160 JM:*LOCATION14,*LOCATION15,*LOCAT
TION16,*LOCATION17,*LOCATION18
```

Note: In Location 18, the treasure status is set to one. Add the following line to the print routine:

```
1670 J(#T=1):*WIN
```

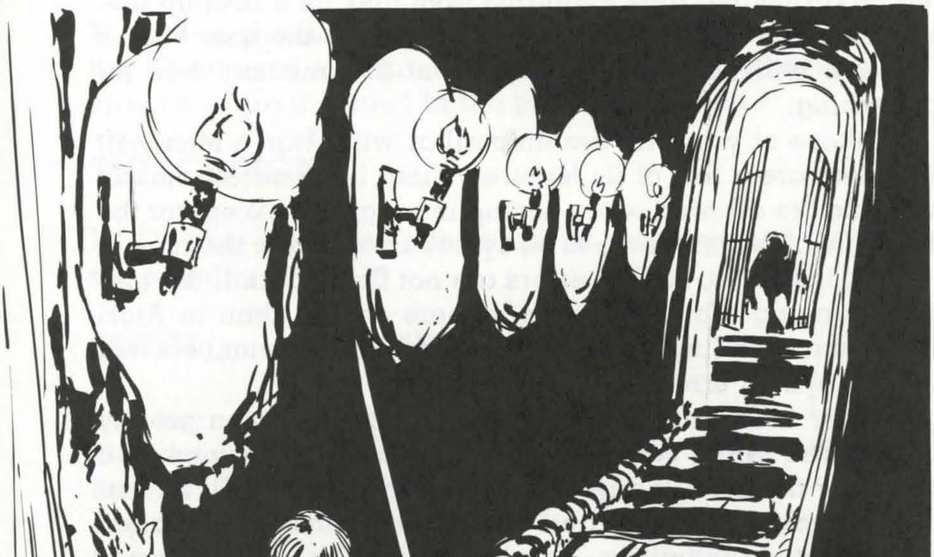
Add the win routine. The program will now play random musical notes and flash the winning message on the screen ten times:

```
3540 *WIN
3550 POS:9,20
3560 T:***YOU FOUND IT!***[***'s in inverse]
3570 C:#T=#T+1
3580 C:#A=?\30
3590 SO:#A
3600 PA:10
3610 POS:9,20
3620 T:***YOU FOUND IT!***[YOU FOUND IT! in
inverse]
3630 C:#A=?\30
3640 SO:#A
3650 PA:10
3660 J(#T<10):*WIN
3670 E:
```

This completes the first program. In the process of entering it, you have had the fun of watching it grow, segment by segment, into its final form. Type LIST, hit RETURN. As you can see, all of the line numbers are now in order. If you would like to clean up the listing and have the program lines incremented by ten, then type REN 10,10 and hit RETURN. List the program again and be sure to save a copy of all work onto cassette (CSAVE) or disk (SAVE "D:filename").

AN ADVENTURE IN ATARI
MICROSOFT BASIC:

HALLS OF DEATH



Halls of Death is written in Atari Microsoft BASIC, a full implementation of the standard BASIC used on the Apple, the Pet, the TRS-80, and many other popular microcomputers. In order to take advantage of all of the capabilities of the Atari, such as sound, player-missile graphics, etc., it was necessary for Microsoft to write a very large language. It occupies almost 20K from disk. There is also a 16K version available on ROM cartridge, with the added programming aids available from disk as a boot-up file. When using the complete Microsoft language, the user with a 48K system will have only a little over 21K memory free for programming.

For those of you who are unfamiliar with Atari's Microsoft BASIC, here are a few of its features. First, in Microsoft, no abbreviations are allowed when keying in programs except for the "?" for PRINT. Second, there is no syntax checking at the time of line entry; therefore, keying errors are not flagged until the program is running. Third, spacing is more critical than in Atari BASIC, because failure to separate statements from numbers will result in a syntax error.

Now for the positive side: I/O flexibility has been greatly improved. The INPUT command will let you substitute your own message for the default question mark. LINE INPUT will let you input a complete sentence and will ignore commas, quotation marks, and other delimiters. This is excellent for interactive adventure games because of the huge and continuous amount of user input required. The execution speed is incredibly fast, much faster than Atari BASIC. A comparison between the length of delay loops required in Atari Microsoft BASIC and Atari BASIC will substantiate this.

The storing of information in Atari Microsoft is a very simple task, since strings can be dimensioned to hold all vocabulary words [e.g., dimensioning VOC$(20) would assign 20 different strings to the variable VOC$. VOC$(1) could equal TAKE, VOC$(2) could equal LOOK, etc.]. All item words, descriptions of all rooms, and all changing variable descriptions can also be stored in strings assigned to them.

Halls of Death was designed with the aid of my good friend, Dave Simpson, who is responsible for introducing me to fantasy role-playing games. The program requires 32K with cassette or 48K with disk drive. You will note that many of the messages the program relays are encrypted to make the game more fun to play. Although they are printed in readable form for the player, anyone keying in the program will have less understanding of what is happening and therefore have more fun when the adventure can finally be played.

THE OBJECTIVE

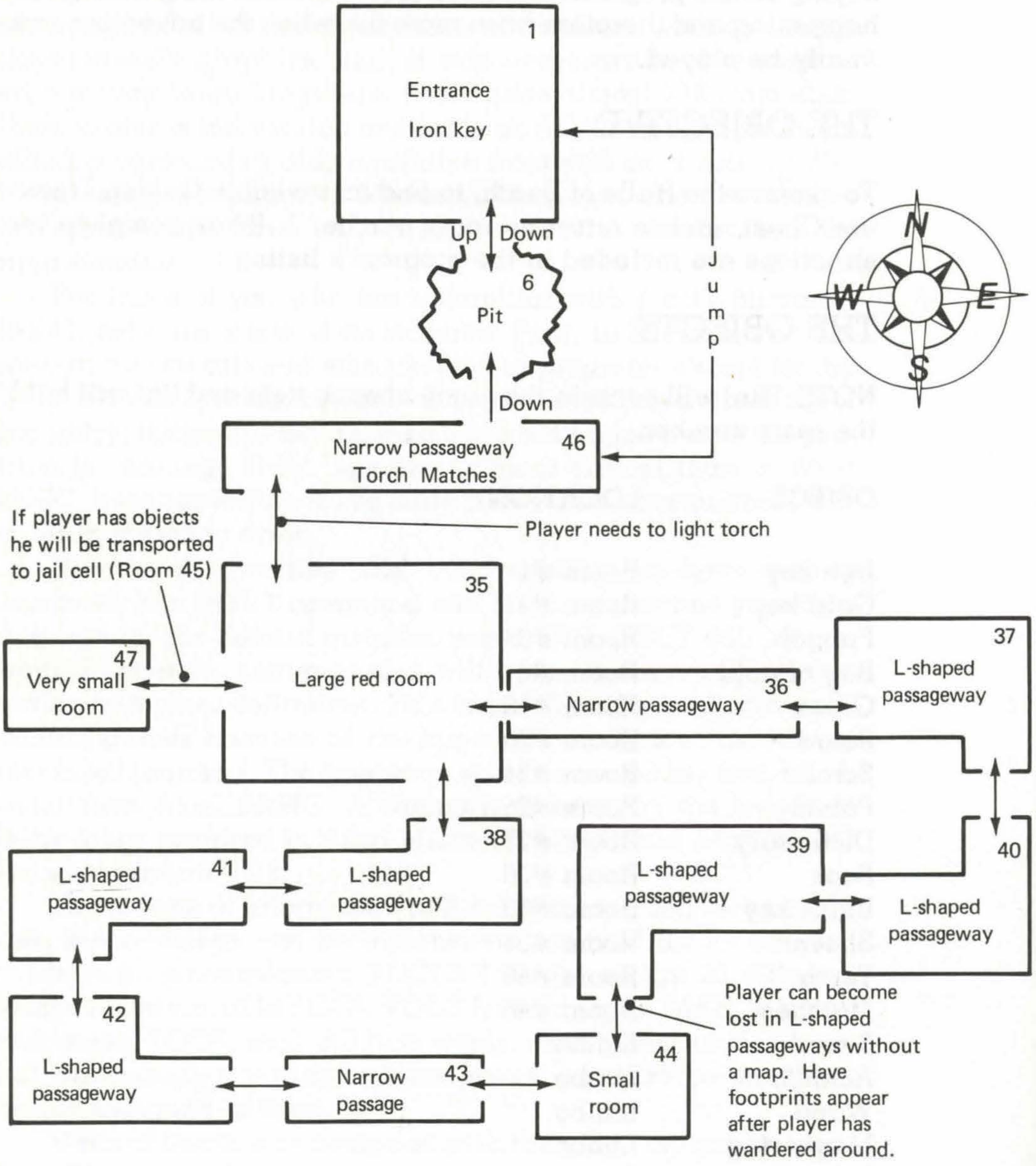
To explore the Halls of Death, to find an invisible Golden Treasure Chest, and to return to room number 1. (More complete instructions are included in the program's listing.)

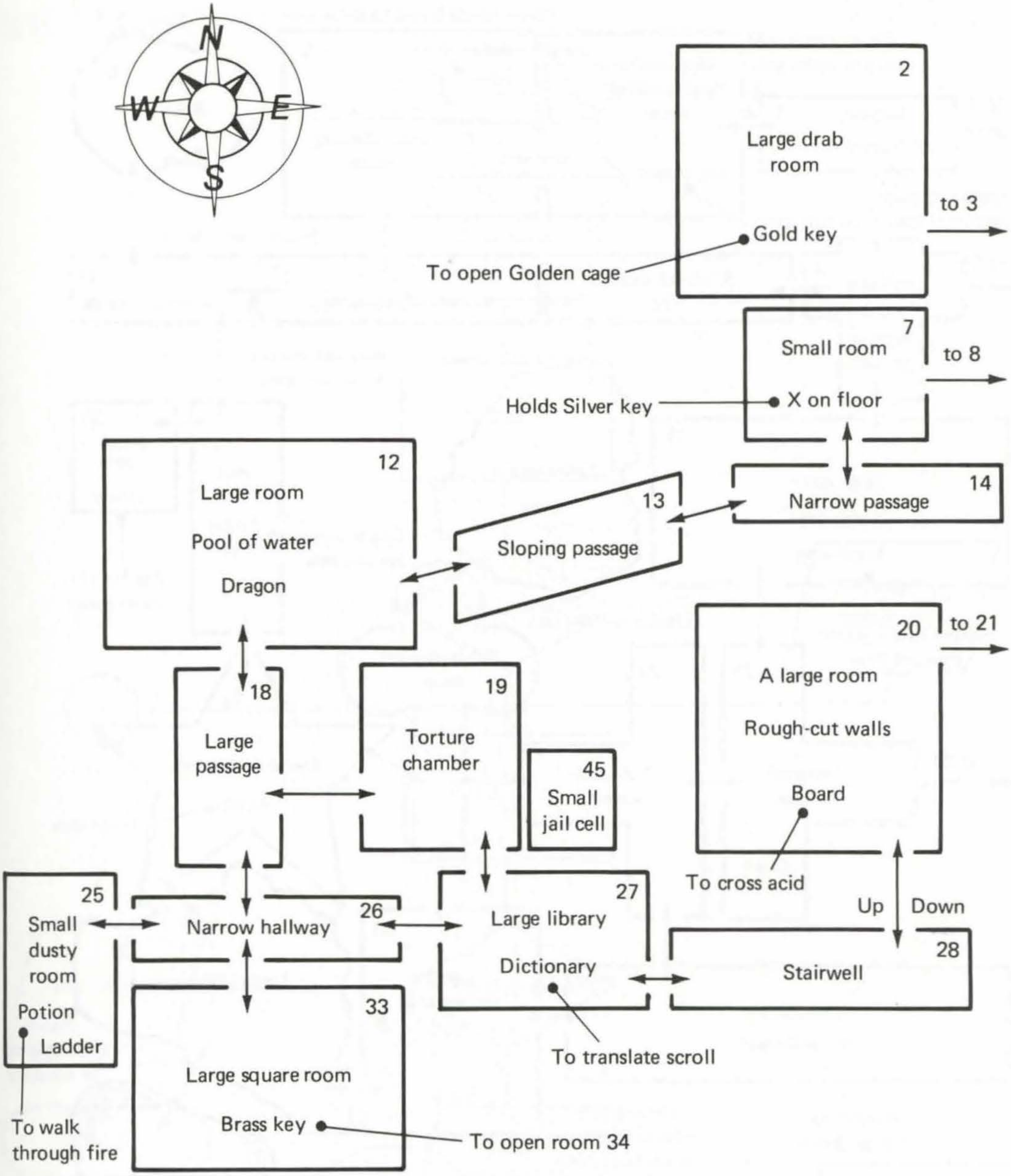
THE OBJECTS

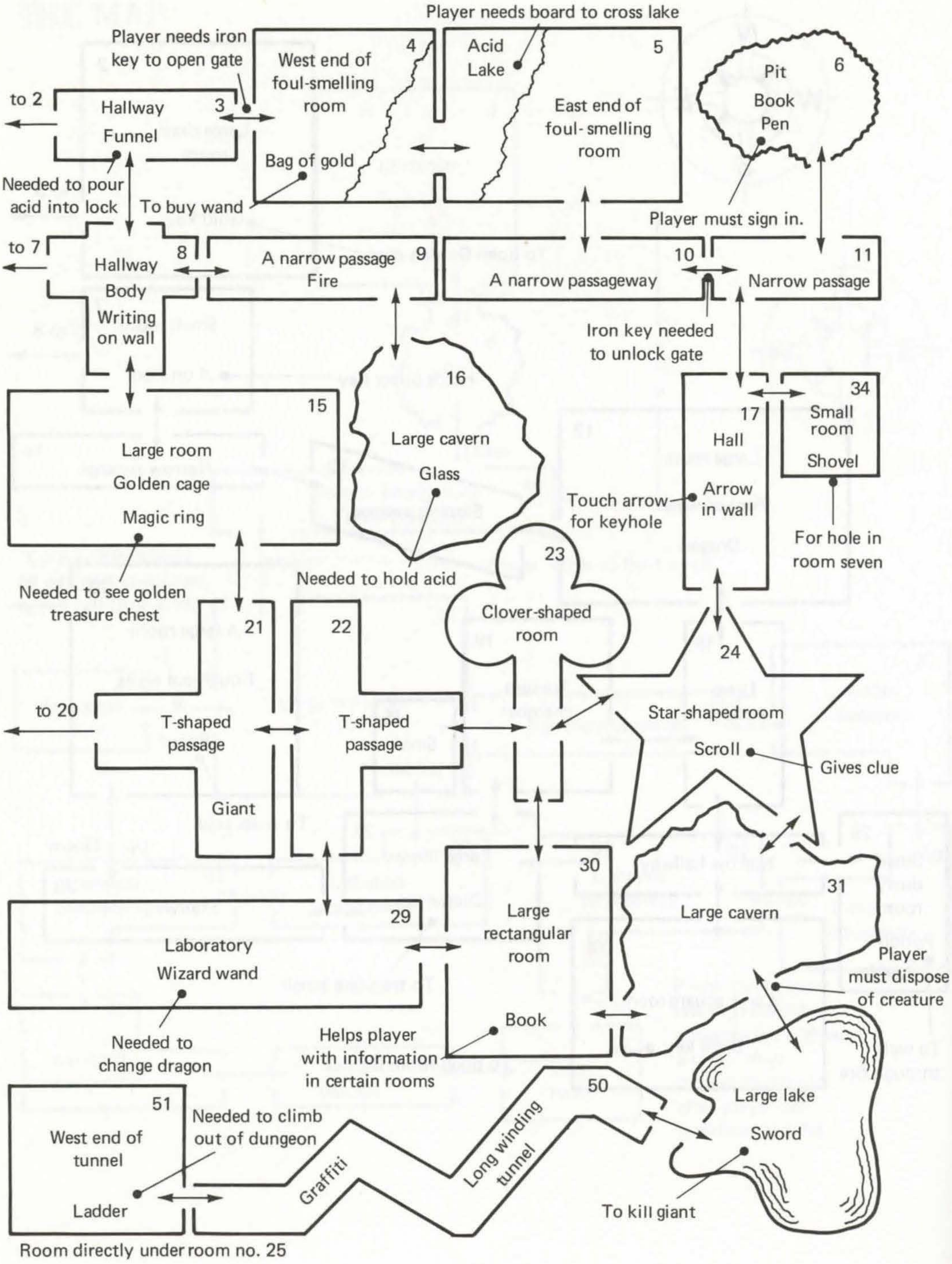
NOTE: I$(n) will contain the name of each item and I(n) will hold the room number.

OBJECT	**LOCATION**
Iron key	Room #1
Gold key	Room #2
Funnel	Room #3
Bag of gold	Room #4
Glass	Room #16
Board	Room #20
Scroll	Room #24
Potion	Room #25
Dictionary	Room #27
Book	Room #30
Brass key	Room #33
Shovel	Room #34
Torch	Room #46
Matches	Room #46
Sword	Limbo
Acid (2)	Limbo
Wand	Limbo
Magic ring	Limbo
Silver key	Limbo
Ladder	Limbo
Fish	Limbo

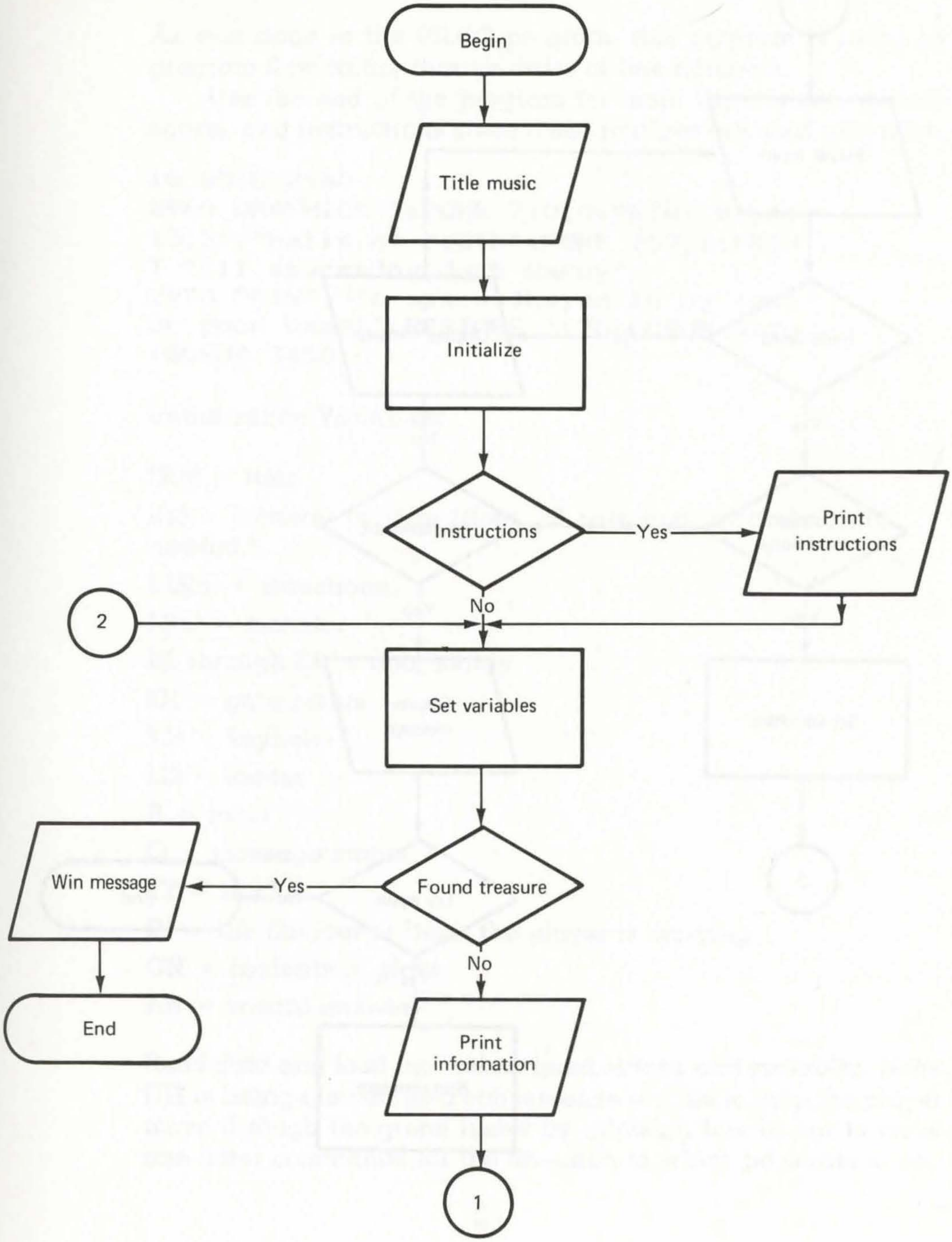
THE MAP

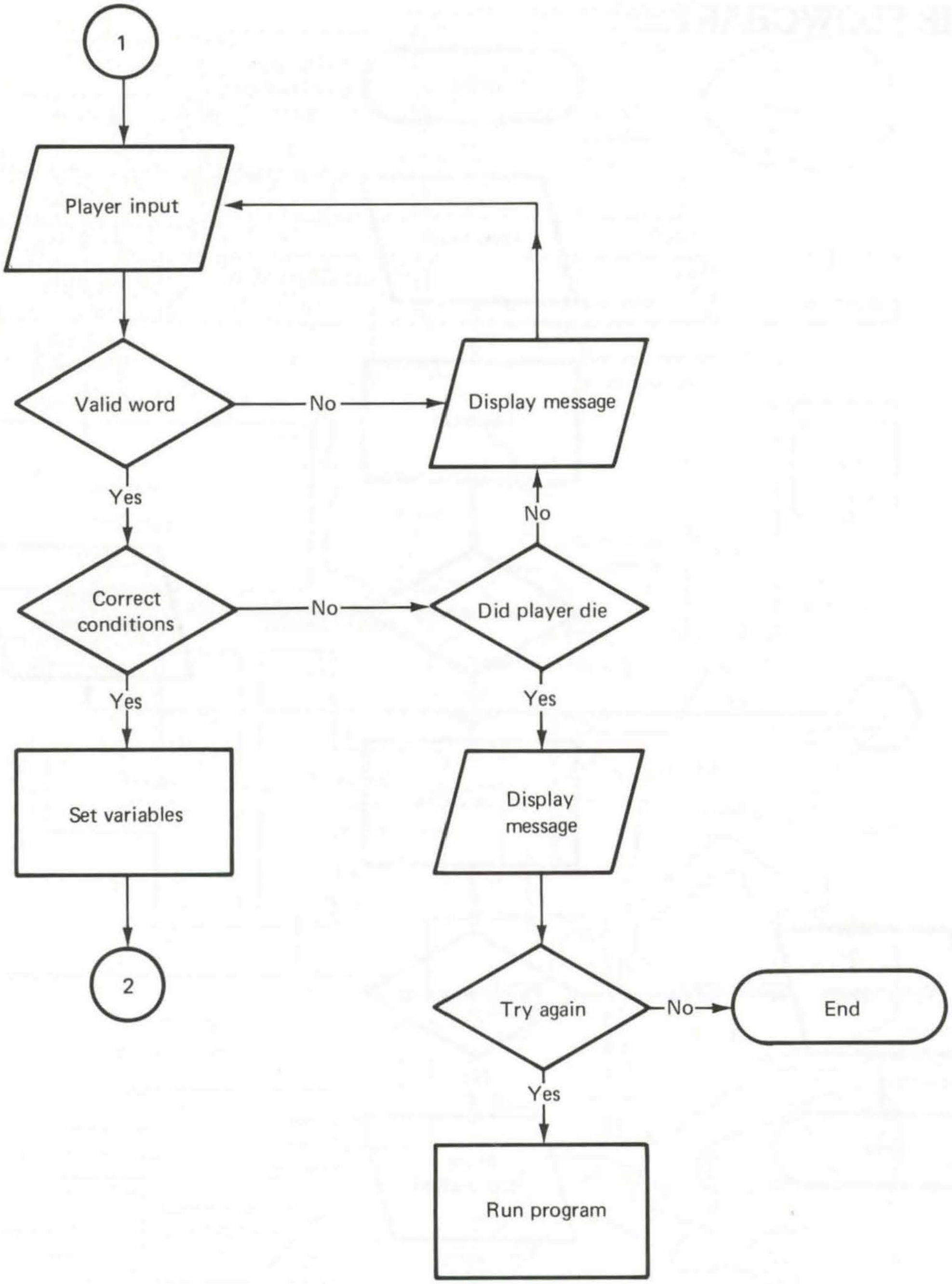

1
Entrance
Iron key
Up
Down
6
Pit
Down
J
u
m
p
N
W
E
S
Narrow passageway
Torch Matches
46
Player needs to light torch
If player has objects
he will be transported
to jail cell (Room 45)
35
Large red room
47
Very small
room
36
Narrow passageway
37
L-shaped
passageway
38
L-shaped
passageway
41
L-shaped
passageway
39
L-shaped
passageway
40
L-shaped
passageway
42
L-shaped
passageway
Narrow
passage
43
44
Small
room
Player can become
lost in L-shaped
passageways without
a map. Have
footprints appear
after player has
wandered around.


N
W
E
S
2
Large drab room
Gold key
To open Golden cage
to 3
7
Small room
X on floor
Holds Silver key
to 8
14
Narrow passage
13
Sloping passage
12
Large room
Pool of water
Dragon
20
A large room
Rough-cut walls
Board
To cross acid
to 21
18
Large passage
19
Torture chamber
45
Small jail cell
25
Small dusty room
Potion
Ladder
To walk through fire
26
Narrow hallway
27
Large library
Dictionary
To translate scroll
Up
Down
28
Stairwell
33
Large square room
Brass key
To open room 34


Player needs board to cross lake
Player needs iron key to open gate
4
West end of foul-smelling room
Acid Lake
5
East end of foul-smelling room
Pit
6
Book
Pen
to 2
Hallway
3
Funnel
Bag of gold
Needed to pour acid into lock
To buy wand
Player must sign in.
to 7
Hallway
8
Body
Writing on wall
A narrow passage
9
Fire
A narrow passageway
10
11
Narrow passage
Iron key needed to unlock gate
16
Large cavern
Glass
15
Large room
Golden cage
Magic ring
34
Small room
Shovel
17
Hall
Touch arrow for keyhole
Arrow in wall
For hole in room seven
23
Needed to hold acid
Needed to see golden treasure chest
21
22
Clover-shaped room
24
to 20
T-shaped passage
T-shaped passage
Star-shaped room
Scroll
Gives clue
Giant
30
31
29
Large rectangular room
Large cavern
Laboratory
Wizard wand
Player must dispose of creature
Needed to change dragon
Book
Helps player with information in certain rooms
50
Large lake
51
Needed to climb out of dungeon
Long winding tunnel
West end of tunnel
Sword
Graffiti
Ladder
To kill giant
Room directly under room no. 25

THE FLOWCHART




1
Player input
Valid word
No
Display message
Yes
No
Correct conditions
No
Did player die
Yes
Yes
Set variables
Display message
2
Try again
No
End
Yes
Run program

THE PROGRAM

As was done in the PILOT program, this program is listed by program flow rather than in order of line numbers.

Use the end of the program for main title, music, initialization, and instructions since these routines are used only once.

```
10 GOTO 2960
2960 GRAPHICS 2:POKE 710,0:PRINT #6,AT
(3,3);"halls of death":POKE 752,1:PRIN
T "[11 spaces]by Jack Hardy"
2970 PRINT "[6 spaces]Keyed in by (ent
er your name)":RESTORE 3170:GOSUB 3120
:GOSUB 3450
```

Initialization Variables:

I$(n) = item

I(n) = location of item (*Note:* 99 sets item in limbo until needed.)

DI$(n) = directions

M(n) = monster

D1 through D8 = door status

G1 = gate status

KH = keyhole

LD = ladder

R = room

Q = message status

TT = tracks

IN = the number of items the player is carrying

CN = contents of glass

AN = wizard answer

Read data and load into subscripted strings and variables. *Note:* DI$ is being created as a convenience routine to help the player move through the game faster by allowing him or her to issue one-letter commands for the direction in which he wants to go.

```
3120 DIM I$(23),I(23),DI$(6),M(3):D1=0
:D2=0:D3=0:D4=0:D5=0:D6=0:D7=0:D8=0:G1
=0:KH=0:LD=0
3130 RANDOMIZE:FOR T=1 TO 3:M(T)=1:NEX
T:CN=0:R=1:TT=0:IN=0:Q=0
3140 FOR T=1 TO 22:READ I$(T):READ I(T
):NEXT
3150 FOR T=1 TO 6:READ DI$(T):NEXT
3160 CL$="[esc ctrl -][39 spaces]":CL1
$="[2 esc ctrl -'s][38 spaces]":RETURN
3170 DATA SEHCTAM,46,LEVOHS,34,LLORCS,
24,YEK NORI,1,DROWS,99,DICA,99,DLOG FO
GAB,4,DICA,99
3180 DATA KOOB,30,SSALG,16,DNAW,99,LEN
NUF,3,GNIR CIGAM,99,HCROT,46,YEK DLOG,
2,YEK REVLIS,99
3190 DATA YRANOITCID,27,NOITOP,25,REDD
AL,99,YEK SSARB,33,DRAOB,20,HSIF,99,N,
S,E,W,U,D
```

Subroutine to play the death march:

```
3450 RESTORE 3480
3460 READ A,B:IF A>1 THEN SOUND 0,A,10
,8:FOR I=1 TO B:NEXT:SOUND 0,0,0,0:FOR
I=1 TO 40:NEXT:GOTO 3460
3470 RETURN
3480 DATA 100,360,100,240,100,60,100,3
60,85,480,90,360,100,240,105,180,100,3
60,0,0
```

```
(*) RUN
```

Ask if instructions are needed. Read the keyboard and send to proper routine. If no key or wrong key is being pressed, continue looping:

```
2980 PRINT #6,AT(1,7);"WANT INSTRUCTIO
NS?[9 spaces](Y/N)"
2990 A$=INKEY$:IF A$="Y" THEN 3020
3000 IF A$="N" THEN GRAPHICS 0:POKE 71
0,0:GOTO 2880
3010 GOTO 2990
```

Instructions wanted ("Y"): The program comes here to change mode to graphics 0, poke colors into inner and outer window, cut off cursor, and print instructions.

```
3020 GRAPHICS 0:POKE 710,130:POKE 712,
130:POKE 752,1
3030 PRINT:PRINT " You have been assig
ned the duty of    giving instructions
to a warrior.":PRINT
3040 PRINT " The warrior's task is to
explore the Halls of Death and if poss
ible to"
3050 PRINT "retrieve the GOLDEN TREASU
RE CHEST     which is invisible to most
mortals."
3060 PRINT:PRINT " The warrior is in t
wo-way communica- tion and will report
everything"
3070 PRINT "he sees but will do nothin
g until     instructed by you."
3080 PRINT:PRINT " There are rumors of
a Magic Ring     which will make the
chest visible"
3090 PRINT "to its owner."
3100 PRINT:PRINT:PRINT "  WHEN READY T
O BEGIN TRANSMISSION[14 spaces][invers
e] PRESS RETURN [inverse off]":GET A$
3110 GOTO 2880
```

Go to the room number indicated by the variable R. *Example:* If R=1, the program will go to line 600; if R=2 it will go to 610, etc. If R is more than 21, excecution drops to the next logical line and checks if R is less than or equal to 40. This step is necessary because each line of BASIC code can be no more than three physical lines long.

```
2880 IF R<=21 THEN ON R GOTO 600,610,5
0,70,90,110,120,130,150,160,170,180,19
0,200,210,220,230,240,250,260,270
2890 IF R<=40 THEN ON R-21 GOTO 290,30
0,310,320,340,350,360,370,380,390,410,
420,430,440,460,470,480,490,500
```

```
2900 ON R-40 GOTO 510,520,530,540,550,
560,570,600,610,520,630
```

Room #1: L$=A description of location. *Note:* Other rooms can contain up to two other variable strings (V1$ and V2$). These can change depending on what the player does in the adventure. L$, however, remains constant. Also, other rooms will contain the variables N,S,E,W,U,D to give R the next room number. But room #1 is a special case requiring a special kind of action to get to another location:

```
600 L$=" THE HALLS OF DEATH. IN FRONT
OF ME    IS A NARROW PIT. BEYOND THE P
IT IS    A PASSAGEWAY.":GOTO 870
```

In the print routine, use the value of the variable R to set the colors of the rooms on the screen so that they will always be the same if the player comes back to them. Format the screen and print the information:

```
870 SETCOLOR 6,R,2:SETCOLOR 8,R,2:PRIN
T AT(1,0);"[esc ctrl clear][inverse/13
spaces]I AM IN:[14 spaces/inverse off]
"
880 PRINT L$:PRINT V1$:PRINT V2$
890 PRINT AT(2,7);"[inverse/13 spaces]
EXITS ARE:[12 spaces/inverse off]"
```

Print directions if value of direction variable is not equal to zero:

```
900 IF N<>0 THEN PRINT " NORTH";
910 IF S<>0 THEN PRINT " SOUTH";
920 IF E<>0 THEN PRINT " EAST";
930 IF W<>0 THEN PRINT " WEST";
940 IF U<>0 THEN PRINT " UP";
950 IF D<>0 THEN PRINT " DOWN";
```

Print the items visible in this location. The routine below checks all available items in Location I(n) against the room number the player is currently occupying. If I(n)=room #, then it adds I$(n) to a string called IT$ along with a comma. When the search is over, it removes the last comma from the string. If IT$ = " " (nothing) then let IT$ = GNIHTON. Go to a subroutine to convert the coded message into something readable:

```
960 PRINT AT(2,10);"[inverse/12 spaces
]ITEMS I SEE:[11 spaces]":IT$=""
970 FOR T=1 TO 22:IF I(T)=R THEN IT$=I
T$+" ,"+I$(T)
980 NEXT:IF IT$<>"" THEN IT$=RIGHT$(IT
$,LEN(IT$)-2)
990 IF IT$="" THEN IT$="  GNIHTON"
1000 M$=IT$:PRINT AT(3,11);:GOSUB 20:G
OTO 1020
```

Place the decoder subroutine near the beginning of the program. *Note:* Whenever BASIC is told to GOSUB (line number), it always starts at the lowest line number and checks down through the program until the line is found. It also uses this method in FOR/NEXT loops; therefore, FOR/NEXT loop at the end of the program takes longer to execute than one at the beginning. So, by placing much-used subroutines at the beginning of the program, it will execute faster.

The subroutine below reads the message in M$, starting at the end and reading to the beginning. It then prints each letter and uses the ATASCII value of that letter to create a sound effect.

```
20 FOR T=LEN(M$) TO 1 STEP-1:PRINT MID
$(M,T,1);:SOUND 0,ASC(MID(M$,T,1),10,1
0:NEXT:SOUND 0,0,0,0:RETURN
```

The program turns on the cursor, prints a message and waits for input. After player's input, it turns off cursor and clears C1$ and C2$. *Note:* Since the player will be using commands of more than one word, strings must be prepared to handle new input by clearing out any old information. This can be easily done by setting the string to equal closed quotation marks. C1$ = first word from player input and C2$ = second word or series of words from input. (In this game, for example, there are four keys, none of which can be picked up with the command TAKE KEY. The player must describe it as the brass key, the gold key, the silver key or the iron key. Therefore, the iron key could only be picked up by the command TAKE IRON KEY.)

```
1020 POKE 752,0:PRINT AT(2,16);"[inver
verse / 10 spaces]BY YOUR COMMAND:[9 s
paces]":INPUT CO$:POKE 752,1:C1$="":C2
$="":D3=0
```

The next line checks to see if user input is more than one letter. If it is, the program jumps to line 1210. If not, the program checks the next line to see if anything at all was entered. If nothing was entered, it enters a message into M$ and jumps to 1010.

```
1030 IF LEN(CO$)>1 THEN 1210
1040 IF CO$="" THEN M$="!EM NODRAP[esc
 ctrl 2]":GOTO 1010
```

Line 1010 sends the program to the decoder subroutine, jumps to a subroutine to delay (timed by the length of the message) the program so the message can be read, then clears it from the screen and returns for more user input.

```
1010 GOSUB 20:GOSUB 3490
```

```
3490 FOR T=1 TO LEN(M$)*100:NEXT:PRINT
 CL$:PRINT CL1$:CO$="":RETURN
```

If the input of the player was indeed one letter, a routine is necessary to check to see if it was to use the convenience directional string, DI$. With the use of a FOR/NEXT loop, the program checks each letter in the DI$ string to see if it matches the player's input. If it does, then T1 will be set to the value of T (the location within DI$). But, if after checking all the letters in DI$, no matches are found (the variable T1 still equals 0), the program places a message in M$ and sends the program to line 1010 again.

```
1050 T1=0:FOR T=1 TO 6:IF CO$=DI$(T) T
HEN T1=T:T=6
1060 NEXT:IF T1<1 THEN M$="!!TAHT DNAT
SREDNU T'NDID I[esc ctrl 2]":GOTO 1010
```

The program checks the value of T1 and goes to a routine which checks the value of the direction variable (N,S,E,W,U, or D). If it equals zero, the program jumps to a message routine. If it is more than zero, R (room number variable) receives the value of the direction variable and jumps to a routine to change locations.

```
1070 ON T1 GOTO 1080,1100,1120,1140,11
60,1180
1080 IF N=0 THEN 1200
1090 R=N:GOTO 2840
1100 IF S=0 THEN 1200
1110 R=S:GOTO 2840
1120 IF E=0 THEN 1200
1130 R=E:GOTO 2840
1140 IF W=0 THEN 1200
1150 R=W:GOTO 2840
1160 IF U=0 THEN 1200
1170 R=U:GOTO 2840
1180 IF D=0 THEN 1200
1190 R=D:GOTO 2840
1200 M$="!!!NOITCERID TAHT NI OG T'NAC
 I[esc ctrl 2]":GOTO 1010
```

The routine to change locations tells the player OK, then clears the variables:

```
2840 PRINT:PRINT "OKAY!"
2870 L$="":V1$="":V2$="":IT$="":CO$=""
:N=0:S=0:E=0:W=0:U=0:D=0
```

Now, type LIST 2840-2900 and hit RETURN. You can see that you are back to the routine that checks variable R and jumps to the appropriate room.

```
(*)RUN
```

Up to now, the program produces the title, gives instructions, and prints a description of the player's location. Now, to solve the problem of moving in Room #1, it is necessary to create a vocabulary. There are several ways to do this. Presented here is the most straightforward way. Later, another way to create and handle the program's vocabulary will be presented.

In line 1030, the program sent control to line 1210 if the player input was more than one letter. Here, the two-word commands will be separated into a string for each word by looking for a space between the words. Upon finding a space it will place the words on either side of the space into two smaller strings, C1$ and C2$. Here is where the fun begins—the words must then be reversed to match the coded versions of the vocabulary. This is done by a subroutine in line 30. Upon returning, each word is then reduced to its first three letters and they will be processed in this manner throughout the rest of the program. *Note:* the player could now input only the first three letters of each word and the program would understand them in that form.

```
1210 FOR T=1 TO LEN(CO$):IF MID$(CO$,T
,1)=" " THEN C2$=RIGHT$(CO$,LEN(CO$)-T
):T=LEN(CO$)
1220 NEXT:C1$=CO$:C4$="":C3$=C1$:GOSUB
 30:C1$=RIGHT$(C4$,3):C4$="":C3$=C2$:G
OSUB 30:C2$=RIGHT$(C4$,3)
```

```
30 FOR T=LEN(C3$) TO 1 STEP-1:C4$=C4$+
MID$(C3,T,1):NEXT:RETURN
```

Note: In line 1210, the command following the IF/THEN statement, T=LEN(CO$), sets the FOR/NEXT loop to maximum so that the program will not continue to loop and change the contents of C2$. This method is used at other times to speed up the program when a match is found, as in line 1050 when T1 is set, T is set to maximum.

Below is the straightforward way of checking the words in the vocabulary. Remember that each word the player typed in has been reduced to three letters and is now in reverse. These are the words that are being matched.

```
1230 IF C1$=" OG" THEN 1450
1240 IF C1$="LNU" OR C1$="EPO" THEN 15
50
1250 IF C1$="MUJ" THEN 1620
1260 IF C1$="ILC" THEN 1680
1270 IF C1$="VIG" OR C1$="ORD" OR C1$=
"RHT" THEN 1720
```

```
1280 IF C1$="TEG" OR C1$="KAT" THEN 18
50
1290 IF C1$="OOL" OR C1$="AXE" THEN 18
10
1300 IF C1$="LEH" THEN M$=".GNIDAER YB
 TOL A NRAEL ELPOEP":GOTO 1010
1310 IF C1$="AER" THEN 1950
1320 IF C1$="IRD" THEN 2220
1330 IF C1$="VAW" THEN 2280
1340 IF C1$="LIK" OR C1$="GIF" THEN 23
30
1350 IF C1$="EPS" OR C1$="YAS" THEN 24
60
1360 IF C1$="EEF" OR C1$="UOT" THEN 25
40
1370 IF C1$="VNI" THEN 2580
1380 IF C1$="TXE" AND I(14)=0 AND DK=1
 THEN DK=0:GOTO 2840
1390 IF C1$="GIL" AND I(1)=0 AND I(14)
=0 AND DK=0 THEN DK=1:I$(14)="HCROT GN
INRUB":GOTO 2840
1400 IF C1$="IRW" OR C1$="GIS" THEN 26
30
1410 IF C1$="UOP" AND RIGHT$(I$(10),8)
="FO SSALG" THEN I$(10)="SSALG":GOTO 2
680
1420 IF C1$="GID" AND I(2)=0 THEN 2740
1430 IF C1$="SNI" OR C1$="ESU" THEN 27
70
1440 M$="!!TNAW UOY TAHW DNATSREDUN T'
NOD I[esc ctrl 2]":GOTO 1010
```

Note: In line 1390, for example, in addition to looking for a match with the word LIGHT, the program is also checking to see if the player has the necessary torch and matches. If no matches are found by the time the player reaches line 1440, M$ is given a message which jumps to line 1010 to be displayed, read, erased, and then the program waits for further input.

Now create a routine for the word JUMP (line 1250). *Note:* there are many reasons to jump in this program. All of them are defined here. First, the program must always check the Room (R) to see if the player is allowed to jump.

```
1620 IF R=51 OR R=6 THEN M$="!!!PU HGI
H OOT S'TI":GOTO 1010
1630 IF R=5 OR R=4 THEN M$="!!!SSORCA
RAF OOT S'TI":GOTO 1010
```

In line 1640 a way to jump into the pit will be created. In line 1650 and 1660 ways to jump over the pit will be created.

```
1640 IF C2$="WOD" AND R=1 OR C2$="WOD"
 AND R=46 THEN R=6:GOTO 2840
1650 IF LEFT$(C4$,3)="TIP" AND R=1 THE
N R=46:GOTO 2840
1660 IF LEFT$(C4$,3)="TIP" AND R=46 TH
EN R=1:GOTO 2840
1670 M$="? ? ? EREHW PMUJ":GOTO 1010
```

Now that movement to other rooms is possible, addition of more program lines describing new rooms is necessary.

```
110 L$=" THE BOTTOM OF A PIT. SOUTH I
SEE A     DOOR WITH NO HANDLE. ON A TAB
LE IS A  BOOK AND PEN.":GOTO 860
440 L$=" A LARGE RED ROOM.":S=38:N=46:
E=36:W=47:IF TT>1 THEN V1$=" THERE ARE
FOOTPRINTS IN THE DUST."
450 GOTO 870
460 L$=" A NARROW PASSAGE.":W=35:E=37:
GOTO 870
470 L$=" AN L-SHAPED PASSAGE.":S=40:W=
36:GOTO 870
480 L$=" AN L-SHAPED PASSAGE.":W=41:N=
35:GOTO 870
490 L$=" AN L-SHAPED PASSAGE.":E=40:S=
42:GOTO 870
500 L$=" AN L-SHAPED PASSAGE.":N=37:W=
39:GOTO 870
510 L$=" AN L-SHAPED PASSAGE.":S=42:E=
38:GOTO 870
520 L$=" AN L-SHAPED PASSAGE.":TT=TT+1
:E=43:N=41:GOTO 870
530 L$=" A NARROW PASSAGE.":TT=TT+1:W=
42:E=44:GOTO 870
540 L$=" A SMALL ROOM.":N=39:W=43:GOTO
```

```
 870
550 L$=" A SMALL JAIL CELL. THERE IS A
 SIGN ON THE WALL. THROUGH THE BARS I
SEE A    TORTURE CHAMBER.":GOTO 870
560 L$=" A NARROW PAGGAGE. THERE IS A
PIT      NORTH.":S=35:GOTO 870
570 T1=0:FOR T=1 TO 22:IF I(T)=0 THEN
T1=T:T=22
580 NEXT:IF T1>0 THEN R=45:GOTO 2880
590 L$=" A VERY SMALL ROOM.":E=35:GOTO
 870
```

Now, line 860 must be added so that line 110 will have some place to go. Check to see if D7 has been set to 1 and, if so, open the door and set the directional variable.

```
860 IF R=6 AND D7=1 THEN V1$=" THE DOO
R IS OPEN":S=11
```

```
(*)RUN
```

Use of the Jump and one-letter commands for direction enable the player to move around in the adventure. Now create a routine to add darkness to the dungeon. Add line 2850 to ascertain if player is carrying a torch.

```
2850 IF I(14)<>0 THEN DK=0
```

Change line 2840.

```
2840 PRINT:PRINT "OKAY!":IF DK=0 THEN
GOSUB 3200
```

Add the following subroutine:

```
3200 I$(14)="HCROT":IF R=1 OR R=46 OR
R=35 THEN RETURN
3210 POKE 764,255:POKE 710,0:POKE 712,
0:PRINT "[esc ctrl clear / inverse / 6
 spaces]I CAN'T SEE - WHERE AM I?[5 sp
aces]"
3220 PRINT AT(2,15);"[inverse / 13 spa
ces]COMMAND?[14 spaces]"
```

```
3230 PRINT AT (16,7);"[ctrl FMG /space
/ctrl FMG]"
3240 FOR EY=1 TO 2:PRINT AT(16,8);"[sp
ace/ctrl T/3 spaces/ctrl T/space]"
3250 FOR T=1 TO 150:NEXT T
3260 PRINT AT(16,8);"[ctrl T/3 spaces/
ctrl T/2 spaces]"
3270 FOR T=1 TO 150:NEXT T
3280 PRINT AT(16,8);"[space/ctrl T/3 s
paces/ctrl T/space]"
3290 FOR T=1 TO 150:NEXT T
3300 PRINT AT(16,8);"[2 spaces/ctrl T/
3 spaces/ctrl T]"
3310 FOR T=1 TO 150:NEXT T
3320 PRINT AT(16,8);"[space/ctrl T/3 s
paces/ctrl T/space]"
3330 FOR T=1 TO 150:NEXT T:NEXT EY:IF
PEEK(764)<>255 THEN 3360
3340 PRINT AT(16,8);"[space/ctrl RR/2
spaces/ctrl RR]"
3350 FOR T=1 TO 200:NEXT:GOTO 3240
```

Note: The first line sets the value of I$(14) to TORCH, then checks to see what room player is in (Room 1, 35, and 46 are the only places the player can be without a burning torch). The second line prepares the keyboard for input, turns the screen black, and prints a message. The rest of the routine is a straightforward way of making a set of moving, blinking eyes and checks for a key to be touched.

Now, set up a way to get back to the main program:

```
3360 POKE 752,0:POKE 764,255:INPUT AT(
2,16) CO$:IF LEN(CO$)<3 THEN PRINT AT(
2,16);"     ":GOTO 3360
3370 POKE 752,1:IF LEFT$(CO$,3)="LIG"
AND I(1)=0 AND I(14)=0 THEN DK=1:GOSUB
 3500:I$(14)=I$(14)+" GNINRUB":RETURN
3380 IF LEFT$(CO$,3)="LIG" AND I(1)<>0
 THEN M$="!!SEHCTAM YNA EVAH T'NOD I":
GOSUB 20:GOSUB 3490:GOTO 3360
3390 IF LEFT$(CO$,3)="LIG" AND I(14)<>
0 THEN M$="!!HCROT A EVAH T'NOD I":GOS
```

```
UB 20:GOSUB 3490:GOTO 3360
3400 M$="!!KRAD     EHT NI SGNIHT OD OT
 SUOREGNAD S'TI":GOSUB 20:GOSUB 3510:G
OTO 2910
```

Note: the only way back to the main program is to light the torch. Any other action can be fatal.

There are three other possible directions in which the above routine can go, depending upon the player's input. Each is self-explanatory. The following subroutines say OK and delay the program for messages to be read:

```
3500 PRINT "OKAY!":FOR T=1 TO 150:NEXT
:RETURN

3510 FOR T=1 TO LEN(M$)*100:NEXT:RETUR
N
```

This routine allows the player another chance:

```
2910 GRAPHICS 0:POKE 752,1:POKE 710,0:
GOSUB 3450:PRINT AT(2,10);"YOU SEEM TO
HAVE GOTTEN YOUR WARRIOR   KILLED."
2920 PRINT:PRINT "WOULD YOU LIKE TO SE
ND ANOTHER          ONE DOWN TO TRY AGAI
N   (Y/N)":PRINT
2930 A$=INKEY$:IF A$="Y" THEN RUN
2940 IF A$="N" THEN M$="!EMITEMOS NIAG
A YRT":GOSUB 20:END
2950 GOTO 2930
```

Now, in order for the player to survive, the routine from line 1280 must be completed to allow the items in each room to be taken.

```
1850 IF C2$="DAL" AND R=25 OR C2$="DAL
" AND R=51 THEN I(19)=0:LD=1:D6=0:GOSU
B 3500:GOTO 2880
1860 IF C2$="OOB" AND R=6 OR C2$="NEP"
 AND R=6 THEN M$=".LLAW EHT OT DENIAHC
 SI TI":GOTO 1010
1870 IF IN>4 THEN M$=".EROM GNIHTYNA Y
RRAC T'NAC I[esc ctrl 2]":GOTO 1010
1880 T1=0:FOR T=1 TO 22:IF C2$=RIGHT$(
I$(T),3) AND I(T)=R THEN I(T)=0:T1=T:T
=22
1890 NEXT:IF T1=6 OR T1=8 THEN 1930
1900 IF R=31 AND I(10)=0 AND C2$="TAW"
 THEN I$(10)="RETAW FO SSALG":M$=I$(10
)+" A EVAH UOY":CN=1:GOTO 1010
1910 IF T1>0 THEN IN=IN+1:GOSUB 3500:G
OSUB 40:GOTO 960
1920 M$="!EREH TI EES T'NOD I":GOTO 10
10
1930 IF I(10)=0 THEN I$(10)="DICA FO S
SALG":I(T1)=R:CN=2:GOSUB 3500:GOSUB 40
:GOTO 960
1940 I(T1)=R:M$="!RENIATNOC ON EVAH I"
:GOTO 1010
```

Note: The routine checks to see how much the player is carrying; if more than 4 items, it prints a message and returns to input. Next it checks for what items the player wants and to ascertain that he or she is in the room with the item. Then it adds the items to the player's inventory by setting the value to zero [I(n) = 0 = in player's possession = n = room number] or prints another message.

A routine to delete the items from the screen that are picked up is needed next.

```
40 PRINT AT(2,11);"[8 esc shift delete
s]":RETURN

(*)RUN
```

In order to allow the player to see the items being carried, the inventory routine from line 1370 must be completed. It loops to check each item and any time it finds one equal to zero, it prints it on the screen.

```
2580 PRINT "[esc ctrl clear / inverse
/ 8 spaces]WARRIOR'S INVENTORY:[7 spac
es]:PRINT:PRINT
2590 FOR CH=1 TO 22:IF I(CH)=0 THEN M$
=I$(CH):GOSUB 20:PRINT
```

```
2600 NEXT:PRINT AT(7,22);"PRESS[invers
e] ANY KEY [inverse off]TO CONTINUE":P
OKE 764,255
2610 IF PEEK(764)=255 THEN 2610
2620 POKE 764,255:GOTO 2880
```

```
(*)RUN
```

To test your program so far, pick up the items you find and then type INVENTORY.

Add the rest of the rooms. *Note:* There will be many variables V1$ and V2$, each of which is checked again in lines 660, 680, etc. These lines will be added shortly.

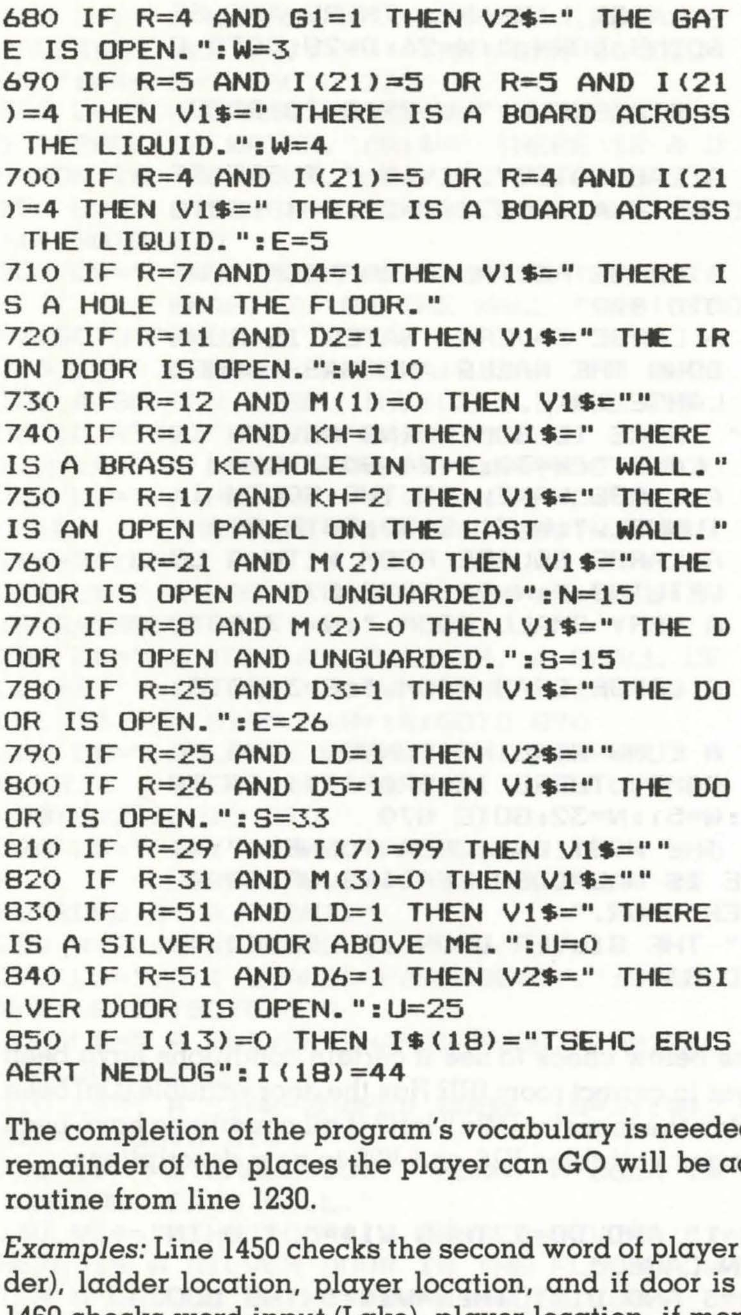
```
50 L$=" A HALLWAY. THERE IS A METAL DO
OR      WEST AND AN IRON GATE EAST.":V
1$=" THE DOOR IS LOCKED."
60 V2$=" THE GATE IS LOCKED.":S=8:GOTO
 660
70 L$=" THE WEST END OF A FOUL SMELLIN
G       ROOM. AN IRON GATE IS WEST."
80 V1$=" MY WAY EAST IS BLOCKED BY A P
OOL OF   LIQUID.":V2$=" THE GATE IS LO
CKED.":GOTO 680
90 L$=" THE EAST END OF A FOUL SMELLIN
G      ROOM."
100 V1$=" MY WAY WEST IS BLOCKED BY A
POOL OF    LIQUID.":S=10:GOTO 690
120 L$=" A SMALL ROOM.":V1$=" AN X IS
MARKED ON THE FLOOR.":E=8:S=14:GOTO 71
0
130 L$=" A HALLWAY. A MAN'S BODY IS ON
 THE      FLOOR. CHALK IS IN HIS HAND A
ND THERE IS WRITING ON THE WALL."
140 N=3:E=9:W=7:V1$=" A GIANT BLOCKS A
 DOORWAY SOUTH.":GOTO 770
150 L$=" A NARROW PASSAGE. FIRE BLOCKS
MY WAY    SOUTH.":W=8:GOTO 870
160 L$=" A NARROW PASSAGE. A FOUL SMEL
L COMES   FROM THE NORTH.":N=5:E=11:GOT
O 870
```

```
170 L$=" A NARROW PASSAGE. AN IRON DOO
R LEADS  WEST.":V1$=" THE DOOR IS LOCK
ED.":N=6:S=17:GOTO 720
180 L$=" A LARGE ROOM. IN THE CORNER I
S A POOL OF WATER.":V1$=" THERE IS A D
RAGON IN THE ROOM.":E=13:S=18:GOTO 730
190 L$=" A SLOPING PASSAGEWAY.":W=12:E
=14:GOTO 870
200 L$=" A NARROW PASSAGEWAY. THERE IS
         GRAFITTI ON THE WALL.":W=13:N
=7:GOTO 870
210 L$=" A LARGE ROOM.":V1$=" I'M OUTS
IDE A GOLDEN CAGE. INSIDE I   SEE THE
MAGIC RING.":N=8:S=21:GOTO 650
220 L$=" A LARGE CAVERN.":N=9:GOTO 870
230 L$=" A HALLWAY WITH AN ARROW CARVE
D IN     THE WALL POINTING SOUTH.":N=1
1:S=24:GOTO 740
240 L$=" A LARGE PASSAGEWAY":N=12:E=19
:S=26:GOTO 870
250 L$=" A TORTURE CHAMBER. A SMALL CE
LL IS    IN THE NORTH CORNER. IN THE C
ELL IS   A SIGN.":W=18:GOTO 870
260 L$=" A LARGE ROOM WITH ROUGH-CUT W
ALLS.    THERE ARE STAIRS GOING UP.":U
=28:E=21:GOTO 870
270 L$=" A T-SHAPED PASSAGE WITH A DOO
R        NORTH.":V1$=" THE DOORWAY IS
GUARDED BY A GIANT."
280 W=20:E=22:GOTO 760
290 L$=" A T-SHAPED PASSAGEWAY.":S=29:
W=21:E=23:GOTO 870
300 L$=" A CLOVER-SHAPED ROOM.":W=22:E
=24:GOTO 870
310 L$=" A STAR-SHAPED ROOM.":S=31:W=2
3:N=17:GOTO 870
320 L$=" A SMALL DUSTY ROOM. A DOOR IS
 ON THE  EAST WALL."
330 V1$=" THE DOOR IS CLOSED.":V2$=" T
HERE IS A SILVER DOOR IN THE FLOOR   W
ITH A LADDER GOING DOWN.":GOTO 780
340 L$=" A NARROW HALLWAY. THERE IS A
DOORWAY  SOUTH.":V1$=" THE DOOR IS CLO
SED.":N=18:E=27:GOTO 800
```

```
350 L$=" A LARGE LIBRARY. THERE ARE ST
AIRS      GOING DOWN.":W=26:D=28:GOTO 8
70
360 L$=" A STAIRWELL.":U=27:D=20:GOTO
870
370 L$=" A LABORATORY.":V1$=" THERE IS
 A WIZARD WITH A WAND.":N=22:E=30:GOTO
 810
380 L$=" A LARGE RECTANGULAR ROOM.":W=
29:E=31:GOTO 870
390 L$=" A LARGE CAVERN. WATER IS RUNN
ING       DOWN THE WALLS AND HAS FORMED
 A        LARGE LAKE."
400 V1$=" THERE IS SOMETHING MOVING IN
 THE       LAKE.":W=30:N=24:GOTO 820
410 L$=" A LARGE LAKE. TO THE SOUTH I
SEE A     TUNNEL.":W=31:S=50:GOTO 870
420 L$=" A LARGE SQUARE ROOM WITH A LO
W         CEILING.":N=26:GOTO 870
430 L$=" A VERY SMALL ROOM.":W=17:GOTO
 870
610 L$=" A LARGE DRAB ROOM.":E=3:GOTO
870
620 L$=" A LONG WINDING TUNNEL. THE FL
OOR IS    DAMP. THERE IS GRAFITTI ON TH
E WALL.":W=51:N=32:GOTO 870
630 L$=" THE WEST END OF A TUNNEL.":V1
$=" THERE IS A LADDER LEADING UP TO A
    SILVER DOOR."
640 V2$=" THE SILVER DOOR IS LOCKED.":
E=50:GOTO 830
```

The routines below check to see if certain conditions have been met: Is player in correct room [R]? Has the door variable [Dn] been changed? Has the monster [M(n)] left? If all conditions have been met, it changes variables V1$ and V2$ to new descriptions.

```
650 IF R=15 AND D8=1 THEN V1$=" I'M IN
 A GOLDEN CAGE."
660 IF R=3 AND D1=1 THEN V1$=" THE DOO
R IS OPEN.":W=2
670 IF R=3 AND G1=1 THEN V2$=" THE GAT
E IS OPEN.":E=4
```



```
680 IF R=4 AND G1=1 THEN V2$=" THE GAT
E IS OPEN.":W=3
690 IF R=5 AND I(21)=5 OR R=5 AND I(21
)=4 THEN V1$=" THERE IS A BOARD ACROSS
 THE LIQUID.":W=4
700 IF R=4 AND I(21)=5 OR R=4 AND I(21
)=4 THEN V1$=" THERE IS A BOARD ACRESS
 THE LIQUID.":E=5
710 IF R=7 AND D4=1 THEN V1$=" THERE I
S A HOLE IN THE FLOOR."
720 IF R=11 AND D2=1 THEN V1$=" THE IR
ON DOOR IS OPEN.":W=10
730 IF R=12 AND M(1)=0 THEN V1$=""
740 IF R=17 AND KH=1 THEN V1$=" THERE
IS A BRASS KEYHOLE IN THE      WALL."
750 IF R=17 AND KH=2 THEN V1$=" THERE
IS AN OPEN PANEL ON THE EAST    WALL."
760 IF R=21 AND M(2)=0 THEN V1$=" THE
DOOR IS OPEN AND UNGUARDED.":N=15
770 IF R=8 AND M(2)=0 THEN V1$=" THE D
OOR IS OPEN AND UNGUARDED.":S=15
780 IF R=25 AND D3=1 THEN V1$=" THE DO
OR IS OPEN.":E=26
790 IF R=25 AND LD=1 THEN V2$=""
800 IF R=26 AND D5=1 THEN V1$=" THE DO
OR IS OPEN.":S=33
810 IF R=29 AND I(7)=99 THEN V1$=""
820 IF R=31 AND M(3)=0 THEN V1$=""
830 IF R=51 AND LD=1 THEN V1$=" THERE
IS A SILVER DOOR ABOVE ME.":U=0
840 IF R=51 AND D6=1 THEN V2$=" THE SI
LVER DOOR IS OPEN.":U=25
850 IF I(13)=0 THEN I$(18)="TSEHC ERUS
AERT NEDLOG":I(18)=44
```

The completion of the program's vocabulary is needed next. The remainder of the places the player can GO will be added to the routine from line 1230.

Examples: Line 1450 checks the second word of player input (Ladder), ladder location, player location, and if door is open. Line 1460 checks word input (Lake), player location, if monster is still there, etc. Lines 1520 and 1530 check to see if the player asks to go in a direction by using the first letter of the player's second word and compares it to the directional string, DI$(n).

```
1450 IF C2$="DAL" AND I(19)=99 AND R=5
1 AND D6=1 THEN R=25:GOTO 2840
1460 IF C2$="KAL" AND R=31 AND M(3)=1
THEN M$="!!GEL YM TA GNILBBIN SI GNIHT
EMOS":GOSUB 20:GOSUB 3490:GOTO 2910
1470 IF C2$="NUT" AND R=32 THEN R=50:G
OTO 2840
1480 IF C2$="KAL" AND R=31 THEN R=32:D
K=0:GOTO 2840
1490 IF C2$="DAL" AND R=6 AND I(19)=R
THEN R=1:GOTO 2840
1500 IF C2$="WOD" AND R=1 OR C2$="WOD"
 AND R=46 THEN R=6:GOTO 2840
1510 IF C2$="RIF" AND I(18)=99 AND R=9
 THEN R=16:GOTO 2840
1520 T1=0:FOR T=1 TO 6:IF RIGHT$(C2$,1
)=DI$(T) THEN T1=T
1530 NEXT:IF T1>0 THEN 1070
1540 M$="? ? ? ? ? EREHW OG[esc ctrl 2
]":GOTO 1010
```

Next, the routine to SIGN or WRITE from line 1400:

```
2630 IF C2$="MAN" AND R=6 OR C2$="OOB"
 AND R=6 THEN 2660
2640 IF C2$="" THEN M$="??TAHW ETIRW":
GOTO 1010
2650 M$="???"+C4$+" ETIRW I DLUOHS YHW
":GOTO 1010
2660 INPUT "ENTER YOUR NAME [esc esc/e
sc ctrl *]";NA$:IF LEN(NA$)<2 THEN PRI
NT "[2 esc ctrl -'s]":GOTO 2660
2670 D7=1:GOTO 2880
```

```
(*)RUN
```

The next words to be added are OPEN and UNLOCK, because both are sent to the same routine from line 1240. Here the program checks for word, room, and sometimes the proper key—I(4), I(15), I(16).

```
1550 IF C2$="OOD" AND R=11 AND I(4)=0
THEN D2=1:GOTO 2880
1560 IF (C2$="TAG" AND R=3 AND I(4)=0)
```

```
 OR (C2$="TAG" AND R=4 AND I(4)=0) THE
N G1=1:GOTO 2880
1570 IF C2$="GAC" AND R=15 AND I(15)=0
 THEN D8=1:I(13)=15:GOTO 2880
1580 IF C2$="OOD" AND R=26 THEN D5=1:G
OTO 2840
1590 IF C2$="OOD" AND R=25 THEN D3=1:E
26:GOTO 2880
1600 IF C2$="LIS" AND I(16)=0 AND R=51
 THEN D6=1:GOTO 2840
1610 M$="!!T'NAC I":GOTO 1010
```

Now, add to the word CLIMB in 1260. Here the routine checks to see if the player is permitted to climb and, if not, uses the fact that the player's second word is stored in C4$ intact to tell what object he or she is not allowed to climb:

```
1680 IF C2$="LAW" THEN M$="!OT NO DLOH
OT GNIHTON SI EREHT":GOTO 1010
1690 IF C2$="DAL" AND I(19)=R AND R=6
THEN R=1:GOTO 2840
1700 IF C2$="DAL" AND I(19)=R AND R=51
 THEN R=25:GOTO 2840
1710 M$="!!"+C4$+" BMILC T'NAC I":GOTO
 1010
```

The next words to be added are DROP, GIVE, and THROW. Each of these sends the program here from line 1270. This checks to see if the player is indeed carrying the object he or she wants to drop, and if dropping it in a certain location will cause something else to happen; if so, it then makes it happen.

```
1720 T1=0:FOR T=1 TO 22:IF C2$=RIGHT$(
I$(T),3) AND I(T)=0 THEN I(T)=R:T1=T:T
=22:IN=IN-1
1730 NEXT:IF R=45 THEN 2790
1740 IF T1=7 AND R=29 THEN I(11)=29:M$
=".SRAEPPASID DNA DNAW SPORD DRAZIW":I
(7)=99:GOSUB 20:GOSUB 3510:GOTO 2880
1750 IF T1=21 THEN 2840
1760 IF T1=22 AND R=31 THEN M(3)=0:I(T
1)=99:I(5)=32:GOTO 1800
```

```
1770 IF T1=19 AND R=25 OR T1=19 AND R=
51 THEN LD=0:GOTO 2840
1780 IF T1>0 THEN GOSUB 3500:GOSUB 40:
GOTO 960
1790 M$="!TAHT EVAH T'NOD I":GOTO 1010
1800 M$=".SRAEPPASID NEHT DNA  ERUTAER
C EHT STAE - SRAEPPA NOGARD A":GOSUB 2
0:GOSUB 3510:GOTO 2880
```

There are two special routines that may happen when the player drops items. The first checks to see if the torch was dropped, as it is necessary to carry the torch at all times for light.

```
2850 IF I(14)<>0 THEN DK=0
```

Second, the only way a player can get out of the jail cell is to drop all of the items he or she is carrying. In order to accomplish this, a routine must be created to check the player's inventory. The program comes here from line 1730.

```
2790 I(T1)=35:IF T1=4 THEN I(T1)=19
2800 IF T1=14 THEN I$(T1)="HCROT"
2810 T1=0:FOR T=1 TO 22:IF I(T)=0 THEN
 T1=T:T=22
2820 NEXT:IF T1>0 THEN GOSUB 3500:GOSU
B 40:GOTO 960
2830 R=47:GOTO 2880
```

Here, add the words LOOK and EXAMINE to the routines from line 1290.

```
1810 IF C2$="KAL" AND R=31 AND M(3)=1
THEN M$="!THGIRLA EREHT ERUTAERC A S'E
REHT":GOTO 1840
1820 IF C2$="QIL" AND R=4 AND C2$="QIL
" AND R=5 THEN I(6)=4:I(8)=5:GOTO 2840
1830 M$=".LAICEPS GNIHTON EES I":GOTO
1010
1840 GOSUB 20:GOSUB 3490:M$="!YRGNUH S
KOOL TI":GOTO 1010
```

Complete the routine in line 1310 to READ books, walls, etc.

```
1950 IF C2$="RCS" AND I(3)=0 AND I(17)
=0 THEN M$="'WORRA HCUOT' :SYAS LLORCS
":GOTO 1010
1960 IF C2$="RCS" AND I(3)=0 THEN M$="
.EM OT KEERG S'TI":GOTO 1010
1970 IF C2$="CID" AND I(17)=0 THEN M$=
"'YRANOITCID KEERG' SYAS REVOC":I$(17)
=I$(17)+" KEERG":GOTO 1010
1980 IF R=6 THEN 2140
1990 IF C2$="OOB" AND I(9)<>0 THEN M$=
"!TI EVAH T'NOD I":GOTO 1010
2000 IF I(13)=0 THEN 2100
2010 IF C2$="OOB" AND R=9 THEN M$=".TO
H NEHW SPLEH DLOC GNIHTEMOS":GOTO 1010
2020 IF C2$="OOB" AND R=3 THEN M$=".LA
TEM TAE SEVISORROC":GOTO 1010
2030 IF C2$="OOB" AND R=4 THEN M$=".DO
OW FO EDAM ERA SEGDIRB":GOTO 1010
2040 IF C2$="OOB" AND R=12 THEN M$=".C
IGAM ERA SNOGARD":GOTO 1010
2050 IF C2$="OOB" AND R=29 THEN M$=".T
NATROPMI SI NOITACINUMMOC":GOTO 1010
2060 IF C2$="OOB" AND R=5 THEN M$="LUO
F LLEMS YLLAUSU SLACIMEHC":I(6)=4:I(8)
=5:GOSUB 20:GOSUB 3510:GOTO 2880
2070 IF C2$="OOB" AND R=30 THEN M$="'E
GDELWONK FO KOOB' :SDAER REVOC":GOTO 1
010
2080 IF C2$="OOB" AND R=25 AND I(18)=0
 THEN M$=".KNIRD GNILOOC A":GOTO 1010
2090 IF C2$="OOB" AND I(9)=0 THEN M$="
.TI NI NOITAMROFNI YNA DNIF T'NAC I":G
OTO 1010
2100 IF C2$="OOB" AND Q=0 THEN M$="!YT
PME SAW MOOR EHT":Q=1:GOTO 1010
2110 IF C2$="OOB" AND Q=1 THEN M$="!OG
 UOY DNUOR DNA DNUOR":Q=2:GOTO 1010
2120 IF C2$="OOB" AND Q=2 THEN M$="!!W
OL DNA HGIH KOOL":GOTO 1010
2130 IF (C2$="GIS" AND R=45) OR (C2$="
GIS" AND R=19) THEN M$=".LLEC NI DEWOL
LA TON ERA SNOISSESSOP":GOTO 1010
2140 IF C2$="OOB" AND R=6 AND I(18)=0
THEN M$=".GNIBMILC ROF PLEH DEEN ELPOE
```

```
P":GOTO 1010
2150 IF C2$="OOB" AND R=6 THEN M$="’.N
I NGIS TSUM STSEUG LLA’:SYAS KOOB":GOT
O 1010
2160 IF R=8 AND C2$="IRW" THEN C4$="":
C3$=NA$:GOSUB 30:NB$=C4$:M$="...EHT NI
 OG T’NOD "+NB$:GOTO 1010
2170 IF R=50 AND C2$="ARG" THEN M$="!!
DEID EW DNA DEIRT EW":GOSUB 20:GOSUB 3
490:M$="EVAD & KCAJ DENGIS":GOTO 1010
2180 IF R=14 AND C2$="ARG" THEN M$=".S
PLEH EGDELWONK FO KOOB EHT":GOTO 1010
2190 IF C2$="CID" OR C2$="RCS" OR C2$=
"OOB" THEN 2200 ELSE 2210
2200 M$="!!TI EVAH T’NOD I":GOTO 1010
2210 M$="!!!TAHT DAER T’NAC I":GOTO 10
10
```

Note: This reading routine can be condensed quite a bit if you wish by breaking down each thing that can be read and sending it to its own subroutine. (If you want to sneak a peek and see how I did it, look at the end of this listing.)

Now to complete the DRINK routine from line 1320. Check to see what the player wants to drink and if he or she is in the room with it or has it in possession.

```
2220 IF C2$="TAW" AND R=31 OR C2$="TAW
" AND R=32 THEN M$=".GNIHSERFER SAW TA
HT !HA":GOTO 1010
2230 IF C2$="TAW" AND I$(10)="RETAW OF
SSALG" THEN I$(10)="SSALG":M$="!TOPS E
HT TIH TAHT":GOTO 1010
2240 IF C2$="TAW" THEN M$="!RETAW YNA
EES T’NOD I":GOTO 1010
2250 IF C2$="QIL" AND R=5 OR C2$="QIL"
 AND R=4 THEN M$="!EHTAERB T’NAC I - S
NRUB TI":GOSUB 20:GOSUB 3510:GOTO 2910
2260 IF C2$="TOP" AND I(18)=0 THEN M$=
".PU EM MRAW T’NOW ERIF A ,DLOC OS M’I
":I(18)=99:GOTO 1010
2270 M$="!!TAHT KNIRD T’NAC I":GOTO 10
10
```

Add to the WAVE routine from line 1330. First, the program checks to see if player wants to wave the wand; next, if he or she has the wand; and finally, it creates the change if the player is in the appropriate room.

```
2280 IF C2$<>"NAW" THEN M$="!!TAHT EVA
W OT TNAW I DLUOW YHW":GOTO 1010
2290 IF I(11)<>0 THEN M$="!!TAHT EVAH
T'NOD I":GOTO 1010
2300 SOUND 2,20,4,15:FOR T=1 TO 300:NE
XT:SOUND 2,0,0,0:M$="!!TLOB YGRENE NA
STIME DNAW EHT":GOSUB 20:GOSUB 3490
2310 IF R=12 AND M(1)=1 THEN M$=".HSIF
 A OTNI SNRUT NOGARD EHT":M(1)=0:I(22)
=12:GOSUB 20:GOSUB 3510:GOTO 2880
2320 M$="!!DEGNAHC SMEES GNIHTON ,EGNA
RTS":GOTO 1010
```

Next come the routines to KILL and FIGHT the monsters. Here, the program checks to see which monster the player wishes to fight and sends control to the proper routine if he or she has the correct weapon.

```
2330 IF C2$="AIG" AND M(2)=1 AND R=8 T
HEN 2400
2340 IF C2$="AIG" AND M(2)=1 AND R=21
THEN 2400
2350 IF C2$="ARD" AND M(1)=1 AND R=12
THEN 2420
2360 IF C2$="ERC" AND M(3)=1 AND R=31
THEN M$="!RETAW EHT NI TON M'I":GOTO 1
010
2370 IF C2$="ZIW" AND R=29 THEN 2430
2380 IF C2$="" THEN M$="?TAHW THGIF":G
OTO 1010
2390 M$="!THGIF OT EREH GNIHTON S'EREH
T":GOTO 1010
2400 IF I(5)=0 THEN GOSUB 3520:M$="!DE
HSIUQNAV NEEB SAH TNAIG EHT":M(2)=0:GO
SUB 20:GOSUB 3510:GOTO 2880
2410 GOTO 2430
2420 IF I(11)=0 THEN M$="??OD I DLUOHS
 TAHW":GOTO 1010
```

```
2430 M$="!!NOPAEW THGIR EHT EVAH T'NOD
 I"
2440 IF RND(0)<0.5 THEN C4$="!!DEKCATT
A NEEB EV'I        "+M$:M$=C4$:GOSUB 20
:GOSUB 3520:GOTO 2910
2450 GOTO 1010
```

Now the subroutine for fight sound effects:

```
3520 FOR L=1 TO 20:SOUND 0,20,INT(8*RN
D(0))+1,15:FOR T=1 TO 8:NEXT T:SOUND 0
,0,0,0
3530 FOR T=1 TO INT(80*RND(0)):NEXT T:
NEXT L:RETURN
```

Continue with the routine to SPEAK or SAY something. First, the program learns what the player wants to say, then checks to see that what is said is something that the program understands.

```
2460 IF C4$="PLEH" THEN C1$="LEH" GOTO
 1300
2470 IF RIGHT$(C4$,1)="H" AND R=29 AND
 I(11)=99 THEN GOSUB 2530:M$="'?DNAW A
 YUB OT TNAW' :DRAZIW":AN=1:GOTO 1010
2480 IF AN=1 AND C2$="SEY" AND R=29 TH
EN GOSUB 2530:AN=2:M$="'.DLOG FO GAB A
 EM EVIG' :SYAS DRAZIW":GOTO 1010
2490 IF RIGHT$(C4$,1)="H" AND R=8 OR R
IGHT$(C4$,1)="H" AND R=21 THEN GOSUB 2
530:GOTO 2520
2500 IF C2$="" THEN M$="?YAS OT EM TNA
W UOY OD TAHW":GOTO 1010
2510 GOSUB 2530:M$="!!SNEPPAH GNIHTON"
:GOTO 1010
2520 M$=".REWSNA TON SEOD TNAIG EHT":G
OTO 1010
2530 M$="!YAKO":GOSUB 20:GOSUB 3490:M$
=C4$:GOSUB 20:GOSUB 3490:RETURN
```

The routine to TOUCH and FEEL is continued from line 1360 by checking to see what player wants to touch and the consequences of touching the object.

```
2540 IF C2$="RRA" AND R=17 THEN KH=1:M
$="!!SRAEPPA ELOHYEK SSARB A":GOSUB 20
:GOSUB 3490:GOTO 2880
2550 IF (C2$="QIL" AND R=4) OR (C2$="Q
IL" AND R=5) THEN I(6)=4:I(8)=5:GOTO 2
570
2560 M$=".EM OT "+C4$+" A EKIL SLEEF T
I":GOTO 1010
2570 M$="!!SNRUB REGNIF YM":GOSUB 20:G
OSUB 3490:M$="!GNISSIM EB OT SMEES REG
NIF YM":GOSUB 20:GOSUB 3510:GOTO 2880
```

The routine to POUR from line 1410 continues here to check to see what is in the glass; next, it checks to see if the player has the other necessary items to open the door. In either case, the player is told the results of his action.

```
2680 IF CN=1 THEN M$=".ROOLF EHT OTNI
SKAOS RETAW EHT":GOTO 1010
2690 IF R<>3 THEN M$="!!ROOLF EHT NI E
LOH A ETA DICA EHT":GOTO 1010
2700 IF I(12)=0 THEN M$="!KCOL EHT NI
ELOH A ETA DICA EHT":D1=1:GOSUB 20:GOS
UB 3490:GOTO 2880
2710 M$=".ROOD EHT NI SI ELOH A":PO=PO
+1:GOSUB 20:GOSUB 3490
2720 IF PO>0 THEN M$=".PLEH DLUOW LENN
UF EHT KNIHT I":GOTO 1010
2730 M$="!!PLEH T'NDID TAHT !EGNARTS":
GOTO 1010
```

Next is the routine to DIG from line 1420. The player is only allowed to dig in Room 7, so it first checks to see if the player is there, then checks to see if a hole has already been dug, and then digs the hole and places the item that was in the hole into Room 7.

```
2740 IF R<>7 THEN M$="!!EREH ROOLF ENO
TS A SI EREHT":GOTO 1010
2750 IF D4>0 THEN M$="!!EREH GUD YDAER
LA EV'I":GOTO 1010
2760 M$="!!ROOLF EHT NI ELOH A SI EREH
T":D4=1:I(16)=7:GOSUB 20:GOSUB 3510:GO
TO 2880
```

The final words in the vocabulary are INSERT or USE from line 1430.

```
2770 IF C2$<>"ARB" AND C2$<>"YEK" THEN
 2780
2775 IF R=17 AND I(20)=0 THEN KH=2:E=3
4:I(20)=99:IN=IN-1:GOTO 2880
2780 M$="!!KROW T'NSEOD TI":GOTO 1010
```

The win routine is needed to complete the program.

```
2860 IF I(18)=0 AND LEFT$(I$(18),5)="T
SEHC" AND R=1 THEN 3410
3410 GRAPHICS 18:PRINT #6,AT(3,4);"[in
verse]CONGRATULATIONS":PRINT #6,AT(6,5
);"you won!"
3420 FOR L=1 TO 3:FOR T=1 TO 10:FOR S=
60 TO 80:SOUND 0,S,10,8:NEXT S:POKE 71
2,T*10:NEXT T:SOUND 0,0,0,0
3430 POKE 712,0:FOR T=1 TO 500:NEXT T:
NEXT L
3440 GRAPHICS 18:PRINT #6, AT(0,5);"TH
ANKS FOR THE GAME!":FOR T=1 TO 4000:NE
XT:END
```

This completes the program, which is capable of running as it is. Below are different ways these routines can be programmed.

The coding on the animated eye routine can be reduced by changing line 3150 to

```
3150 FOR T=1 TO 6:READ DI$(T):NEXT:EYE
$="[space/ctrl T/3 spaces/ctrl T/space
]"
```

and lines 3240, 3250, and 3330 to

```
3240 FOR EY=1 TO 2:FOR EYE=17 TO 15 ST
EP-1:PRINT AT(EYE,8);EYE$:FOR T=1 TO 1
00:NEXT T:NEXT EYE
3250 FOR EYE=15 TO 17:PRINT AT(EYE,8);
EYE$:FOR T=1 TO 100:NEXT T:NEXT EYE
3330 NEXT EY:IF PEEK(764)<>255 THEN 33
60
```

In direct mode, type DEL 3260-3320 and hit RETURN. Now you have a much more compact code that does the same animation routine.

The coding in lines 1230 through 1440 can also be reduced. Below is the new vocabulary routine:

First, in line 3120, change to

```
3120 DIM I$(23),I(23),DI$(6),M(3),VOC$
(31):D1=0:D2=0:D3=0:D4=0:D5=0:D6=0:D7=
0:D8=0:G1=0:KH=0:LD=0
```

Next, add the following lines:

```
3155 FOR T=1 TO 31:READ VOC$(T):NEXT
3195 DATA OG,LNU,EPO,MUJ,ILC,VIG,ORD,R
HT,TEG,KAT,OOL,AXE,LEH,AER,IRD,VAW,LIK
,GIF,EPS,YAS,EEF,UOT,VNI,TXE,GIL
3197 DATA IRW,GIS,UOP,GID,SNI,ESU
```

In direct mode, type DEL 1230-1430 and hit RETURN. Then add the following lines:

```
1225 IF C1$=" OG" THEN C1$="OG"
1230 T1=0:FOR T=1 TO 31:IF C1$=VOC$(T)
 THEN T1=T:T=31
1240 NEXT:IF T1=0 THEN 1440
1250 IF T1<17 THEN ON T1 GOTO 1450,155
0,1550,1620,1680,1720,1720,1720,1850,1
850,1810,1810,1300,1950,2220,2280
1260 ON T1-16 GOTO 2330,2330,2460,2460
,2540,2540,2580,1310,1320,2630,2630,13
30,1340,2770,2770
1300 M$=".GNIDAER YB TOL A NRAEL ELPOE
P":GOTO 1010
1310 IF I(14)=0 AND DK=1 THEN DK=0:GOT
O 2840
1315 GOTO 1440
1320 IF I(1)=0 AND I(14)=0 AND DK=0 TH
EN DK=1:I$(14)="HCROT GNIBRUB":GOTO 28
40
```

```
1325 GOTO 1440
1330 IF RIGHT$(I$(10),8)="FO SSALG" TH
EN I$(10)="SSALG":GOTO 2680
```

This completes the two modifications to improve the program. The modification of the reading routine is still left to complete. In direct mode, type DEL 1950-2210, hit RETURN, then type in the following changes:

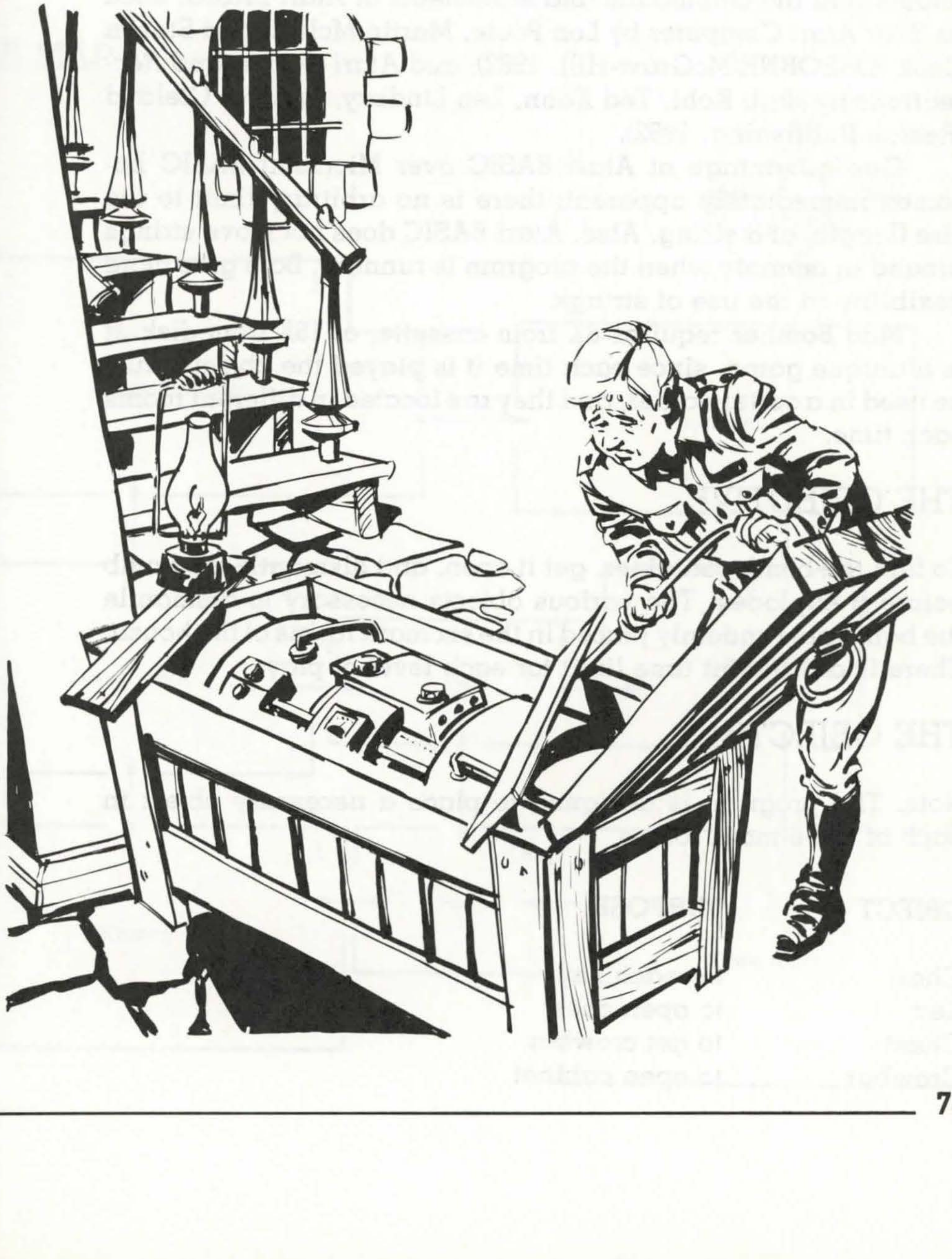
```
1950 IF C2$="RCS" THEN 1959
1952 IF C2$="CID" THEN 1970
1954 IF C2$="OOB" THEN 1980
1955 IF C2$="IRW" THEN 2160
1956 IF C2$="GIS" THEN 2155
1957 IF C2$="ARG" THEN 2170
1958 M$="!!!TAHT DAER T'NAC I":GOTO 10
10
1959 IF I(3)=0 AND I(17)=0 THEN M$="'W
ORRA HCUOT' :SYAS LLORCS":GOTO 1010
1960 IF I(3)=0 THEN M$=".EM OT KEERG S
'TI":GOTO 1010
1965 GOTO 2200
1970 IF I(17)=0 THEN M$="'YRANOITCID K
EERG' :SYAS REVOC":I$(17)=I$(17)+" KEE
RG":GOTO 1010
1975 GOTO 2200
1980 IF R=6 THEN 2140
1990 IF I(9)<>0 THEN 2200
2000 IF I(13)=0 THEN 2100
2010 IF R=9 THEN M$=".TOH NEHW SPLEH D
LOC GNIHTEMOS":GOTO 1010
2020 IF R=3 THEN M$=".LATEM TAE SEVISO
RROC":GOTO 1010
2030 IF R=4 THEN M$=".DOOW FO EDAM ERA
 SEGDIRB":GOTO 1010
2040 IF R=12 THEN M$=".CIGAM ERA SNOGA
RD":GOTO 1010
2050 IF R=29 THEN M$=".TNATROPMI SI NO
ITACINUMMOC":GOTO 1010
2060 IF R=5 THEN M$=".LUOF LLEMS YLLAU
SU SLACIMEHC":I(6)=4:I(8)=5:GOSUB 20:G
OSUB 3510:GOTO 2880
2070 IF R=30 THEN M$="'EGDELWONK FO KO
```

```
OB' :SYAS REVOC":GOTO 1010
2080 IF R=25 AND I(18)=0 THEN M$=".KNI
RD GNILOOC A":GOTO 1010
2090 M$=".TI NI NOITAMROFNI YNA DNIF T
'NAC I":GOTO 1010
2100 IF Q=0 THEN M$="!YTPME SAW MOOR E
HT":Q=1:GOTO 1010
2110 IF Q=1 THEN M$="!OG UOY DNUOR DNA
 DNUOR":Q=2:GOTO 1010
2120 M$="!!WOL DNA HGIH KOOL":GOTO 101
0
2140 IF I(18)=0 THEN M$=".GNIBMILC ROF
 PLEH DEEN ELPOEP":GOTO 1010
2150 M$="'.NI NGIS TSUM STSEUG LLA':SY
AS KOOB":GOTO 1010
2155 IF R=45 OR R=19 THEN M$=".LLEC NI
 DEWOLLA TON ERA SNOISSESSOP":GOTO 101
0
2156 GOTO 2210
2160 IF R=8 THEN C4$="":C3$=NA$:GOSUB
30:NB$=C4$:M$="...EHT NI OG T'NOD "+NB
$:GOTO 1010
2165 GOTO 2210
2170 IF R=50 THEN M$="!!DEID EW DNA DE
IRT EW":GOSUB 20:GOSUB 3490:M$="EVAD &
 KCAJ DENGIS":GOTO 1010
2180 IF R=14 THEN M$=".SPLEH EGDELWONK
 FO KOOB EHT":GOTO 1010
2190 GOTO 2210
2200 M$="!!TI EVAH T'NOD I":GOTO 1010
2210 M$="!EREH TON S'TI":GOTO 1010
```

You now have a complete modified version of the program. Be sure to save your new version onto tape with a CSAVE command or onto disk with SAVE "D:filename."

AN ADVENTURE IN
ATARI BASIC:

MAD BOMBER



Mad Bomber is written in Atari BASIC, the language cartridge which is sold separately as part number CXL4002.

I recommend the following sources of information on the Atari BASIC language provided with the computer: *Atari BASIC* by Bob Albrecht, LeRoy Finkel, and Jerald R. Brown (Wiley and Sons, 1979); and *BASIC Reference Manual* by Shaw and Brewster (Atari, 1980). There are also many other fine books to help you understand the commands and statements of Atari BASIC, such as *Your Atari Computer* by Lon Poole, Martin McNiff, and Steven Cook (OSBORNE/McGraw-Hill, 1982); and *Atari Games and Recreations* by Herb Kohl, Ted Kahn, Len Lindsay, and Pat Cleland (Reston Publishing, 1982).

One advantage of Atari BASIC over Microsoft BASIC becomes immediately apparent: there is no arbitrary limit to the size (length) of a string. Also, Atari BASIC does not move strings around in memory when the program is running. Both give more flexibility in the use of strings.

Mad Bomber requires 8K from cassette, or 16K from disk. It is a unique game, since each time it is played the objects must be used in a certain order, and they are located in different rooms each time.

THE OBJECTIVE

To find the bomb container, get it open, and dismantle the bomb before it explodes. The various objects necessary to dismantle the bomb are randomly placed in the six main rooms of the house. There is a different time limit for each level of play.

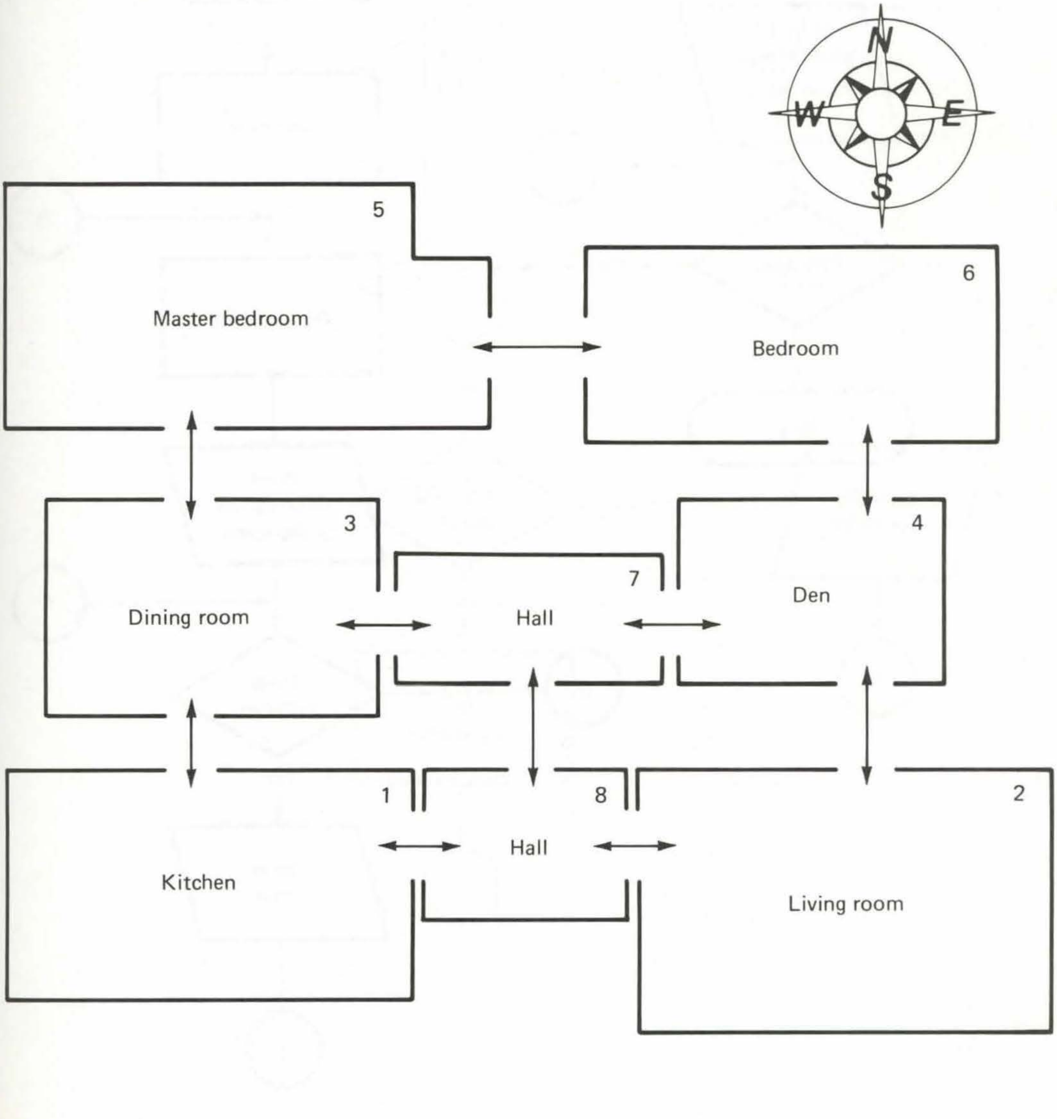
THE OBJECTS

Note: The program is designed to place a necessary object in each of the 6 main rooms.

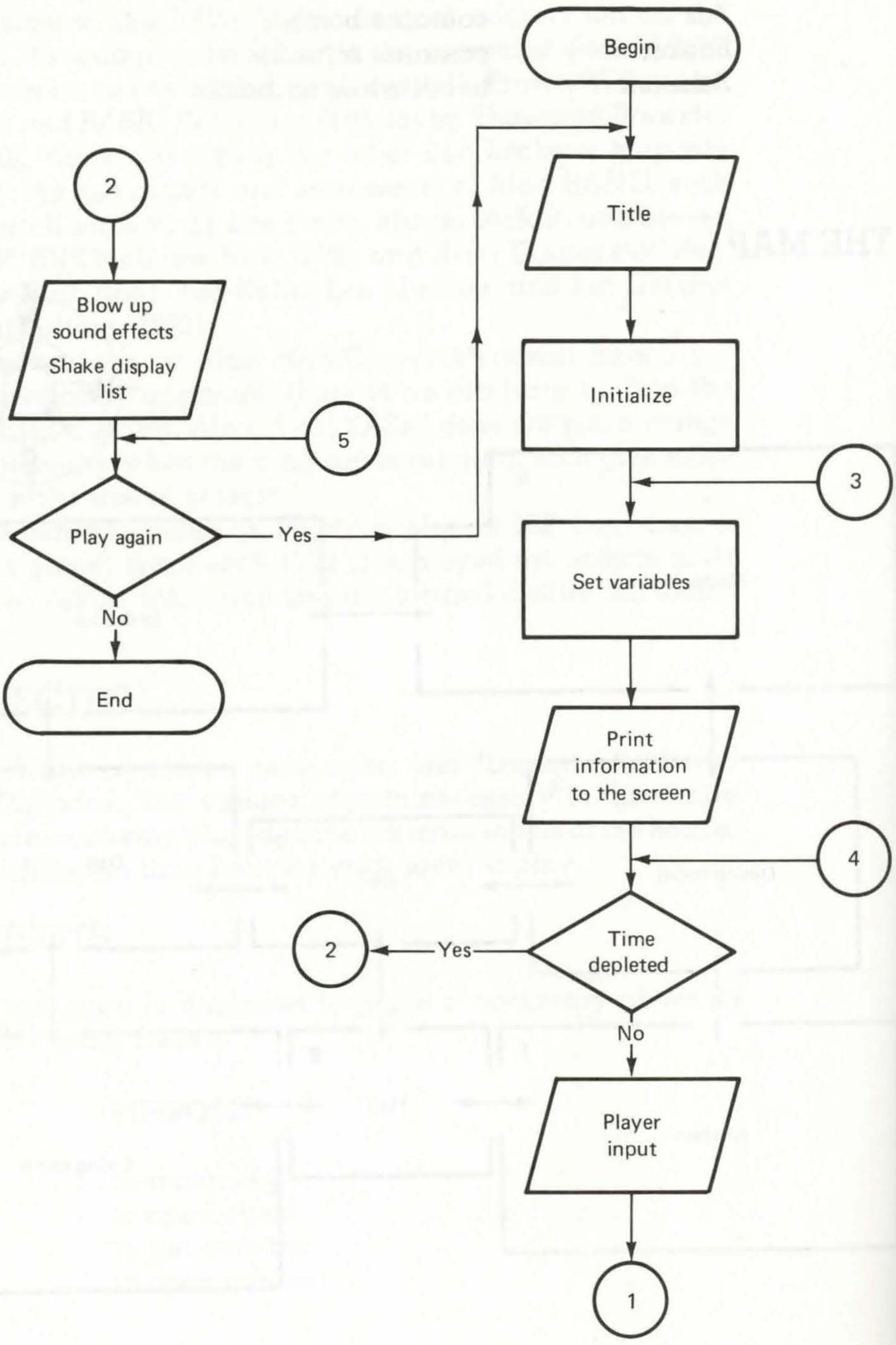
OBJECT	**PURPOSE**
Chair	to reach key
Key	to open chest
Chest	to get crowbar
Crowbar	to open cabinet

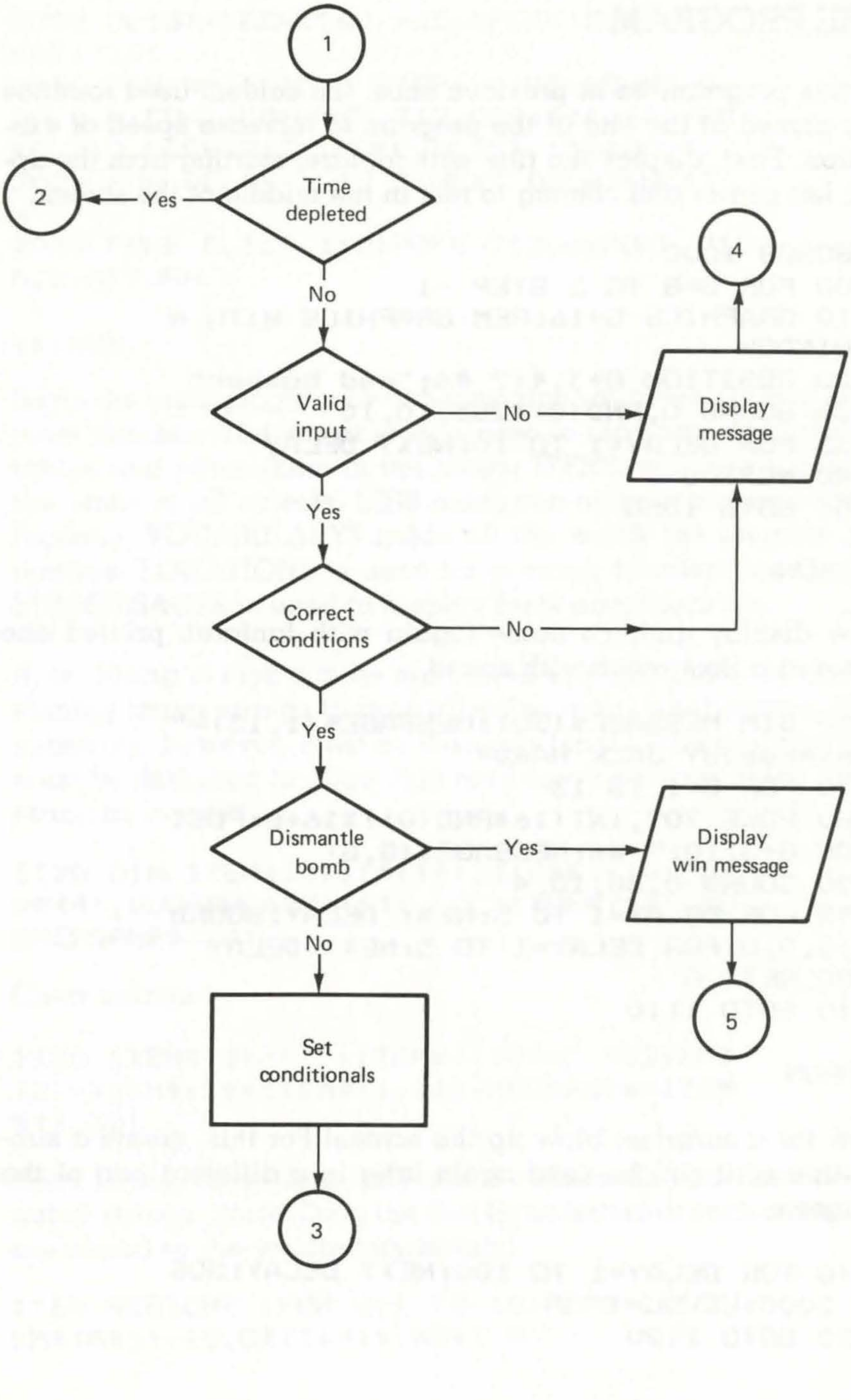
Cabinet	to get screwdriver
Screwdriver	to open box
Box	contains bomb
Basket	contains scissors
Scissors	to cut wires on bomb.

THE MAP



THE FLOWCHART




1
Time depleted
Yes
2
No
Valid input
No
Display message
4
Yes
Correct conditions
No
Yes
Dismantle bomb
Yes
Display win message
5
No
Set conditionals
3

THE PROGRAM

In this program, as in previous ones, the seldom-used routines are placed at the end of the program to increase speed of execution. First, display the title with fanfare, starting from the upper left corner and coming to rest in the middle of the screen.

```
5 GOSUB 1000
1000 FOR G=8 TO 2 STEP -1
1010 GRAPHICS G+16:REM GRAPHICS WITH N
O WINDOW
1020 POSITION G+3,4:? #6;"mad bomber"
1030 SOUND 0,RND(0)*255,10,10
1035 FOR DELAY=1 TO 10:NEXT DELAY
1040 NEXT G
1050 GOTO 1050
```

```
(*)RUN
```

Now display author's name (again with fanfare), printed one letter at a time, each with sound.

```
1050 DIM MESSAGE$(30):MESSAGE$(1,13)="
[inverse]BY JACK HARDY"
1070 FOR G=1 TO 13
1080 POKE 709,INT(16*RND(0))*16+8:POSI
TION G+3,10:? #6;MESSAGE$(G,G)
1090 SOUND 0,80,10,4
1095 FOR DELAY=1 TO 5:NEXT DELAY:SOUND
 0,0,0,0:FOR DELAY=1 TO 5:NEXT DELAY
1100 NEXT G
1110 GOTO 1110
```

```
(*)RUN
```

Now for a surprise: blow up the screen! For this, create a subroutine so it can be used again later in a different part of the program.

```
1110 FOR DELAY=1 TO 100:NEXT DELAY:GOS
UB 2000:LEVEL=1
1120 GOTO 1120
```

```
2000 DLIST=PEEK(560)+256*PEEK(561):G=P
EEK(712)
2010 FOR W=15 TO 0 STEP -0.25:SOUND 0,
50,0,W:CO=1-CO:POKE 712,CO*(4*16+6):PO
KE 710,PEEK(712):POKE DLIST,112*CO
2020 FOR DELAY=1 TO 5:NEXT DELAY:NEXT
W
2030 POKE DLIST,112:POKE 712,G:POKE 71
0,G:RETURN
```

```
(*)RUN
```

Begin the main program by dimensioning variables. ITEM(n) holds room number. The variable I$ is used to read objects from data tables and place them in the longer ITEM$, which will contain the name of all objects. DIR$ holds one-letter commands for directions. VOCABULARY$ holds all the words the program recognizes. LOCATION$ is used for printing location description. SUBMESSAGE$ is used to display facts about location.

Note: String arrays can be simulated in Atari BASIC by dimensioning larger strings to store all string data as substrings. Each substring, however, must be the same length. The program then must be designed to allow it to read small portions of the larger string as needed.

```
1120 DIM ITEM(10),I$(14),ITEM$(110),DI
R$(4),VOCABULARY$(54),LOCATION$(14),SU
BMESSAGE$(30)
```

Clear strings:

```
1130 ITEM$(1)=" ":ITEM$(110)=" ":ITEM$
(2)=ITEM$:I$=ITEM$(1,11):MESSAGE$=ITEM
$(1,30)
```

Read items, directions, and game commands into their designated strings. (*Note:* Only the first three letters of each command are stored in the vocabulary string.)

```
1160 RESTORE :FOR G=1 TO 10:READ I$:IT
EM$(G*11-10,G*11)=I$:NEXT G
```

```
1170 DIR$="NSEW"
1180 FOR G=1 TO 18:READ I$:VOCABULARY$
(G*3-2,G*3)=I$(1,3):NEXT G
4000 DATA CHAIR,KEY,CHEST,CABINET,BOX,
BASKET,CROWBAR,SCREWDRIVER,BOMB,SCISSO
RS
4010 DATA WALK,RUN,GO ,GET,TAKE,LOOK,E
XAMINE,OPEN,UNSCREW,UNLOCK,DROP,PUT,ST
AND ON,CLIMB,TOUCH,FEEL,INVENTORY,CUT
```

Assign each item a room number, then shuffle them so they do not appear in the same room in successive games.

```
1190 FOR G=1 TO 6:ITEM(G)=G:NEXT G:FOR
 G=7 TO 10:ITEM(G)=0:NEXT G
1200 FOR G=6 TO 2 STEP -1:J=INT(G*RND(
0))+1:I=ITEM(J):ITEM(J)=ITEM(G):ITEM(G
)=I:NEXT G
```

Set variables. Delete characters from I$, set Variable CHAIR to 0, and establish the number of the beginning room.

```
1210 I$="":CHAIR=0:ROOM=8
```

This part of the program allows the player to choose the level of play and to begin when he or she wishes.

```
1220 POSITION 2,8:PRINT #6;"[inverse]p
ress"
1230 POSITION 2,9:PRINT #6;"START TO B
EGIN"
1235 POSITION 2,10:PRINT #6;"SELECT /
LEVEL ";LEVEL
1240 IF PEEK(53279)=6 THEN 1270
1250 IF PEEK(53279)=5 THEN LEVEL=LEVEL
+1:IF LEVEL>3 THEN LEVEL=1
1260 FOR DELAY=1 TO 10:NEXT DELAY:GOTO
 1235
1270 IF LEVEL=1 THEN MAXTIME=15
1280 IF LEVEL=2 THEN MAXTIME=5
1290 IF LEVEL=3 THEN MAXTIME=2

(*)RUN
```

Note: You may run the program and watch how the SELECT key functions, but if you push START, the program will error.

Now add the routine to start the program. Change to a graphics 0 mode, change the screen color to black, make the cursor invisible, position and print the message, set the internal clocks to 0, and return to the beginning of the program.

```
1300 GRAPHICS 0:POKE 710,0:POKE 752,1
1310 POSITION 11,11:PRINT "THE GAME IS
 AFOOT!":POKE 18,0:POKE 19,0:POKE 20,0
:RETURN
```

Create the sound of the pitter-patter of little feet, change screen color, and set direction variables to zero. (*Note:* The colors for each room will remain the same since they are set by the room number.)

```
7 FOR G=1 TO 10:SOUND 0,80,2,4:FOR D=1
 TO 2:NEXT D:SOUND 0,0,0,0:FOR D=1 TO
55:NEXT D:NEXT G
8 POKE 710,ROOM*16+2:POKE 712,PEEK(710
):N=0:S=0:E=0:W=0
```

The program checks the value of the variable ROOM and goes to the corresponding line number (e.g., Room 1=line 10, Room 2=line 20, etc.). Each room sets its direction variables to the room number located in every direction.

```
9 ON ROOM GOTO 10,20,30,40,50,60,70,80
10 LOCATION$="KITCHEN":N=3:E=8:GOTO 10
0
20 LOCATION$="LIVING ROOM":N=4:W=8:GOT
O 100
30 LOCATION$="DINING ROOM":S=1:N=5:
E=7:GOTO 100
40 LOCATION$="DEN":S=2:N=6:W=7:GOTO 10
0
50 LOCATION$="MASTER BEDROOM":S=3:E=6:
GOTO 100
60 LOCATION$="BEDROOM":S=4:W=5:GOTO 10
0
70 LOCATION$="HALL WAY":S=8:W=3:E=4:GO
TO 100
```

```
80 LOCATION$="HALL WAY":N=7:W=1:E=2:GO
TO 100
```

To begin the PRINT routine, format screen, print description of location, give directions, and print items in the room. *Note:* All items are not the same in length, so any trailing spaces are deleted by the subroutine in lines 270 and 272. To see this, run the program, hit BREAK then, in direct mode, type PRINT ITEM$ and hit RETURN. This can also be used to "debug" your program. By using this simple command to PRINT variable, you can immediately find the value of any variable in the program.

```
100 PRINT "[esc ctrl clear]"
110 PRINT "[inverse / 37 ctrl R's]"
120 PRINT :PRINT "YOU'RE IN THE ";LOCA
TION$
130 PRINT :PRINT "[inverse / 18 ctrl G
's / 19 ctrl F's]"
```

Print exits:

```
140 PRINT "[inverse / 5 ctrl F's /spac
e]THERE ARE DOORWAYS GOING:[space/5 ctrl
 G's]"
150 PRINT :IF N>0 THEN PRINT "NORTH, "
;
160 IF S>0 THEN PRINT "SOUTH, ";
170 IF E>0 THEN PRINT "EAST, ";
180 IF W>0 THEN PRINT "WEST, ";
190 PRINT "[2 esc ctrl +'s]"
```

Print items the player can see:

```
230 PRINT :PRINT "[inverse / 11 ctrl W
's / 15 ctrl R's / 11 ctrl W's]"
240 PRINT "[inverse / 11 ctrl X's / sp
ace ]ITEMS YOU SEE[space / 11 ctrl X's
]"
250 PRINT :FOR G=1 TO 10
260 IF ITEM(G)=ROOM THEN I$=ITEM$(G*11
-10,G*11):GOSUB 270:PRINT I$;", ";:I$=
""
```

```
265 NEXT G:GOTO 275
270 IF I$(LEN(I$),LEN(I$))=" " THEN I$
=I$(1,LEN(I$)-1):GOTO 270
272 RETURN
```

This is the area prepared for user input:

```
275 PRINT "[2 esc ctrl +’s]"
280 PRINT :PRINT "[inverse / 36 ctrl L
’s / ctrl U]"
290 PRINT "[inverse / 12 spaces]INSTRU
CTIONS?[12 spaces]":PRINT
```

Before player can input a message, the program checks the time to see how much is used and if bomb should explode. If time is up, control of program is sent to line 950 for explosion routine. (This line will be added later.)

```
294 TIME=PEEK(19)/14:IF TIME>=MAXTIME
THEN 950
296 IF TIME>=MAXTIME-1 THEN SOUND 1,9,
2,1
```

Set TRAP and clear old strings:

```
300 TRAP 3000:MESSAGE$="":SUBMESSAGE$=
""
```

Now let player input a message, then check to see if it is only one letter; if so, check for valid Directional command in DI$. Upon finding a direction, the program will go to line 340 and check to see if player can indeed go in that direction.

```
310 INPUT MESSAGE$:IF LEN(MESSAGE$)>1
THEN 400
320 FOR G=1 TO 4:IF DIR$(G,G)=MESSAGE$
 THEN DIR=G:POP :GOTO 340
330 NEXT G:PRINT "I DIDN’T UNDERSTAND
THAT!":GOTO 3050
340 ON DIR GOTO 350,360,370,380
350 IF N>0 THEN ROOM=N:GOTO 7
355 GOTO 390
```

```
360 IF S>0 THEN ROOM=S:GOTO 7
365 GOTO 390
370 IF E>0 THEN ROOM=E:GOTO 7
375 GOTO 390
380 IF W>0 THEN ROOM=W:GOTO 7
390 PRINT "YOU CAN'T GO IN THAT DIRECT
ION!":GOTO 3050
```

If player inputs wrong length message or no message at all, the program will error, sending control to the routine below (TRAP 3000). The program pauses for appropriate message to be read and then jumps back for more input.

```
3000 PRINT "PLEASE SAY THAT AGAIN!":TR
AP 40000
3050 FOR G=1 TO 200:NEXT G:PRINT "[3 e
sc ctrl -'s / 6 esc shift insert]":GOT
O 300
```

This routine separates the player's input into two strings (MESSAGE$ and SUBMESSAGE$). Check the time again after player's input, check the first three letters of the player's input, and, upon finding a match, set the variable VOC and jump to 440 to send the program to the proper routine for that word.

```
400 FOR G=1 TO LEN(MESSAGE$):IF MESSAG
E$(G,G)=" " THEN SUBMESSAGE$=MESSAGE$(
G+1,LEN(MESSAGE$)):POP :GOTO 415
410 NEXT G
415 IF TIME>=MAXTIME THEN 950
420 FOR G=1 TO 18:IF MESSAGE$(1,3)=VOC
ABULARY$(G*3-2,G*3) THEN VOC=G:POP :GO
TO 440
430 NEXT G:PRINT "THAT IS NOT PART OF
MY VOCABULARY!":GOTO 3050
440 ON VOC GOTO 450,450,450,470,470,52
0,520,600,600,600,670,670,710,710,730,
730,770,830
```

The routine for the words WALK, RUN, and GO are checked in the vocabulary routine by seeing first if the player wants to go in a direction, and if not, whether he or she is permitted to go elsewhere (i.e., onto Chair).

```
450 FOR G=1 TO 4:IF SUBMESSAGE$(1,1)=D
IR$(G,G) THEN DIR=G:POP :GOTO 340
455 IF SUBMESSAGE$(1,3)="CHA" THEN 710
460 NEXT G:PRINT "YOU CAN'T GO THERE!"
:GOTO 3050
```

Following is the routine for the words GET and TAKE. First, the program checks for a valid object which can be taken. Second, it checks to see if player already has the object. Third, it checks to see if it is indeed in the room with the player. Fourth, it checks to see if the player wants the KEY, since the chair is needed in order to reach it. Finally, it gives the player the object requested.

```
470 FOR G=1 TO 10:IF SUBMESSAGE$(1,3)=
ITEM$(G*11-10,G*11-8) THEN POP :GOTO 4
90
480 NEXT G:PRINT "YOU CAN'T TAKE A ";S
UBMESSAGE$:GOTO 3050
490 IF ITEM(G)=10 THEN PRINT "YOU ALRE
ADY HAVE IT!":GOTO 3050
500 IF ITEM(G)<>ROOM THEN PRINT "THAT
ITEM IS NOT HERE!":GOTO 3050
510 IF SUBMESSAGE$="KEY" AND (CHAIR<>1
 OR ITEM(1)<>ROOM) THEN 518
515 ITEM(G)=10:GOTO 9
518 PRINT "IT'S TOO HIGH. YOU NEED TO
STAND ON   SOMETHING![esc ctrl -]":GOT
O 3050
```

Create the routines for the words LOOK and EXAMINE. Check each item to see if it is a valid object to be looked at and then jump to the proper response.

```
520 FOR G=1 TO 10:IF SUBMESSAGE$(1,3)=
ITEM$(G*11-10,G*11-8) THEN POP :GOTO 5
40
530 NEXT G:PRINT "IT LOOKS LIKE A ";SU
BMESSAGE$;" TO ME!":GOTO 3050
540 IF ITEM(G)<>ROOM AND ITEM(G)<>10 T
HEN PRINT "IT'S NOT HERE TO LOOK AT!":
GOTO 3050
550 IF SUBMESSAGE$(1,3)="CHE" THEN PRI
NT "THERE IS A KEYHOLE!":GOTO 3050
```

```
560 IF SUBMESSAGE$(1,3)="CAB" THEN PRI
NT "IT WILL HAVE TO BE FORCED OPEN!":G
OTO 3050
570 IF SUBMESSAGE$(1,3)="BOX" THEN PRI
NT "SCREWS HOLD THE LID CLOSED!":GOTO
3050
580 IF SUBMESSAGE$(1,3)="BAS" THEN PRI
NT "YOU FIND A PAIR OF SCISSORS!":ITEM
(10)=ROOM:FOR D=1 TO 200:NEXT D:GOTO 9
585 IF SUBMESSAGE$(1,3)="BOM" THEN PRI
NT "THE WIRES NEED TO BE CUT!":GOTO 30
50
590 PRINT "I SEE NOTHING SPECIAL!":GOT
O 3050
```

Below are the routines for the words OPEN, UNSCREW, and UNLOCK. Only the items which can be opened are checked—Items 3 through 6 (Chest, Cabinet, Box and Basket). Note that they follow each other in the DATA statements. If the item can be opened, jump to the proper routine for that word and, if the proper tool is in the possession of the player, allow the object to be opened.

```
600 FOR G=3 TO 6:IF SUBMESSAGE$(1,3)=I
TEM$(G*11-10,G*11-8) THEN POP :GOTO 61
0
605 NEXT G:PRINT "YOU CAN'T OPEN THAT!
":GOTO 3050
610 IF ITEM(G)<>ROOM THEN PRINT "IT'S
NOT IN THIS ROOM!":GOTO 3050
612 IF ITEM(G+4)<>0 THEN PRINT "IT'S A
LREADY OPEN!":GOTO 3050
620 IF SUBMESSAGE$(1,3)="CHE" AND ITEM
(2)=10 THEN ITEM(7)=ROOM:GOTO 9
630 IF SUBMESSAGE$(1,3)="CAB" AND ITEM
(7)=10 THEN ITEM(8)=ROOM:GOTO 9
640 IF SUBMESSAGE$(1,3)="BOX" AND ITEM
(8)=10 THEN ITEM(9)=ROOM:GOTO 9
650 IF SUBMESSAGE$(1,3)="BAS" THEN PRI
NT "OKAY!":GOTO 3050
660 PRINT "YOU NEED HELP TO OPEN THAT!
":GOTO 3050
```

Note: In line 612, the object is checked to see if it is already open by checking ITEM(G + 4) to see if it has been assigned a number. This is the way it works: If you look at the data statement in line 4000 you will note that the CHEST contains the CROWBAR, the CABINET contains the SCREWDRIVER, etc., and that each of them is 4 items away from its container. At the beginning of the program, the status number of each was set to zero, so that none would appear until its container was opened.

Further routines for words DROP and PUT check to see what item the player wants to drop and, if he or she has it, places it in the room.

```
670 FOR G=1 TO 10:IF SUBMESSAGE$(1,3)=
ITEM$(G*11-10,G*11-8) THEN POP :GOTO 6
90
680 NEXT G:PRINT "DROP WHAT!!!!":GOTO
3050
690 IF ITEM(G)=10 THEN ITEM(G)=ROOM:GO
TO 9
700 PRINT "YOU'RE NOT CARRYING IT!!":G
OTO 3050
```

The words STAND ON and CLIMB are checked to see if player wants to stand on the chair and, if so, it goes there.

```
710 IF SUBMESSAGE$(1,3)="CHA" THEN 722
715 IF MESSAGE$(1,3)="STA" AND SUBMESS
AGE$(4,6)="CHA" THEN 722
720 PRINT "YOU CAN'T ";MESSAGE$(1,5);"
 ON THAT!!":GOTO 3050
722 IF ITEM(1)<>ROOM THEN PRINT "IT'S
NOT THERE!":GOTO 3050
725 CHAIR=1:GOSUB 3060:GOTO 3050
```

The necessity of a new subroutine was created above:

```
3060 PRINT "OKAY!":RETURN
```

Now enter the routine for the words TOUCH and FEEL:

```
730 FOR G=1 TO 10:IF SUBMESSAGE$(1,3)=
ITEM$(G*11-10,G*11-8) THEN POP :GOTO 7
50
740 NEXT G:PRINT "IT FEELS LIKE A ";SU
BMESSAGE$;" TO ME!":GOTO 3050
750 IF ITEM(G)<>ROOM THEN PRINT "IT'S
NOT HERE TO TOUCH!":GOTO 3050
755 IF SUBMESSAGE$(1,3)="BOM" THEN PRI
NT "I FEEL SOME WIRES!":GOTO 3050
760 GOSUB 3060:GOTO 3050
```

When the player asks for INVENTORY, the program comes here and checks to see what items are in the player's possession. (*Note:* When ITEM(n) = 10 the player has the item in possession.)

```
770 PRINT "[esc shift clear]":PRINT "[
inverse / 12 ctrl O's]ITEMS YOU HAVE[1
2 ctrl I's]":PRINT :FOR G=1 TO 10
780 IF ITEM(G)=10 THEN PRINT ITEM$(G*1
1-10,G*11)
790 NEXT G
```

The way to get back to the main program is shown below:

```
800 POSITION 5,22:PRINT "TOUCH [invers
e / space]SPACE BAR[space / inverse of
f] TO CONTINUE"
810 OPEN #1,4,0,"K:":GET #1,K:IF K<>32
 THEN CLOSE #1:GOTO 800
820 CLOSE #1:GOTO 9
```

The last word in the program's vocabulary is CUT:

```
830 IF SUBMESSAGE$(1,3)<>"WIR" THEN PR
INT "I CAN'T CUT THAT!":GOTO 3050
840 IF ITEM(9)<>ROOM THEN PRINT "THE B
OMB IS NOT HERE!":GOTO 3050
845 IF ITEM(10)<>10 THEN PRINT "YOU DO
N'T HAVE ANY SCISSORS!":GOTO 3050
```

If player has the scissors, he or she can cut the wire on the bomb and the program continues on to the win routine.

```
850 GRAPHICS 18:POSITION 4,1:PRINT #6;
"[inverse]YOU DID IT!"
860 RESTORE 4050:FOR D=1 TO 7:READ SO,
DELAY
870 SOUND 1,SO,10,10:POKE 712,SO:FO
R D2=1 TO DELAY:NEXT D2:NEXT D:SOUND 1
,0,0,0
880 IF LEVEL=3 THEN POSITION 2,2:PRINT
 #6;"YOU SURVIVED THE":POSITION 3,3:PR
INT #6;"HIGHEST LEVEL":GOTO 965
890 LEVEL=LEVEL+1
900 POSITION 2,3:PRINT #6;"YOU HAVE BE
EN":POSITION 1,4:PRINT #6;"PROMOTED TO
 LEVEL ";LEVEL
910 FOR DELAY=1 TO 100:NEXT DELAY
920 GOSUB 1130:GOTO 7
```

To have music to accompany the above routine, add the following data:

```
4050 DATA 96,15,72,15,57,15,48,30,57,1
5,48,60,0,0
```

Create the routine to be used when the time limit is reached:

```
950 SOUND 1,0,0,0:GOSUB 2000:FOR DELAY
=1 TO 20:NEXT DELAY
960 GRAPHICS 18:POSITION 1,3:PRINT #6;
"[inverse]you were too late!"
```

Both win and lose routines come here to see if player wants another chance to dismantle the bomb.

```
965 POSITION 1,5:PRINT #6;"would you l
ike to":POSITION 2,6:PRINT #6;"try aga
in? (y/n)"
970 OPEN #1,4,0,"K:":GET #1,K:IF K<>78
 AND K<>89 THEN CLOSE #1:GOTO 970
980 CLOSE #1:IF K=89 THEN GOSUB 1130:G
OTO 7
990 POSITION 1,8:PRINT #6;"[inverse]TH
ANKS FOR PLAYING":FOR DELAY=1 TO 250:N
EXT DELAY:END
```

(*) RUN

You now have a completed working program. Examine the coding to see what can be done to improve the techniques.

Begin with the rooms. A much more efficient way to handle these routines would be to enter them into a string entitled LOCATION$. Since there is already a string by that name, it can be used if it is enlarged (re-dimensioned). To enlarge LOCATION$ change the following:

```
1120 DIM ITEM(10),I$(14),ITEM$(110),DI
R$(4),VOCABULARY$(54),LOCATION$(112),S
UBMESSAGE$(30)
```

LOCATION$ is now dimensioned to 112 instead of to 14, as it was in the first program. It can now hold all of the room descriptions at once.

Since a line for each location will no longer be used, another way must be created to hold the values of all connecting locations. Add the following lines:

```
1125 DIM N(8),S(8),E(8),W(8)
```

Create a more efficient way to get the information into the LOCATION$ and direction variables N,E,S,W. First, clear the string:

```
1140 LOCATION$(1)=" ":LOCATION$(112)="
 ":LOCATION$(2)=LOCATION$
```

Then add the data statements and a line to read them:

```
1185 FOR G=1 TO 8:READ I$,D1,D2,D3,D4:
LOCATION$(G*14-13,G*14)=I$:N(G)=D1:S(G
)=D2:E(G)=D3:W(G)=D4:NEXT G


4020 DATA KITCHEN,3,0,8,0,LIVING ROOM,
4,0,0,8,DINING ROOM,5,1,7,0,DEN,6,2,0,
7
4030 DATA MASTER BEDROOM,0,3,6,0,BEDRO
OM,0,4,0,5,HALL WAY,0,8,4,3,HALL WAY,7
0,2,1
```

Now, delete lines 10 through 80. This can easily be done by typing only the line number and hitting RETURN.

Change the following:

```
9 N=N(ROOM):S=S(ROOM):E=E(ROOM):W=W(RO
OM)
120 PRINT :PRINT "YOU'RE IN THE ";LOCA
TION$(ROOM*14-13,ROOM*14)
```

Compare the above routine with the old routine for finding and printing the room information.

A few other things can be done to make the coding more efficient: In lines 150 through 180, delete the >0 from those routines. Now do the same thing to lines 350, 360, 370, 380. If you would like to know exactly how those lines look after the change, see the complete listing in Appendix A. If you're also wondering how this can work, just remember that the IF/THEN statement checks for either true or false by comparing numbers within variables. Anything other than zero in Atari BASIC is considered by the program to be true. If the variable is zero, it is false, and the remainder of the program line will not be executed.

Now, look at lines 520 through 590. One way to improve this routine is to add following line:

```
545 ON G GOTO 590,590,550,560,570,580,
590,590,585,590
```

Then change these lines:

```
550 PRINT "THERE IS A KEYHOLE!":GOTO 3
050
560 PRINT "IT WILL HAVE TO BE FORCED O
PEN!":GOTO 3050
570 PRINT "SCREWS HOLD THE LID CLOSED!
":GOTO 3050
580 PRINT "YOU FIND A PAIR OF SCISSORS
!":ITEM(10)=ROOM:FOR D=1 TO 200:NEXT D
:GOTO 9
```

```
585 PRINT "THE WIRES NEED TO BE CUT!":
GOTO 3050
```

Comparing the old and the new routines makes it obvious that the new is a much more efficient method of accomplishing the same task.

Add these lines to make the same conversions to lines 600 through 660:

```
615 ON G-2 GOTO 620,630,640,650
625 GOTO 660
635 GOTO 660
645 GOTO 660
```

And change these lines:

```
620 IF ITEM(2)=10 THEN ITEM(7)=ROOM:GO
TO 9
630 IF ITEM(7)=10 THEN ITEM(8)=ROOM:GO
TO 9
640 IF ITEM(8)=10 THEN ITEM(9)=ROOM:GO
TO 9
650 PRINT "OKAY!":GOTO 3050
```

Again, this is a much more efficient way to handle the routine.

The program is again complete, and a faster response to most of the input should be noticed. Be sure to save a copy of the new program onto cassette with a CSAVE command, or onto disk with a SAVE "D:filename."

PART 3

Graphic Adventures

Microcomputers, such as the Atari, are becoming increasingly more graphics-oriented. With these capabilities, many programs can be enhanced. Adventure games in particular are improved, because now you can see the room and its contents. The thrill of exploration and discovery can now become visual as well as imaginary. Along with these enhancements comes the challenge for the programmer to use and manipulate these graphic capabilities within his program.

The Atari has allowed easy access to most of its capabilities through either BASIC or PILOT. Access to many screen resolutions can be achieved with the use of its 12 graphics modes (0–11) and in many there is a palette of 128 colors from which to choose. Through mixing shades and luminance, some truly beautiful pictures can be created.

The next three programs will present some of the ways in which to utilize the graphic capabilities of your Atari through PILOT, Microsoft BASIC, and Atari BASIC.

A GRAPHIC ADVENTURE
IN ATARI PILOT:

ESCAPE

Escape is a graphics adventure written in Atari PILOT. Although PILOT was originally designed for text output only, it has been enhanced with Turtle Graphics, developed by Dr. Seymour Papert and the LOGO group at the Massachusetts Institute of Technology, which enables the program to draw pictures during its execution.

Escape is the result of a combined effort of all the members of my family. My six-year-old son, Jason, created routines for the dog and the gorilla, and the rest of us devised the other obstacles. The program requires 32K from cassette and 48K from disk.

THE OBJECTIVE

To escape from the top floor of a building. You awaken in a dark room. The last thing you remember is being hit on the head and thrown into a car. The only thing on your mind now is escape.

THE OBJECTS

Atari PILOT does not support arrays, so it is necessary to devise a way to store and check the status of objects in the adventure. For this, the program uses page 6, a favorite, usually untouched, area in memory. This area lies between 600 Hex (1536 decimal) and 700 Hex (1792 decimal) and is the place in which will be stored room numbers for some objects and status of others (door locked/unlocked, etc.).

Below is a listing of the objects in this adventure and the memory locations which hold their room numbers. Note that when a zero is stored in a memory location, the object is in the player's possession; when a number other than zero is stored there, the object is in the room of that number.

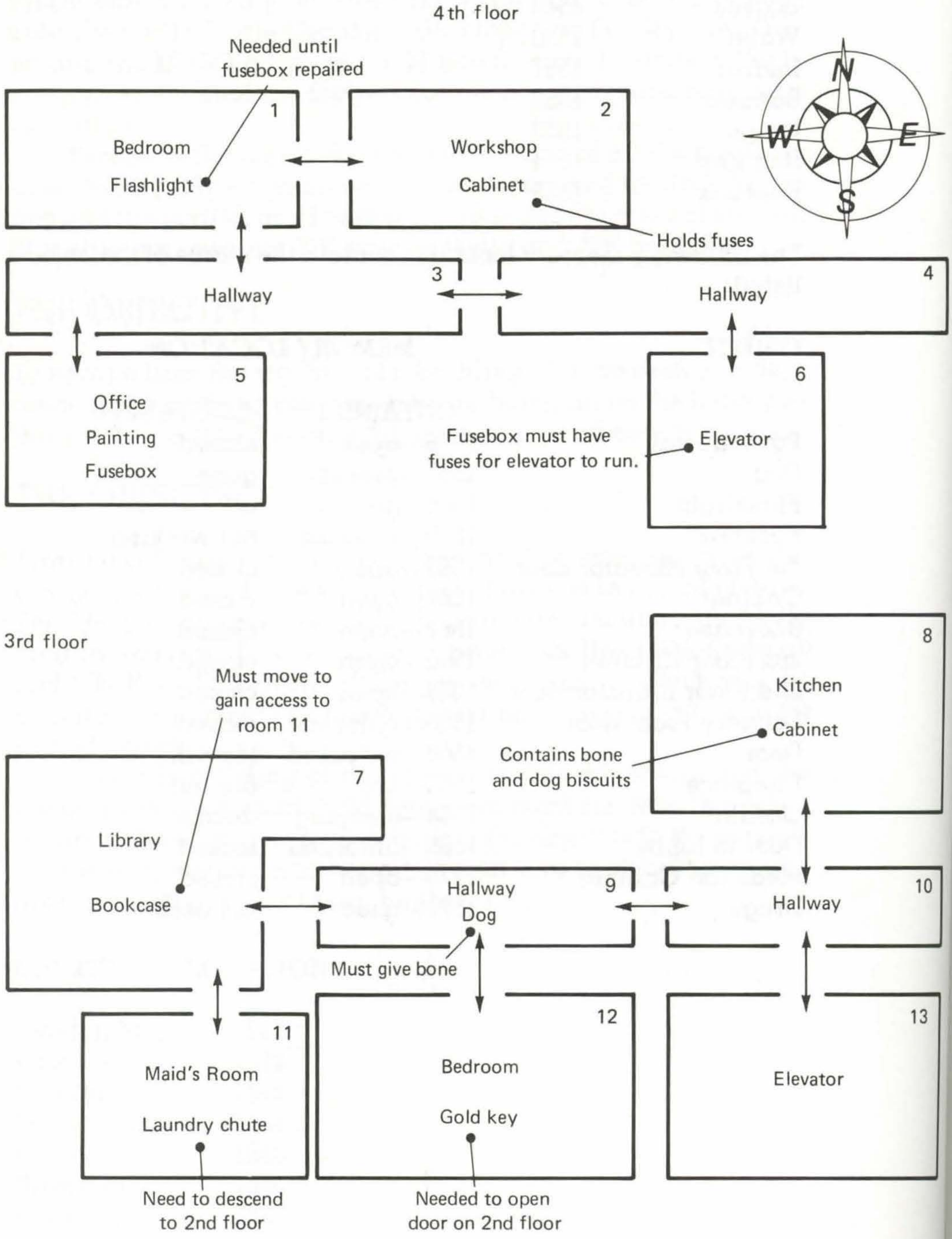
OBJECT	LOCATION
Flashlight	1541
Fuses	1542
Painting	1543
Dog Biscuit	1544
Bone	1545
Brown Key	1546
Crowbar	1547

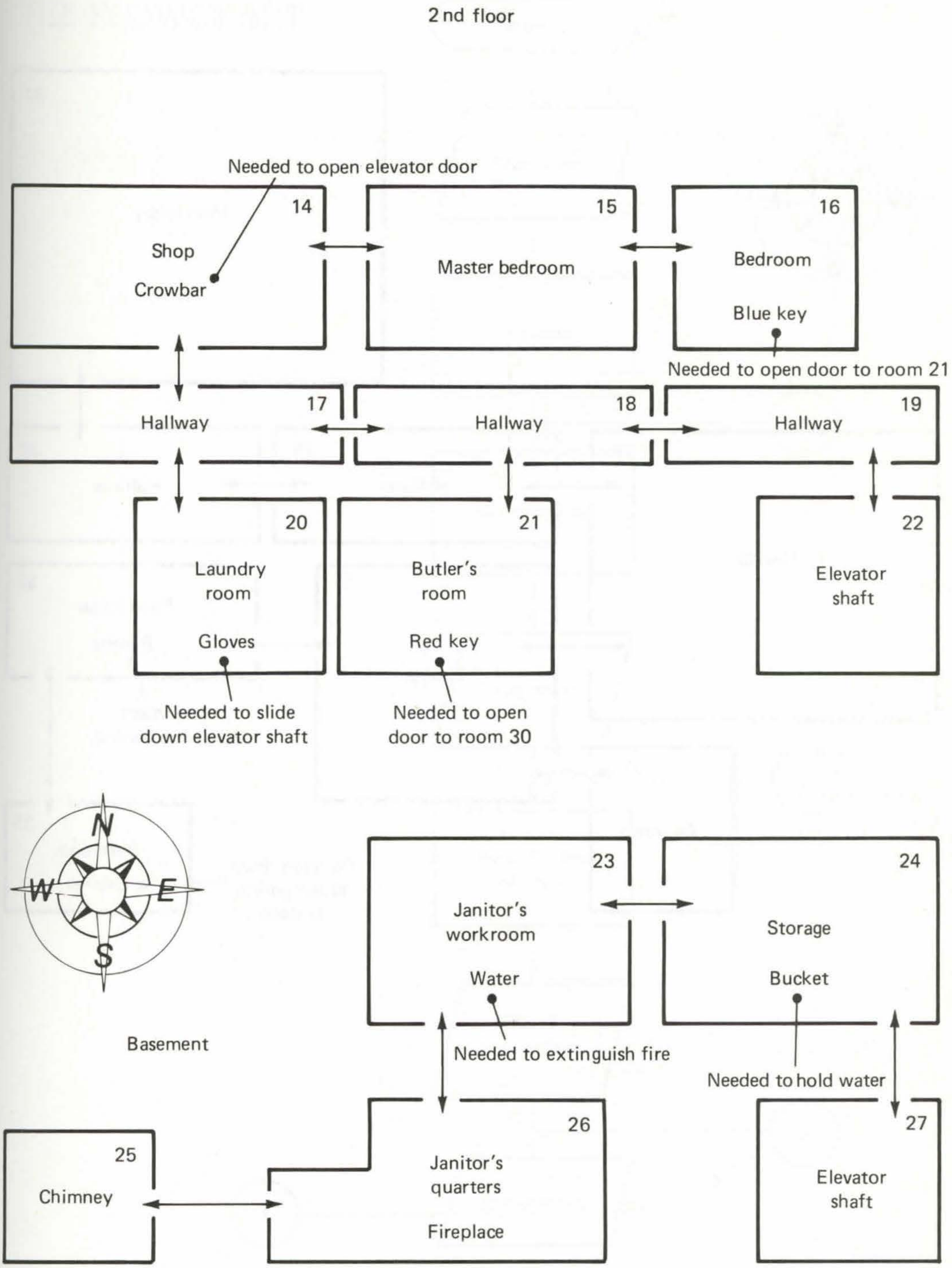
Blue Key	1548
Gloves	1549
Water	1550
Bucket	1551
Banana	1552
Drugs	1553
Red Key	1554
Fusebox	1555

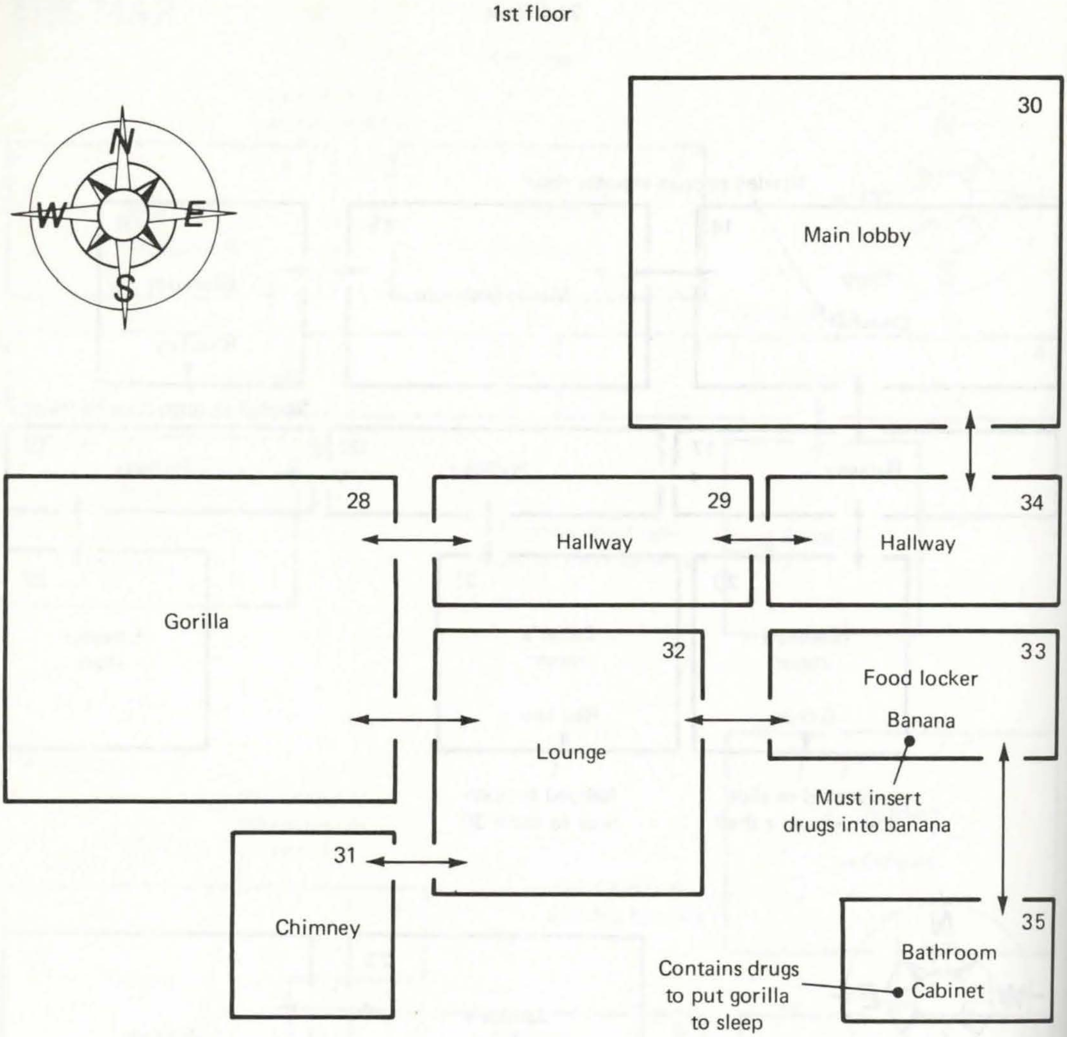
The following memory locations contain the status of the objects listed:

OBJECT	MEMORY LOCATION	
	CONTAINS 1	CONTAINS 0
Passageway	1556—open	closed
Dog	1557—present	gone
Flashlight	1558—on	off
Fusebox	1559—working	not working
4th Floor Elevator door	1560—open	closed
Cabinet	1561—open	closed
Bookcase	1562—open	closed
3rd Floor Cabinet	1563—open	closed
2nd Floor Elevator door	1564—open	closed
Laundry room door	1565—unlocked	locked
Door	1566—unlocked	locked
Fireplace	1567—fire	fire out
Gorilla	1568—sleeping	awake
Door to lobby	1569—unlocked	locked
Medicine Cabinet	1570—open	closed
Drugs	1571—used	not used

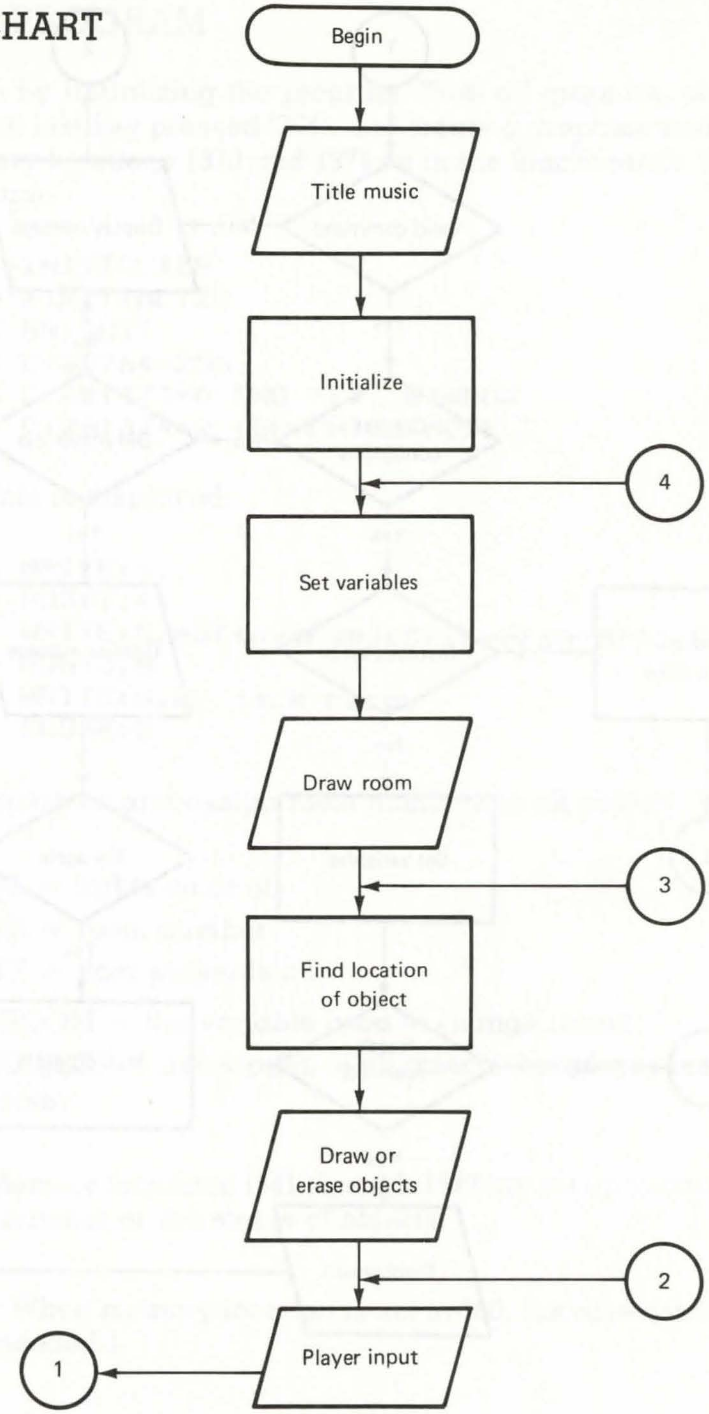
THE MAP

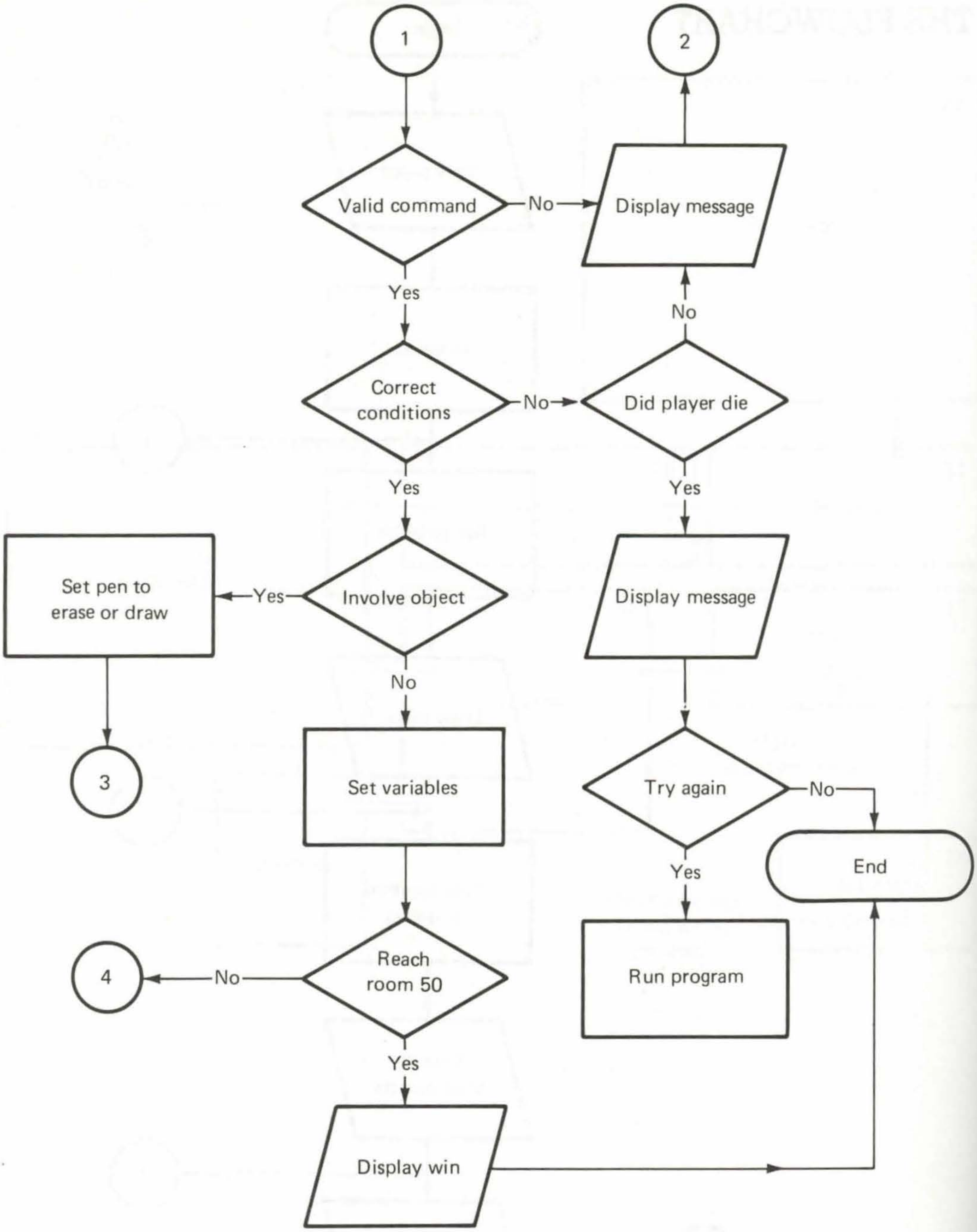

4th floor
Needed until
fusebox repaired
1
Bedroom
Flashlight
2
Workshop
Cabinet
N
W
E
S
Holds fuses
3
Hallway
4
Hallway
5
Office
Painting
Fusebox
6
Fusebox must have
fuses for elevator to run.
Elevator
3rd floor
Must move to
gain access to
room 11
7
Library
Bookcase
8
Kitchen
Cabinet
Contains bone
and dog biscuits
9
Hallway
Dog
Must give bone
10
Hallway
11
Maid's Room
Laundry chute
Need to descend
to 2nd floor
12
Bedroom
Gold key
Needed to open
door on 2nd floor
13
Elevator


2nd floor
Needed to open elevator door
14
Shop
Crowbar
15
Master bedroom
16
Bedroom
Blue key
Needed to open door to room 21
17
Hallway
18
Hallway
19
Hallway
20
Laundry room
Gloves
Needed to slide down elevator shaft
21
Butler's room
Red key
Needed to open door to room 30
22
Elevator shaft
N
W
E
S
Basement
23
Janitor's workroom
Water
Needed to extinguish fire
24
Storage
Bucket
Needed to hold water
25
Chimney
26
Janitor's quarters
Fireplace
27
Elevator shaft


1st floor
N
W
E
S
30
Main lobby
28
Gorilla
29
Hallway
34
Hallway
32
Lounge
33
Food locker
Banana
Must insert drugs into banana
31
Chimney
35
Bathroom
Cabinet
Contains drugs to put gorilla to sleep

THE FLOWCHART




1
2
Valid command
No
Display message
Yes
No
Correct conditions
No
Did player die
Yes
Yes
Set pen to erase or draw
Yes
Involve object
Display message
No
3
Set variables
Try again
No
End
Yes
Run program
4
No
Reach room 50
Yes
Display win

THE PROGRAM

Begin by initializing the program. Turn off graphics, clear register of last key pressed (764), and create a graphics mode using memory locations 1373 and 1374 as in the Blackbeard's Treasure program.

```
5 U:*INITIALIZE
5350 *INITIALIZE
5355 GR:QUIT
5360 C:@B764=255
5365 C:@B1373=0 [NO TEXT WINDOW
5370 C:@B1374=2 [GRAPHICS MODE 2
```

The title is displayed:

```
5375 WRITE:S,
5380 POS:7,4
5385 WRITE:S,eS[inverse]cA[inverse off]pE
5390 POS:3,8
5395 WRITE:S,by jack hardy
5400 CLOSE:S
```

Set Variables and assign room numbers to all objects:

#L = lights on or off

#R = room number

#F = floor player is on

$ROOM = the variable used to change rooms

$OBJECT = description of all objects that player sees in the room

Memory locations 1541 through 1572 are set up to contain the room number or the status of objects.

(*Note:* When memory location is set to 100, the object is in limbo until needed.)

```
5405 C:#L=1
5410 C:#R=1
5415 C:#F=4
5420 C:$ROOM=ROOM1
5425 C:@B1541=1
5430 C:@B1542=100
5435 C:@B1543=5
5440 C:@B1544=100
5445 C:@B1545=100
5450 C:@B1546=12
5455 C:@B1547=14
5460 C:@B1548=16
5465 C:@B1549=100
5470 C:@B1550=23
5475 C:@B1551=24
5480 C:@B1552=33
5485 C:@B1553=100
5490 C:@B1554=21
5495 C:@B1555=100
5500 C:@B1556=100
5505 C:#M=1557
5510 *LOOP1
5515 C:@B#M=0
5520 C:#M=#M+1
5525 J(#M<1572):*LOOP1
5530 C:$OBJECT= NOTHING
5535 C:$COMMA=,
5540 U:*DARK
5545 E:
```

The adventure begins in the dark. First, check to see if Fusebox is working (memory location 1559 set to 1), and if it is, return to the main program. If it isn't, continue by turning the inner screen black.

```
5550 *DARK
5555 E(@B1559=1):
5560 GR:CLEAR
5565 C:@B710=0
```

Display message and wait for input.

```
5570 *COMMAND
5575 T:[esc shift clear]IT'S VERY DARK
!!
5580 T:WHAT DO YOU WANT TO DO?
5585 A:
5590 M:FEE,LOO,GO ,MOV,LIG
5595 JM:*F,*L,*G,*G,*LI
5600 T:[esc shift clear]YOU CAN'T DO T
HAT NOW!!
5605 *WAIT2
5610 PA:140
5615 J:*COMMAND
```

Player tries to LOOK around in the dark.

```
5650 *L
5655 T:[esc shift clear]IT IS TOO DARK
 TO SEE ANYTHING!!
5660 J:*WAIT2
```

Player wants to move around in the dark.

```
5665 *G
5670 T:[esc shift clear]IT'S DANGEROUS
 TO MOVE IN THE DARK!!
5675 PA:240
5680 J:*DEAD4
5790 *DEAD4
5795 T:[esc shift clear]YOU FALL AND B
REAK YOUR NECK!!
5800 J:*DEATHMARCH
```

If player moves, the program plays music and checks to see if player wants to play again.

```
5820 *DEATHMARCH
5825 C:#P=2
5830 SO:5
5835 PA:32
5840 U:*PAUSE
5845 SO:5
5850 PA:16
```

```
5855 U:*PAUSE
5860 SO:5
5865 PA:4
5870 U:*PAUSE
5875 SO:5
5880 PA:32
5885 U:*PAUSE
5890 SO:8
5895 PA:16
5900 U:*PAUSE
5905 SO:7
5910 PA:8
5915 U:*PAUSE
5920 SO:7
5925 PA:8
5930 U:*PAUSE
5935 SO:5
5940 PA:8
5945 U:*PAUSE
5950 SO:5
5955 PA:8
5960 U:*PAUSE
5965 SO:4
5970 PA:8
5975 U:*PAUSE
5980 SO:5
5985 PA:32
5990 SO:0
5995 *PLAYAGAIN
6000 T:[esc shift clear/6 spaces]WANT
TO TRY AGAIN? (Y/N)
```

Set memory location 764 for no key pressed and then check for which key pressed (43 = Y, 35 = N). (For more internal values stored in location 764 when keys are pressed, see Appendix B.)

```
6005 C:@B764=255
6010 *YNKEY
6015 J(@B764=43):*INITIALIZE
6020 J(@B764=35):*STOP
6025 J:*YNKEY
```

Pause routine for music.

```
6030 *PAUSE
6035 SO:0
6040 PA:#P
6045 E:
```

If player wants to quit:

```
6050 *STOP
6055 GR:QUIT
6060 C:@764=255
6065 T:[esc shift clear]THANKS FOR PLA
YING!!!
6070 E:
```

If the player decides to FEEL around, the program first checks to make sure flashlight is in the room (memory location 1541 holds the room number where flashlight is located).

```
5620 *F
5625 T(@B1541<>#R):[esc shift clear]YO
U DON'T FEEL ANYTHING!!
5630 J(@B1541<>#R):*WAIT2
```

When player finds the flashlight, assign a value of 0 to memory location 1541, which tells the computer the object is now in the possession of the player.

```
5635 T:[esc shift clear]YOU FIND A FLA
SHLIGHT!!
5640 C:@B1541=0
5645 J:*WAIT2
```

When the player wishes to turn the flashlight on, the program makes sure it is in his or her possession.

```
5685 *LI
5690 T(@B1541<>0):[esc shift clear]YOU
 DON'T HAVE THE FLASHLIGHT!!
5695 J(@B1541<>0):*WAIT2
```

When memory location 1541 equals 0 (i.e., player has flashlight), the flashlight can be turned on. Set the colors and jump to the first room (#1).

```
5700 C:@B1558=1
5705 C:@B710=148
5710 C:@B709=42
5715 E:
```

```
(*)RUN
```

Assign values to all of the necessary variables for Room #1.

```
10 *ROOM1
15 U:*BR
20 C:$AREA=BEDROOM
25 C:#E=2
30 C:#S=3
35 J:*P
```

The first graphics routine draws the room:

```
3875 *BR
3880 GR:CLEAR;PEN YELLOW
3885 GR:GOTO74,44;TURNTO180;DRAW23;TUR
N45;DRAW71;TURN135;FILL23;TURN45;FILL7
0
3890 GR:GOTO-50,-6;DRAW37
3895 GR:GOTO-76,-30;TURNTO0;FILL23;TUR
N45;FILL72
3900 GR:PEN ERASE;GOTO50,21;DRAW40
3905 GR:GOTO-75,-6;TURN45;DRAW26
3910 GR:GOTO-23,45;TURN90;DRAW24
3915 GR:GOTO23,-6;DRAW24
3920 E:
```

Now print information.

Note: If flashlight is not on, jump to dark routine:

```
1310 *P
1315 U(@B1558=0):*DARK
```

Description of room:

```
1320 T:[esc shift clear]LOCATION: $ARE
A
```

The floor number the player is on:

```
1325 T(#F=4):[4 esc tabs/esc ctrl -/es
c ctrl +]4TH FLOOR
1330 T(#F=3):[4 esc tabs/esc ctrl -/es
c ctrl +]3RD FLOOR
1335 T(#F=2):[4 esc tabs/esc ctrl -/es
c ctrl +]2ND FLOOR
1340 T(#F=1):[4 esc tabs/esc ctrl -/es
c ctrl +]1ST FLOOR
1345 T(#F=0):[4 esc tabs/esc ctrl -]BA
SEMENT
```

Draw arrows for the directions in which player can travel:

```
1350 U(#N<>0):*NORTHDOOR
1355 U(#S<>0):*SOUTHDOOR
1360 U(#E<>0):*EASTDOOR
1365 U(#W<>0):*WESTDOOR
1370 U(#D<>0):*DOWNARROW
1375 U(#U<>0):*UPARROW
```

Jump to module that checks placement of the object on the screen:

```
1380 U:*PL
```

Set other variables and jump to routine to print items seen in the room:

```
1385 C:#L=1
1390 C:#M=1540
1395 C:#I=0
1400 C:$ITEMS=$OBJECT
1405 A:=$ITEMS
1410 M: NOTHING
1415 CN:#I=#I+1
1420 U:*ITEMS
```

Print all items in the room:

```
1425 T:VISIBLE ITEMS:\
1430 T:$ITEMS
```

The graphics routines to draw the doors and arrows.

Note: The program is matching room numbers in the routines below to see if a door should be drawn. Sometimes the player can go in a direction where there is no door (e.g., hallway).

```
4110 *SOUTHDOOR
4115 A:=#R
4120 M:7 ,
4125 JY:*SOUTHARROW
4130 GR:PEN ERASE;TURNTO0;GOTO-40,-30;
2(DRAW21;TURN90;DRAW15;TURN90);GOTO-28
,-20;4(DRAW1;TURN90)
4135 J:*SOUTHARROW
4140 *SOUTHARROW
4145 GR:PEN RED;TURNTO90;GOTO61,-30;DR
AW4;TURN-90;DRAW2;TURN-90;DRAW4;TURN90
;DRAW2;TURN90;DRAW4
4150 GR:TURN90;GOTO63,-20;DRAW4;TURN13
5;DRAW3;TURN90;GOTO63,-24;DRAW3
4155 E:
4160 *NORTHDOOR
4165 A:=$ROOM
4170 M:M6 ,13 ,22 ,27 ,
4175 CY:#X=10
4180 CY:#Y=20
4185 JY:*ELEDOOR
4190 GR:PEN ERASE;TURNTO0;GOTO8,20;2(D
RAW21;TURN90;DRAW15;TURN90);GOTO20,30;
4(DRAW1;TURN90)
4195 J:*NORTHARROW
4200 *NORTHARROW
4205 GR:PEN RED;GOTO63,-18;DRAW4;TURN1
35;DRAW3;TURN90;GOTO63,-14;DRAW3
4210 GR:TURNTO0;GOTO61,-12;DRAW4;TURN1
35;DRAW5;TURN-135;DRAW4
4215 E:
4220 *EASTDOOR
4225 A:=$ROOM
4230 M:M3 ,M9 ,17 ,18 ,29 ,
4235 JY:*EASTARROW
4240 GR:PEN ERASE
4245 GR:TURNTO0;GOTO48,-8;2(DRAW21;TUR
N45;DRAW13;TURN135);GOTO54,8;2(DRAW1;T
URN45;DRAW1;TURN135)
4250 J:*EASTARROW
4255 *EASTARROW
4260 GR:PEN RED;TURNTO90;GOTO64,-19;DR
AW6;TURN135;DRAW3;TURN90;GOTO70,-19;DR
AW3
4265 GR:TURNTO270;GOTO75,-21;DRAW3;TUR
N90;DRAW4;TURN90;DRAW3;GOTO73,-19;DRAW1
4270 E:
4275 *WESTDOOR
4280 A:=$ROOM
4285 M:M4 ,10 ,18 ,19 ,34 ,
4290 JY:*WESTARROW
4295 M:26 ,32 ,
4300 EY:
4305 GR:PEN ERASE
4310 GR:TURNTO0;GOTO-48,-5;2(DRAW21;TU
RN45;DRAW13;TURN135);GOTO-42,10;(DRAW1
;TURN45;DRAW1;TURN135)
4315 J:*WESTARROW
4320 *WESTARROW
4325 GR:PEN RED;TURNTO270;GOTO62,-19;D
RAW6;TURN135;DRAW3;TURN90;GOTO56,-19;D
RAW3
4330 GR:TURNTO180;GOTO54,-17;DRAW4;TUR
N135;DRAW3;TURN-90;DRAW3;TURN135;DRAW4
4335 E:
4405 *DOWNARROW
4410 A:=#R
4415 M:11 ,
4420 EY:
4425 GR:PEN RED;TURNTO180;GOTO-72,34;D
RAW4;TURN135;DRAW3;TURN90;GOTO-72,30;D
RAW3
4430 GR:GOTO-74,28;TURN45;11(DRAW1;TUR
N18);TURNTO0;DRAW6;PEN ERASE
4435 E:
4440 *UPARROW
4445 GR:PEN RED;TURNTO0;GOTO-72,36;DRA
W4;TURN135;DRAW3;TURN90;GOTO-72,40;DRA
W3
4450 GR:TURNTO180;GOTO-70,46;3(DRAW4;T
URN90);PEN ERASE
4455 E:
```

The routine below is used to determine where objects should be placed on screen.

```
4460 *PL
4465 C:#X=-24
4470 C:#Y=16
4475 A:=$ROOM
4480 M:M3 ,M4 ,M6 ,10 ,13 ,17 ,18 ,19
,22 ,25 ,27 ,31 ,34 ,35 ,
4485 CY:#X=-54
4490 CY:#Y=-12
4495 M:M9 ,29 ,
4500 CY:#X=10
4505 CY:#Y=24
4510 E:
```

The program below checks for items in each location.

Note: #M is a memory location which holds the current status of the item. This was set to 1540, in line 1390; it is then incremented by 1 in line 1490 as the first location to check. As it loops, it checks the value of the memory location to find which is equal to the room the player is now occupying. On each loop #M is incremented to the next memory location. If it finds something in the room, it jumps out of the routine to find out what item has been found, and then jumps to a graphics routine to draw a picture of the item.

```
1485 *ITEMS
1490 C:#M=#M+1
1495 U(#R=@B#M):*VISIBLE
1500 J(#M<1557):*ITEMS
1505 E:
```

Next add the name of the object to the variable $ITEMS.

```
1585 *VISIBLE
1590 C(#I>0):$ITEMS=$ITEMS$COMMA
1595 C:#I=#I+1
```

```
1600 C:#V=#M-1540
1605 C(#V=1):$ITEM= FLASHLIGHT
1610 C(#V=2):$ITEM= FUSES
1615 C(#V=3):$ITEM= PAINTING
1620 C(#V=4):$ITEM= DOG BISCUITS
1625 C(#V=5):$ITEM= BONE
1630 C(#V=6):$ITEM= BROWN KEY
1635 C(#V=7):$ITEM= CROWBAR
1640 C(#V=8):$ITEM= BLUE KEY
1645 C(#V=9):$ITEM= GLOVES
1650 C(#V=10):$ITEM= WATER
1655 C(#V=11):$ITEM= BUCKET
1660 C(#V=12):$ITEM= BANANA
1665 C(#V=13):$ITEM= DRUGS
1670 C(#V=14):$ITEM= RED KEY
1675 C(#V=15):$ITEM= FUSEBOX
1680 C(#V=16):$ITEM= PASSAGEWAY
1685 C(#I=1):$ITEMS=$ITEM
1690 C(#I>1):$ITEMS=$ITEMS$ITEM
1695 U:*DRAWPICTURE
1700 E:
```

The program then jumps to routines to draw a picture of the object.

```
1705 *DRAWPICTURE
1710 C:#V=#M-1540
1715 U(#V=1):*FLASHPIC
1720 U(#V=2):*FUSEPIC
1725 U(#V=3):*PICPIC
1730 U(#V=4):*BISPIC
1735 U(#V=5):*BONEPIC
1740 U(#V=6):*KEYPIC
1745 U(#V=7):*CROPIC
1750 U(#V=8):*KEYPIC
1755 U(#V=9):*GLOVEPIC
1760 U(#V=11):*BUCKETPIC
1765 U(#V=12):*BANANAPIC
1770 U(#V=13):*DRUGPIC
1775 U(#V=14):*KEYPIC
1780 E:
```

Now add the graphic routines to draw all of the items in the adventure.

```
4515 *FLASHPIC
4520 GR(#L=1):PEN BLUE
4525 GR(#L=0):PEN ERASE
4530 GR:TURNTO0;GOTO#X,#Y-1
4535 GR:2(DRAW1;TURN90;DRAW3;TURN90)
4540 GR(#L=1):PEN YELLOW
4545 GR(#L=0):PEN ERASE
4550 GR:GOTO#X+3,#Y;TURN45;DRAW3;TURN1
35;DRAW4;TURN135;DRAW3
4555 E:
4560 *FUSEPIC
4565 GR(#L=1):PEN RED
4570 GR(#L=0):PEN ERASE
4575 GR:TURNTO315;GOTO#X+9,#Y-1;3(DRAW
1;TURN135)
4580 E:
4585 *PICPIC
4590 GR:PEN RED;TURNTO 0
4595 GR(@B1555=#R):PEN BLUE
4600 GR:GOTO-48,-2;2(DRAW15;TURN45;DRA
W15;TURN135);PEN BLUE
4605 GR(@B1555=#R):PEN YELLOW
4610 GR:GOTO -47,4
4615 C:#A=0
4620 *SPIRAL
4625 GR:2(DRAW#A;TURN45;DRAW#A*2;TURN1
35)
4630 C:#A=#A+2
4635 J(#A<8):*SPIRAL
4640 E:
4645 *BISPIC
4650 GR(#L=1):PEN RED
4655 GR(#L=0):PEN ERASE
4660 GR:TURNTO45;GOTO#X+13,#Y-1;4(DRAW
1;TURN90)
4665 E:
4670 *KEYPIC
4675 A:=$ITEM
4680 M:BRO,RED,BLUE
4685 JM:*BROWNPIC,*REDPIC,*BLUEPIC
4690 *BROWNPIC
```

```
4695 GR:PEN YELLOW
4700 C:#K=#X+40
4705 J:*KEYPIC2
4710 *REDPIC
4715 GR:PEN RED
4720 C:#K=#X+24
4725 J:*KEYPIC2
4730 *BLUEPIC
4735 GR:PEN BLUE
4740 C:#K=#X+32
4745 J:*KEYPIC2
4750 *KEYPIC2
4755 GR(#L=0):PEN ERASE
4760 GR:TURNTO0;GOTO#K,#Y-1;4(DRAW2;TU
RN90);TURNTO90;GOTO#K+3,#Y;DRAW3;TURN9
0;DRAW1;GOTO#K+4,#Y;DRAW1
4765 E:
4770 *BONEPIC
4775 GR:PEN BLUE
4780 GR(#L=0):PEN ERASE
4785 GR:TURNTO180;GOTO#X+17,#Y;2(DRAW2
;TURN90;DRAW1;TURN90);GO1;TURN-90;DRAW
4
4790 GR:2(TURN90;DRAW1);TURN90;DRAW2;2
(TURN90;DRAW1)
4795 E:
4800 *CROPIC
4805 GR:PEN YELLOW
4810 GR(#L=0):PEN ERASE
4815 GR:TURNTO325;GOTO#X,#Y-8;DRAW2;TU
RNTO90;DRAW5;TURN-135;DRAW2
4820 E:
4825 *GLOVEPIC
4830 GR:PEN RED
4835 GR(#L=0):PEN ERASE
4840 GR:TURNTO0;GOTO#X+10,#Y-8;2(DRAW2
;TURN90;DRAW4;TURN90);TURN-45;DRAW3
4845 GR:GOTO#X+10,#Y-6;TURNTO0;DRAW2;G
OTO#X+12,#Y-6;DRAW2;GOTO#X+14,#Y-6;DRA
W1
4850 E:
4855 *BUCKETPIC
4860 GR:PEN YELLOW;TURNTO 0
4865 GR(#L=0):PEN ERASE
```

```
4870 GR:GOTO#X+17,#Y-8;DRAW3;6(TURN30;
DRAW1);DRAW3;TURN90;DRAW4;PENBLUE
4875 GR(#L=0):PEN ERASE
4880 GR:GOTO#X+18,#Y-6;TURNTO90;DRAW2
4885 E:
4890 *DRUGPIC
4895 GR:PEN RED
4900 GR(#L=0):PEN ERASE
4905 GR:GOTO#X+24,#Y-6;GOTO#X+26,#Y-7;
GOTO#X+28,#Y-8
4910 GR:PEN BLUE
4915 GR(#L=0):PEN ERASE
4920 GR:GOTO#X+24,#Y-8;GOTO#X+30,#Y-6;
GOTO#X+27,#Y-7
4925 E:
4930 *BANANAPIC
4935 GR:PEN YELLOW
4940 GR(#L=0):PEN ERASE
4945 GR:GOTO#X+32,#Y-8;TURNTO90;DRAW3;
TURN-30;DRAW2
4950 E:
```

```
(*)RUN
```

Now add the input routine. Erase last letter pressed and prepare for new input.

```
1435 C:@B764=255
1440 T:[inverse]COMMAND =>[inverse off
] \
1445 A:
```

Next check for direction.

Note: Here the match command is set up to check only the first three letters of a word. Therefore, the player could input GO NORTH, GO NOR, NORTH, NOR, and even NORSE and the program would interpret each as the player's command to go north from the present location.

```
1450 M: NOR, SOU, EAS, WES, UP , DOW,
1455 JM:*NORTH,*SOUTH,*EAST,*WEST,*UP,
*DOWN
```

```
1470 T:[esc shift clear]I DON'T UNDERS
TAND THAT!!
1475 PA:90
1480 J:*P
```

Following are the routines for directions.

Note: If direction variable = −1, the door is not yet unlocked, nor has it been forced open. If the direction variable = 0 there is nothing in that direction.

```
1785 *NORTH
1790 J(#N=0):*CANTGO
1795 J(#N=-1):*CHECKDOOR
1800 C:#R=#N
1805 J:*CR
1810 *SOUTH
1815 J(#S=0):*CANTGO
1820 J(#S=-1):*CHECKDOOR
1825 C:#R=#S
1830 J:*CR
1835 *EAST
1840 J(#E=0):*CANTGO
1845 J(#E=-1):*CHECKDOOR
1850 C:#R=#E
1855 J:*CR
1860 *WEST
1865 J(#W=0):*CANTGO
1870 J(#W=-1):*CHECKDOOR
1875 C:#R=#W
1880 J:*CR
1885 *UP
```

Note: Change the status of the #F variable in the up and down routines to show that the player has either gone up or down a floor.

```
1890 J(#U=0):*CANTGO
1895 C:#R=#U
1900 C:#F=#F+1
1905 C(#R=27):#F=#F+1
1910 J:*CR
1915 *DOWN
```

```
1920 J(#D=0):*CANTGO
1925 J(#R=11):*HOW
1930 J(#R=22):*CKG
1935 C:#R=#D
1940 C:#F=#F-1
1945 J:*CR
1950 *HOW
1955 T:[esc shift clear]HOW????
1960 J:*WAIT
1965 *CANTGO
1970 T:[esc shift clear]I CAN'T GO IN
THAT DIRECTION!!
1975 J:*WAIT
```

Note: The *CKG routine checks to see if the player is wearing the gloves.

```
1980 *CKG
1985 J(@B1549<>50):*DEAD1
1990 C:#R=#D
1995 C:#F=#F-2
2000 J:*CR
```

If the direction status equals −1, check room numbers and display the proper message.

```
2005 *CHECKDOOR
2010 A:=$ROOM
2015 M:M4 ,M9 ,18 ,19 ,20 ,28 ,26 ,32
,34 ,
2020 JM:*STUCK,*BLOCKED,*LOCKED,*STUCK
,*LOCKED,*BLOCKED2,*HOW,*HOW,*LOCKED
2025 *STUCK
2030 T:[esc shift clear]THE DOOR WILL
NOT OPEN!!
2035 J:*WAIT
2040 *LOCKED
2045 T:[esc shift clear]THE DOOR IS LO
CKED!!
2050 J:*WAIT
2055 *BLOCKED
2060 T:[esc shift clear]THE DOG BLOCKS
 YOUR WAY!!!
```

```
2065 J:*WAIT
2070 *BLOCKED2
2075 T:[esc shift clear]THE GORILLA BL
OCKS YOUR WAY!!
2080 J:*WAIT
```

The WAIT and OKAY routines:

```
2575 *WAIT
2580 PA:240
2585 J:*P
2590 *OK
2595 T:[esc shift clear]OKAY!!
2600 PA:60
2605 J:*P
```

The Change Room routine:

Note: In the routine above, the value of variable #R is set to equal that of the direction variable. Now the program sets $ROOMNUM to equal #R. It also sets $OBJECT to nothing and clears the direction variables.

```
2085 *CR
2090 C:$ROOMNUM=#R
2095 C:$OBJECT= NOTHING
2100 C:#N=0
2105 C:#S=0
2110 C:#E=0
2115 C:#W=0
2120 C:#U=0
2125 C:#D=0
```

Change the room number to the new room and input it into the running program to match the correct room.

```
2130 C:$ROOM=ROOM$ROOMNUM
2135 A:=$ROOM
2140 M:M1 ,M2 ,M3 ,M4 ,M5 ,M6 ,M7 ,M8
,M9 ,M10 ,11 ,12 ,13 ,14 ,
2145 JM:*ROOM1,*ROOM2,*ROOM3,*ROOM4,*R
OOM5,*ROOM6,*ROOM7,*ROOM8,*ROOM9,*ROOM
10,*ROOM11,*ROOM12,*ROOM13,*ROOM14
```

```
2150 M:15 ,16 ,17 ,18 ,19 ,20 ,21 ,22
,23 ,24 ,25 ,26 ,
2155 JM:*ROOM15,*ROOM16,*ROOM17,*ROOM1
8,*ROOM19,*ROOM20,*ROOM21,*ROOM22,*ROO
M23,*ROOM24,*ROOM25,*ROOM26
2160 M:27 ,28 ,29 ,30 ,31 ,32 ,33 ,34
,35 ,50 ,
2165 JM:*ROOM27,*ROOM28,*ROOM29,*ROOM3
0,*ROOM31,*ROOM32,*ROOM33,*ROOM34,*ROO
M35,*WIN
2170 E:
```

Next, add more rooms.

Note: In the Blackbeard's Treasure program, the variables contained information about the current status of objects. In this program, the memory location assigned to certain objects is used to inform the program of the current status of those objects. The conditional command checks the number stored in these memory locations to change descriptions of the game locations or to relay messages.

```
40 *ROOM2
45 U:*BR
50 U:*CP
55 C:$AREA=WORKSHOP
60 C:$OBJECT= CABINET
65 C(@B1561=1):$OBJECT= OPEN CABINET
70 C:#W=1
75 J:*P
80 *ROOM3
85 U:*HW
90 U:*CW
95 U:*HD
100 C:$AREA=HALLWAY
105 C:#N=1
110 C:#S=5
115 C:#E=4
120 J:*P
125 *ROOM4
130 U:*HW
135 U:*CE
140 U:*HD
```

```
145 C:$AREA=HALLWAY
150 C:#S=-1
155 C(@B1559=1):#S=6
160 C:#W=3
165 J:*P
170 *ROOM5
175 U:*BR
180 U:*PICPIC
185 C:$AREA=OFFICE
190 C:#N=3
195 J:*P
200 *ROOM6
205 U:*SR
210 C:$AREA=ELEVATOR
215 C:#N=4
220 C:#D=13
225 J:*P
230 *ROOM7
235 U:*BR
240 U:*BOOKCPIC
245 C:$AREA=LIBRARY
250 C:#E=9
255 C:$OBJECT= BOOKCASE
260 C(@B1562=1):#S=11
265 C(@B1562=1):$OBJECT= OPEN BOOKCASE
270 J:*P
275 *ROOM8
280 U:*BR
285 U:*CP
290 C:$AREA=KITCHEN
295 C:#S=10
300 C:$OBJECT= CABINET
305 C(@B1563=1):$OBJECT= OPEN CABINET
310 J:*P
315 *ROOM9
320 U:*ELBOW
325 U:*DOGPIC
330 C:$AREA=HALLWAY
335 C:$OBJECT= DOG
340 C:#S=12
345 C:#W=-1
350 C(@B1557=1):#W=7
355 C(@B1557=1):$OBJECT= NOTHING
360 C:#E=10
```

```
365 J:*P
370 *ROOM10
375 U:*HW
380 U:*CE
385 C:$AREA=HALLWAY
390 U:*HD
395 C:#S=13
400 C:#W=9
405 C:#N=8
410 J:*P
415 *ROOM11
420 U:*BR
425 C:$AREA=MAID'S ROOM
430 C:#N=7
435 C:#D=20
440 C:$OBJECT= LAUNDRY CHUTE
445 J:*P
450 *ROOM12
455 U:*BR
460 C:$AREA=BEDROOM
465 C:#N=9
470 J:*P
475 *ROOM13
480 U:*SR
485 C:$AREA=ELEVATOR
490 C:#N=10
495 C:#U=6
500 J:*P
505 *ROOM14
510 U:*BR
515 C:$AREA=SHOP
520 C:#S=17
525 C:#E=15
530 J:*P
535 *ROOM15
540 U:*BR
545 C:$AREA=MASTER BEDROOM
550 C:#W=14
555 C:#E=16
560 J:*P
565 *ROOM16
570 U:*BR
575 C:$AREA=BEDROOM
580 C:#W=15
```

```
585 J:*P
590 *ROOM17
595 U:*HW
600 U:*CW
605 U:*HD
610 C:$AREA=HALLWAY
615 C:#N=14
620 C:#S=20
625 C:#E=18
630 J:*P
635 *ROOM18
640 U:*HW
645 U:*HD
650 C:$AREA=HALLWAY
655 C:#E=19
660 C:#W=17
665 C:#S=-1
670 C(@B1566=1):#S=21
675 J:*P
680 *ROOM19
685 U:*HW
690 U:*HD
695 U:*CE
700 C:$AREA=HALLWAY
705 C:#W=18
710 C:#S=-1
715 C(@B1564=1):#S=22
720 J:*P
725 *ROOM20
730 U:*BR
735 C:$AREA=LAUNDRY ROOM
740 C:#N=-1
745 C:#U=11
750 C(@B1565=1):#N=17
755 C:$OBJECT= DIRTY CLOTHES
760 J:*P
765 *ROOM21
770 U:*BR
775 C:$AREA=BUTLER'S ROOM
780 C:#N=18
785 J:*P
790 *ROOM22
795 U:*SR
800 C:$AREA=ELEVATOR SHAFT
```

```
805 C:#D=27
810 C:#N=19
815 J:*P
820 *ROOM23
825 U:*BR
830 U:*SINKPIC
835 C:$AREA=JANITOR'S WORKROOM
840 C:$OBJECT= SINK
845 C:#E=24
850 C:#S=26
855 J:*P
860 *ROOM24
865 U:*BR
870 C:$AREA=STORAGE
875 C:#S=27
880 C:#W=23
885 J:*P
890 *ROOM25
895 U:*SR
900 U:*FIREPIC
905 C:$AREA=CHIMNEY
910 C:#U=31
915 C:#E=26
920 J:*P
925 *ROOM26
930 U:*BR
935 U:*FIREPIC
940 C:$AREA=JANITOR'S QUARTERS
945 C:$OBJECT= FIRE IN FIREPLACE
950 C:#N=23
955 C:#W=-1
960 C(@B1567=1):#W=25
965 C(@B1567=1):$OBJECT= FIREPLACE
970 J:*P
975 *ROOM27
980 U:*SR
985 C:$AREA=ELEVATOR SHAFT
990 C:#N=24
995 C:#U=22
1000 J:*P
1005 *ROOM28
1010 U:*BR
1015 U:*GORILLAPIC
1020 C:$AREA=ZOO
```

```
1025 C:#S=32
1030 C:$OBJECT= GORILLA
1035 C(@B1568=1):$OBJECT= SLEEPING GORILLA
1040 C:#E=-1
1045 C(@B1568=1):#E=29
1050 J:*P
1055 *ROOM29
1060 U:*ELBOW
1065 C:$AREA=HALLWAY
1070 C:#W=28
1075 C:#E=34
1080 J:*P
1085 *ROOM30
1090 U:*BR
1095 C:$AREA=MAIN LOBBY
1100 C:#N=50
1105 C:#S=34
1110 J:*P
1115 *ROOM31
1120 U:*SR
1125 U:*FIREPIC
1130 C:$AREA=CHIMNEY
1135 C:#E=32
1140 C:#D=25
1145 J:*P
1150 *ROOM32
1155 U:*BR
1160 U:*FIREPIC
1165 C:$AREA=LOUNGE
1170 C:$OBJECT= FIREPLACE
1175 C:#N=28
1180 C:#E=33
1185 C:#W=31
1190 J:*P
1195 *ROOM33
1200 U:*BR
1205 C:$AREA=FOOD LOCKER
1210 C:#S=35
1215 C:#W=32
1220 C(@B1552<>0):@B1552=33
1225 J:*P
1230 *ROOM34
1235 U:*HW
1240 U:*CE
```

```
1245 C:$AREA=HALLWAY
1250 C:#W=29
1255 C:#N=-1
1260 C(@B1569=1):#N=30
1265 J:*P
1270 *ROOM35
1275 U:*SR
1280 U:*MCP
1285 C:$AREA=BATHROOM
1290 C:#N=33
1295 C:$OBJECT= MEDICINE CABINET
1300 C(@B1570=1):$OBJECT= OPEN MEDICIN
E CABINET
1305 J:*P
```

Now the graphic routines are needed to draw the small room, halls, elevators, etc.

```
3925 *SR
3930 GR:CLEAR;PEN YELLOW
3935 GR:GOTO56,44;TURNTO180;DRAW23;TUR
N45;DRAW41;TURN135;FILL23;TURN45;FILL4
0
3940 GR:GOTO-6,16;DRAW6
3945 GR:GOTO-32,-9;TURNTO0;FILL24;TURN
45;FILL41
3950 GR:PEN ERASE;GOTO32,21;DRAW40
3955 GR:GOTO-31,16;TURN45;DRAW26
3960 GR:GOTO26,16;TURN90;DRAW24
3965 GR:GOTO-1,45;DRAW24;GOTO-32,-9
3970 E:
3975 *HW
3980 GR:CLEAR;PEN YELLOW
3985 GR:GOTO74,44;TURNTO180;DRAW23;TUR
N90;DRAW121;TURN90;FILL23
3990 GR:GOTO46,16;TURN180;DRAW23;TURN9
0;DRAW121;TURN90;FILL23
3995 GR:TURN45;GOTO-50,17;DRAW4
4000 GR:GOTO46,-7;DRAW38
4005 E:
4010 *CE
4015 GR:PEN YELLOW;TURNTO0;GOTO46,-7;F
ILL24;TURN45;FILL4;PEN ERASE
```

```
4020 GR:TURN-45;GO1;TURN45;DRAW38
4025 GR:TURN-45;GOTO45,-7;DRAW24
4030 E:
4035 *CW
4040 GR:PEN YELLOW;TURNTO45;GOTO-75,17
;FILL38
4045 GR:PEN ERASE;GO3;TURN135;DRAW29;T
URN90;DRAW30
4050 E:
4055 *ELBOW
4060 GR:CLEAR;PEN YELLOW
4065 GR:TURNTO180;GOTO44,10;DRAW23;TUR
N45;DRAW23;TURN135;DRAW16;FILL7;TURN18
0;DRAW23;TURN90
4070 GR:DRAW104;TURN90;FILL23
4075 GR:PEN ERASE;TURN90;DRAW46;TURN-9
0;PEN YELLOW;FILL16
4080 GR:TURN90;GO23;TURN-45;GO1;DRAW22
;TURN-45;DRAW23;TURN-135;FILL53
4085 GR:GOTO-22,23;DRAW44;GOTO-22,48;F
ILL76
4090 GR:TURN-135;PEN ERASE;DRAW104;TUR
N-45;DRAW23
4095 GR:TURN-135;DRAW74;TURN135;DRAW52
4100 GR:TURN-45;GOTO-21,24;DRAW23;GOTO
28,-29;DRAW23
4105 E:
```

These routines draw hall and elevator door pictures:

```
4340 *HD
4345 A:=$ROOM
4350 M:ROOM4 ,ROOM10 ,ROOM19 ,
4355 CY:#X=-25
4360 CY:#Y=-8
4365 JY:*ELEDOOR
4370 GR:PEN ERASE;TURNTO0;GOTO-20,-8;2
(DRAW21;TURN90;DRAW15;TURN90);GOTO-8,2
;4(DRAW1;TURN90)
4375 J:*SOUTHARROW
4380 *ELEDOOR
4385 GR:PEN ERASE
4390 GR:TURNTO0;GOTO#X,#Y;3(DRAW20;TUR
N90);DRAW10;TURN90;DRAW20;GOTO#X+22,#Y
+10;GOTO#X+22,#Y+12
4395 J(#X=-25):*SOUTHARROW
4400 J:*NORTHARROW
```

These routines draw the room contents pictures:

```
4955 *CP
4960 GR:PEN RED;TURNTO0;GOTO10,25
4965 GR:4(DRAW 15;TURN 90)
4970 U:*CHECKCABPIC
4975 GR(#O=1):PEN YELLOW
4980 GR:GOTO17,26;DRAW14;GOTO15,32;GOT
O19,32
4985 GR(#O=1):PEN RED;GOTO10,25;TURNTO
225;DRAW5;TURN135;DRAW15;TURN45;DRAW5
4990 GR(#O=1):TURN45;DRAW15;TURN45;DRA
W5;TURN45;DRAW15;TURN135;DRAW5
4995 E:
5000 *CHECKCABPIC
5005 A:=#R
5010 M:2 ,8 ,
5015 JM:*CAB2PIC,*CAB8PIC
5020 E:
5025 *CAB2PIC
5030 C:#O=0
5035 C(@B1561=1):#O=1
5040 E:
5045 *CAB8PIC
5050 C:#O=0
5055 C(@B1563=1):#O=1
5060 E:
5065 *BOOKCPIC
5070 GR:PEN BLUE;GOTO-30,-6;TURNTO0
5075 GR(@B1562=0):2(DRAW3;TURN90;DRAW3
2;TURN90);FILL3
5080 GR(@B1562=1):GOTO-27,-6;PEN RED;T
URN67;DRAW26;TURN-90;DRAW3;TURN-90;FIL
L32
5085 E:
5090 *MCP
5095 GR:PEN RED;GOTO-4,15;TURNTO45;2(D
RAW3;TURN45;DRAW15;TURN135);FILL3
```

```
5100 GR(@B1570=1):GOTO-2,18;DRAW9
5105 E:
5110 *GORILLAPIC
5115 GR:PEN YELLOW;TURNTO270
5120 GR(@B1568=1):PEN ERASE
5125 GR:GOTO32,4;DRAW1;TURN90;DRAW2;TU
RN-45;DRAW3;TURN90;DRAW1;TURN-45;DRAW3
;TURN90;DRAW2
5130 GR:TURN90;DRAW2;TURNTO90;DRAW1;TU
RN-90;DRAW3;TURN-90;DRAW4;2(TURN90;DRA
W1);TURN-90
5135 GR:DRAW2;TURN-90;DRAW4;2(TURN-90;
DRAW2);TURN90;DRAW1;TURN90;DRAW5;TURN-
90;DRAW3
5140 GR:TURN-90;DRAW1;TURN-90;DRAW2;TU
RN90;DRAW2;TURN90;DRAW4;GOTO27,8;TURNT
O225;DRAW3;TURN-45
5145 GR:DRAW2;TURN90;DRAW1;GOTO28,15;G
OTO30,8;GOTO25,15;GOTO31,15
5150 GR:GOTO29,12;DRAW2;GOTO29,11;DRAW
2;GOTO29,10;DRAW2;GOTO29,9;DRAW2
5155 GR(@B1568=1):PEN BLUE;TURNTO0;GOT
O18,4;DRAW1;TURN90;DRAW3;TURN-135;DRAW
3
5160 GR(@B1568=1):PEN YELLOW;TURN90;DR
AW1;TURN45;DRAW4;TURN45;DRAW3;TURNTO0;
4(DRAW3;TURN90)
5165 GR(@B1568=1):TURN-90;DRAW5;TURN45
;FILL3;GOTO27,8;TURN135;DRAW1;GOTO27,7
;DRAW1
5170 GR(@B1568=1):PEN BLUE;GOTO25,7;TU
RN45;DRAW3;TURN-45;DRAW3
5175 E:
5180 *DOGPIC
5185 GR:PEN YELLOW;TURNTO0
5190 GR(@B1557=1):PEN ERASE
5195 GR:GOTO-40,-4;DRAW1;2(DRAW2;TURN9
0;DRAW4;TURN90);GO2;TURN-45;DRAW 3
5200 GR:GOTO-33,-2;DRAW3;GOTO-33,-2;TU
RN-45;DRAW6;GOTO-35,-1;TURN-90;GOTO-36
,-3;DRAW1
5205 E:
5210 *FIREPIC
5215 C:#X=-48
```

```
5220 C:#Y=-5
5225 A:=$ROOM
5230 M:25 ,31 ,
5235 CY:#X=36
5240 CY:#Y=0
5245 GR:PEN RED;TURNTO0;GOTO#X,#Y;2(DR
AW21;TURN45;DRAW21;TURN135)
5250 GR:GOTO#X+4,#Y+4;2(DRAW18;TURN45;
DRAW14;TURN135);GOTO#X,#Y+23;TURN45;DR
AW20
5255 C:#P=#Y+3
5260 *BRICKS
5265 GR:GOTO#X+1,#P;DRAW3
5270 C:#P=#P+3
5275 J(#P<16):*BRICKS
5280 GR:PENBLUE;GOTO#X+6,#Y+8;TURNTO0;
DRAW1;TURN90;DRAW2;TURN-90;DRAW1
5285 GR:GOTO#X+10,#Y+12;DRAW1;TURN-90;
DRAW4;TURN180;DRAW7;TURN-90;DRAW1
5290 GR:PEN RED
5295 GR(@B1567=1):PEN YELLOW
5300 GR:GOTO#X+5,#Y+15;TURNTO45;DRAW3;
TURN90;DRAW3;TURNTO45;GO2;DRAW3
5305 E:
5310 *SINKPIC
5315 GR:PEN RED;TURNTO0;GOTO-48,5;2(DR
AW2;TURN90;DRAW6;TURN90);GO1;TURN90;DR
AW6;TURN-90;GO1;TURN-90
5320 GR:DRAW6;TURN90;GO1;PEN BLUE;2(TU
RN45;DRAW6);TURN135;DRAW6;2(TURN45;DRA
W5;TURN135;DRAW5)
5325 GR:2(TURN45;DRAW3;TURN135;DRAW3);
2(TURN45;DRAW2;TURN135;DRAW2);TURN45;D
RAW2
5330 GR:PEN RED;GOTO-41,8;TURNTO45;DRA
W6;GOTO-41,7;DRAW6
5335 GR:GOTO-41,6;DRAW5;GOTO-41,5;DRAW
5
5340 GR:GOTO-41,5;TURNTO180;DRAW6;GOTO
-37,8;DRAW6
5345 E:
```

Add more words to the vocabulary.

```
1460 M:TAK,GET,DRO,GIV,UNL,OPE,LOO,EXA
,LIG,INS,USE,PUS,GO ,INV,THR,WEA
1465 JM:*TD,*TD,*TD,*TD,*OPEN,*OPEN,*L
OOK,*LOOK,*LIGHT,*USE,*USE,*PUSH,*GO,*
INVENTORY,*THROW,*WEAR
```

And add the necessary routines to go with them. These are routines for the words TAKE, GET, DROP, GIVE.

Note: The first step of the different routines TAKE and DROP have been combined. Since the first step was to find the memory location which held the status of the object in question, and then to decide whether the object could be dropped or taken, combining these two routines was quite easy and saved a great deal of memory.

```
2175 *TD
2180 M:FLA,FUS,PAI,BIS,BON,KEY,CRO,GLO
,WAT,BUC,BAN,DRU,
2185 JM:*FLASHLIGHT,*FUSES,*PAINTING,*
BISCUITS,*BONE,*KEY,*CROWBAR,*GLOVES,*
WATER,*BUCKET,*BANANA,*DRUGS
2190 M:GET,TAK
2195 TY:[esc shift clear]I CAN'T TAKE
THAT!!
2200 JN:*DONTHAVE
2205 J:*WAIT
2210 *FLASHLIGHT
2215 C:#M=1541
2220 J:*CM
2225 *FUSES
2230 C:#M=1542
2235 J:*CM
2240 *PAINTING
2245 C:@B1555=5
2250 C:#M=1543
2255 J:*CM
2260 *BISCUITS
2265 C:#M=1544
2270 J:*CM
2275 *BONE
2280 C:#M=1545
2285 J:*CM
```

```
2290 *CROWBAR
2295 C:#M=1547
2300 J:*CM
2305 *GLOVES
2310 C:#M=1549
2315 J:*CM
2320 *WATER
2325 J(@B1551<>0):*ME8
2330 C:#M=1550
2335 J:*CM
2340 *BUCKET
2345 C:#M=1551
2350 J:*CM
2355 *BANANA
2360 C:#M=1552
2365 J:*CM
2370 *DRUGS
2375 C:#M=1553
2380 J:*CM
```

Since there are three colored keys in the adventure, the player is requested to input the proper color of the key wanted. The program then sets the proper memory location.

```
2385 *KEY
2390 M:BROWN,BLUE,RED
2395 JM:*BROWN,*BLUE,*RED
2400 T:[esc shift clear]WHICH COLOR KE
Y???
2405 J:*WAIT
2410 *BROWN
2415 C:#M=1546
2420 J:*CM
2425 *BLUE
2430 C:#M=1548
2435 J:*CM
2440 *RED
2445 C:#M=1554
2450 J:*CM
```

Here the program actually branches to the proper routines for TAKE and DROP.

```
2455 *CM
2460 M:GET,TAK,DRO,GIV
2465 JM:*GET,*GET,*DROP,*DROP
```

The TAKE routine checks to see that the player already has the object and is in the correct room with the object by checking the status of the memory location set in the combined TAKE and DROP routine above.

Example: The player wants to take the red key, and therefore types GET RED KEY and hits RETURN. The program quickly scans the words and, finding a match, jumps to the routine *TD. There it finds a match to the word KEY and jumps to the routine *KEY, where it finds a match to the color RED and sends it to the proper routine to set the #M (memory location variable) to the value of 1554. The program then continues to the *CM routine, where it again sees the match to the word GET and sends the program to the routine below where the value of the memory location for that item is compared to ascertain that it is legal for the player to do what is wanted (e.g., Does the player already have the red key and, if not, is the key really in the room with the player?).

```
2470 *GET
2475 J(@B#M=0):*HAVE
2480 J(@B#M<>#R):*NOTHERE
2485 C:@B#M=0
2490 C:#L=0
2495 U:*DRAWPICTURE
2500 J:*OK
```

Add the routine to drop an item.

Note: When an item is dropped, it is assigned the same value as the room. This is how the computer knows the item is in that room.

```
2505 *DROP
2510 J(@B#M<>0):*DONTHAVE
2515 C:@B#M=#R
2520 C:#L=1
```

```
2525 J:*OK
2530 *HAVE
2535 T:[esc shift clear]YOU ALREADY HA
VE IT!!
2540 J:*WAIT
2545 *NOTHERE
2550 T:[esc shift clear]THAT ITEM IS N
OT HERE!!
2555 J:*WAIT
2560 *DONTHAVE
2565 T:[esc shift clear]YOU AREN'T CAR
RYING THAT!!
2570 J:*WAIT
```

Now add the routine for the words OPEN and UNLOCK.

Note: The program checks for items that can be opened, then opens them or displays the appropriate message.

```
2610 *OPEN
2615 M:CAB,FUSEB,BOOKC,DOO
2620 JM:*CABOP,*FUSEBOP,*BOOKCOP,*DOOP
2625 M:BOO,FUS
2630 TY:[esc shift clear]PLEASE TYPE T
HE WHOLE WORD!!
2635 JY:*WAIT
2640 T:[esc shift clear]I CAN'T OPEN T
HAT!!
2645 J:*WAIT
```

Since there are three cabinets in the adventure, in order for the program to know which cabinet the player wants to open it is necessary for it to check which room the player is in.

```
2650 *CABOP
2655 A:=$ROOM
2660 M:M2 ,M8 ,35 ,
2665 JM:*CAB1,*CAB2,*CAB3
2670 J:*NOTHERE
2675 *CAB1
2680 J(@B1561=1):*ALREADYOPEN
2685 C:@B1561=1
```

```
2690 C:@B1542=2
2695 U:*CP
2700 U:*OPCAB
2705 J:*OK
2710 *CAB2
2715 J(@B1563=1):*ALREADYOPEN
2720 C:@B1563=1
2725 C:@B1544=8
2730 C:@B1545=8
2735 U:*CP
2740 U:*OPCAB
2745 J:*OK
2750 *CAB3
2755 J(@B1570=1):*ALREADYOPEN
2760 C:@B1570=1
2765 C:@B1553=35
2770 U:*MCP
2775 U:*OPCAB
2780 J:*OK
```

When a cabinet is opened, the program changes the string value from CABINET to OPEN CABINET.

```
2985 *OPCAB
2990 C:$OBJECT= OPEN$OBJECT
2995 E:
```

To open the fusebox:

```
2785 *FUSEBOP
2790 J(#R<>5):*NOTHERE
2795 J(@B1555<>5):*NOTHERE
2800 J(#R=5):*LOOK
2805 J:*OK
```

To open the bookcase:

```
2810 *BOOKCOP
2815 J(#R<>7):*NOTHERE
2820 J(@B1562=1):*ALREADYOPEN
2825 C:@B1562=1
```

```
2830 U:*BOOKCPIC
2835 C:#S=11
2840 J:*OK
```

To open the door (again, the room the player is in must be checked as there are 5 locked or jammed doors in the adventure):

```
2845 *DOOP
2850 A:=$ROOM
2855 M:M4  ,18 ,19  ,20  ,34  ,
2860 JM:*D4,*D18,*D19,*D20,*D34
2865 J:*OK
2870 *D4
2875 J(@B1559<>1):*ME1
2880 J:*ALREADYOPEN
2885 *D18
2890 J(@B1548<>0):*ME2
2895 C:#S=21
2900 C:@B1566=1
2905 J:*OK
2910 *D19
2915 J(@B1547<>0):*ME3
2920 C:#S=22
2925 C:@B1564=1
2930 J:*OK
2935 *D20
2940 J(@B1546<>0):*ME4
2945 C:#N=17
2950 C:@B1565=1
2955 J:*OK
2960 *D34
2965 J(@B1554<>0):*ME5
2970 C:#N=30
2975 C:@B1569=1
2980 J:*OK
```

Various messages help the player find what is needed to open a door:

```
3000 *ME1
3005 T:[esc shift clear]ELECTRICITY IS
 NEEDED FOR THAT!!
3010 J:*WAIT
3015 *ME2
```

```
3020 T:[esc shift clear]YOU NEED THE B
LUE KEY!!
3025 J:*WAIT
3030 *ME3
3035 T:[esc shift clear]IT'S STUCK!!
3040 J:*WAIT
3045 *ME4
3050 T:[esc shift clear]YOU NEED THE B
ROWN KEY!!
3055 J:*WAIT
3060 *ME5
3065 T:[esc shift clear]YOU NEED THE R
ED KEY!!
3070 J:*WAIT
```

All other doors are already open.

```
3075 *ALREADYOPEN
3080 T:[esc shift clear]IT'S ALREADY O
PEN!!
3085 J:*WAIT
```

Routines for the words LOOK and EXAMINE:

```
3505 *LOOK
3510 C(#R=20):@B1549=#R
3515 J(#R=20):*OK
3520 T(#R=5):[esc shift clear]THERE AR
E FUSES MISSING!!
3525 J(#R=5):*WAIT
3530 T:[esc shift clear]I SEE NOTHING
SPECIAL!!
3535 J:*WAIT
```

The routines for the word GO:

Note: The program checks for valid places to which the player can go.

```
3540 *GO
3545 M:CHI,FIR,SHA,GOR,DOG,CHU
3550 JM:*GOFIR,*GOFIR,*GOSHA,*GOGOR,*G
ODOG,*GOCHU
3555 J:*CANTGO
```

To go into the fireplace.

```
3560 *GOFIR
3565 A:=$ROOM
3570 M:26 ,32 ,
3575 JN:*NOTHERE
3580 J(@B1567<>1):*DEAD5
3585 C(#R=26):#R=25
3590 C(#R=32):#R=31
3595 J:*CR
```

To go to the gorilla:

```
3600 *GOGOR
3605 J(#R<>28):*NOTHERE
3610 J(@B1568<>1):*DEAD2
3615 J:*OK
```

To go to the dog:

```
3620 *GODOG
3625 J(#R<>9):*NOTHERE
3630 J(@B1557=0):*DEAD3
3635 J:*NOTHERE
```

To go into the laundry chute:

```
3640 *GOCHU
3645 J(#R<>11):*ME7
3650 C:#R=20
3655 C:#F=#F-1
3660 J:*CR
```

To go into the elevator shaft:

```
3665 *GOSHA
3670 J(#R<>22):*ME7
3675 C:#R=27
3680 J:*CR
```

The routines for the words USE and INSERT:

```
3175 *USE
3180 M:FUS,DRU,KEY,CRO
```

```
3185 JM:*USEFUS,*USEDRU,*USEKEY,*USECR O
3190 J:*CANTDO
```

Player wants to use fuses:

```
3195 *USEFUS
3200 J(@B1542<>0):*DONTHAVE
3205 J(#R<>5):*ME7
3210 C:@B1559=1
3215 C:@B1542=100
3220 T:[esc shift clear]THE LIGHTS COM
E ON!!
3225 J:*WAIT
```

To use the drugs:

```
3230 *USEDRU
3235 J(@B1553<>0):*DONTHAVE
3240 J(@B1552<>0):*ME8
3245 C:@B1571=1
3250 T:[esc shift clear]THE DRUGS ARE
IN THE BANANA!!
3255 J:*WAIT
```

To use the key:

```
3260 *USEKEY
3265 J(@B1546=0):*OK
3270 J(@B1548=0):*OK
3275 J(@B1554=0):*OK
3280 J:*DONTHAVE
```

To use the crowbar:

```
3285 *USECRO
3290 J(@B1547<>0):*DONTHAVE
3295 J(#R<>19):*ME7
3300 C:@B1564=1
3305 J:*OK
```

Various other messages help the player through the adventure:

```
3310 *ME7
3315 T:[esc shift clear]YOU CANT DO TH
```

```
AT HERE!!
3320 J:*WAIT
3325 *ME8
3330 T:[esc shift clear]IN WHAT!!
3335 J:*WAIT
3340 *CANTDO
3345 T:[esc shift clear]YOU CAN'T DO T
HAT!!
3350 J:*WAIT
```

The routine for the word LIGHT:

```
3120 *LIGHT
3125 M:FLA
3130 TN:[esc shift clear]I CAN'T LIGHT
 THAT!!!
3135 JN:*WAIT
3140 J(@B1541<>0):*DONTHAVE
3145 J(@B1558=1):*LIT
3150 C:@B1558=1
3155 J:*OK
3160 *LIT
3165 T:[esc shift clear]IT'S ALREADY L
IT!!
3170 J:*WAIT
```

The routine for the word PUSH:

```
3355 *PUSH
3360 M:BOOKC,GOR
3365 JM:*BOOKCOP,*PUSHGO
3370 T:[esc shift clear]THAT DOESN'T H
ELP!!
3375 J:*WAIT
3380 *PUSHGO
3385 J(#R<>28):*NOTHERE
3390 J:*DEAD2
```

The routine for the word INVENTORY:

Note: To display a picture of the items the player is carrying, use the same graphic routines that drew the items in the rooms.

```
1510 *INVENTORY
1515 C:$ITEMS= NOTHING
1520 C:#I=0
1525 C:#M=1541
1530 C:#X=-16
1535 C:#Y=8
```

Clear the picture of the room and draw the pictures of the items in the inventory.

```
1540 GR:CLEAR
1545 J:*INV2
1550 *INV2
1555 U(@B#M=0):*VISIBLE
1560 C:#M=#M+1
1565 J(#M<1557):*INV2
```

Then print the item names, and if the player is wearing the gloves, print that also.

```
1570 T:[esc shift clear]$ITEMS
1575 T(@B1549=50):WEARING GLOVES
1580 J:*PRESSKEY
1585 *VISIBLE
```

The Presskey routine is needed for above.

Note: The routine will only respond if the space bar is pressed. The program will continue to loop until it finds the internal keyboard code 33 stored in memory location 764. This memory location always retains the internal code of the last key pressed. The internal code of the space bar is 33. (See Appendix B for internal code of each key.)

```
3685 *PRESSKEY
3690 T:[inverse/4 spaces]PRESS SPACE B
AR TO CONTINUE[4 spaces/inverse off]\
3695 C:@B764=255
3700 *LOOP2
3705 J(@B764=33):*CR
3710 J:*LOOP2
```

The routine for the word THROW:

```
3395 *THROW
3400 M:BAN,WAT,BIS,BON
3405 JM:*THRBAN,*THRWAT,*THRBIS,*THRBO
N
3410 T:[esc shift clear]WHY DO YOU WAN
T TO THROW THAT!!
3415 J:*WAIT
```

To throw the banana:

```
3420 *THRBAN
3425 J(@B1552<>0):*DONTHAVE
3430 C:@B1552=#R
3435 J(#R=28):*BANGOR
3440 J:*OK
3445 *BANGOR
```

If the banana is thrown in the correct room, the gorilla will eat it.

```
3450 T:[esc shift clear]THE GORILLA EA
TS THE BANANA!!
3455 C:@B1552=100
3460 J(@B1571=1):*SLEEP
3465 J:*WAIT
```

If the drugs are inserted into the banana, the gorilla will fall asleep after eating it.

```
3470 *SLEEP
3475 C:@B1568=1
3480 U:*GORILLAPIC
3485 C:#E=29
3490 C:$OBJECT= SLEEPING GORILLA
3495 C:@B1552=100
3500 J:*P
```

Other items that can be thrown:

Water

```
3715 *THRWAT
3717 J(@B1567=1):*ALREADYOUT
3718 J(@B1550<>0):*ME6
3720 C:@B1550=23
3725 J(#R<>26):*OK
3730 C:@B1567=1
3735 C:$OBJECT= FIREPLACE
3740 U:*FIREPIC
3745 J:*OK
```

Biscuits

```
3750 *THRBIS
3755 J(@B1544<>0):*DONTHAVE
3760 C:@B1544=#R
3765 J(#R<>9):*OK
3770 T:[esc shift clear]THE DOG EATS T
HE BISCUIT!!
3775 C:@B1544=8
3780 J:*WAIT
```

Bone

```
3785 *THRBON
3790 J(@B1545<>0):*DONTHAVE
3795 C:@B1545=#R
3800 J(#R<>9):*OK
3805 C:@B1557=1
3810 C:#W=7
3815 C:@B1545=100
3820 T:[esc shift clear]THE DOG LEAVES
 TO BURY THE BONE!!
3825 C:$OBJECT= NOTHING
3830 U:*DOGPIC
3835 J:*WAIT
```

Fire already out:

```
3090 *ALREADYOUT
3095 T:[esc shift clear]IT'S ALREADY O
UT!!
3100 J:*WAIT
```

Water is needed:

```
3105 *ME6
3110 T:[esc shift clear]YOU NEED SOME
WATER!!
3115 J:*WAIT
```

The routine for the word WEAR:

```
3840 *WEAR
3845 M:GLO
3850 TN:[esc shift clear]I CAN'T WEAR
THAT!!!
3855 JN:*WAIT
3860 J(@B1549<>0):*DONTHAVE
```

When the gloves are worn, the program sets the value of the memory location which holds their status to 50.

```
3865 C:@B1549=50
3870 J:*OK
```

There are many ways for the player to get 'clobbered.' When he does, the program comes here:

```
5720 *DEAD1
5725 T:[esc shift clear]YOUR HANDS SLI
P AND YOU FALL!!
5730 C:#P=31
5735 *REPEAT
5740 SO:#P
5745 C:#P=#P-1
5750 J(#P<>0):*REPEAT
5755 J:*DEATHMARCH
5760 *DEAD2
```

```
5765 T:[esc shift clear]THE GORILLA TE
ARS YOU LIMB FROM LIMB!!!
5770 J:*DEATHMARCH
5775 *DEAD3
5780 T:[esc shift clear]THE DOG ATTACK
S!!!
5785 J:*DEATHMARCH
5805 *DEAD5
5810 T:[esc shift clear]YOU ARE COOKED
 ALIVE!!!
5815 J:*DEATHMARCH
```

There is only one way to win—reach room 50.

```
6075 *WIN
6080 GR:QUIT
6085 C:@B1373=16 [TEXT WINDOW
6090 C:@B1374=2
6095 WRITE:S,
6100 POS:3,4
6105 WRITE:S,you escaped!
6110 C:#P=0
6115 SO:20
6120 PA:2
6125 U:*PAUSE
6130 SO:22
6135 PA:2
6140 U:*PAUSE
6145 SO:24
6150 PA:2
6155 U:*PAUSE
6160 SO:26
6165 PA:16
6170 U:*PAUSE
6175 SO:22
6180 PA:2
6185 U:*PAUSE
6190 SO:26
6195 PA:32
6200 SO:0
6205 WRITE:S,
6210 WRITE:S, CONGRATULATIONS!!
6215 CLOSE:S
```

```
6220 J:*PLAYAGAIN
```

The ESCAPE program is now complete. Be sure to save a copy onto tape with CSAVE , or onto disk with SAVE "D:filename."

A GRAPHIC ADVENTURE IN
ATARI MICROSOFT BASIC

THE HUNTER

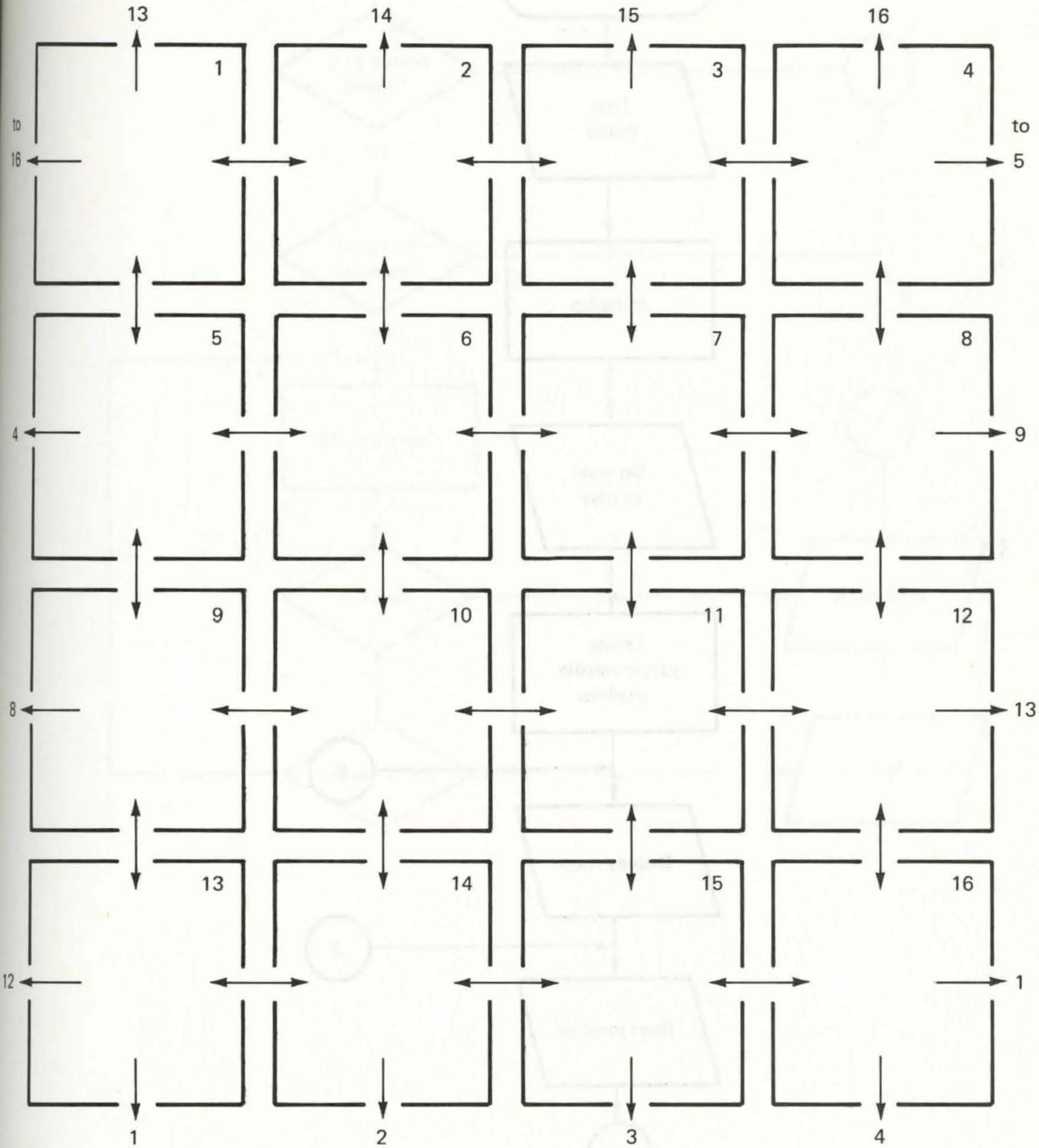
Atari Microsoft BASIC has been greatly expanded from standard Microsoft BASIC to allow easy access to the Atari graphics capabilities. For example, player-missile graphics in Atari BASIC are not easily utilized because strings must be allocated to hold player-missile data and special routines used to access this information. Atari Microsoft BASIC has special features which allow easy reservation of memory for alternate character sets and player-missile graphics. With the simple command OPTION PLM1 you can reserve 1280 bytes in memory for player-missile graphics. OPTION CHR1 reserves 1024 bytes in memory for alternate character data. Also, very little effort is required by the programmer to move player-missile graphics in memory. This is accomplished with the use of the keyword MOVE, which greatly simplifies the vertical movement of player-missile graphics, usually a very slow process for Atari BASIC. These features are utilized in **The Hunter** program to help you understand them.

The Hunter requires 32K when loaded from disk or cassette.

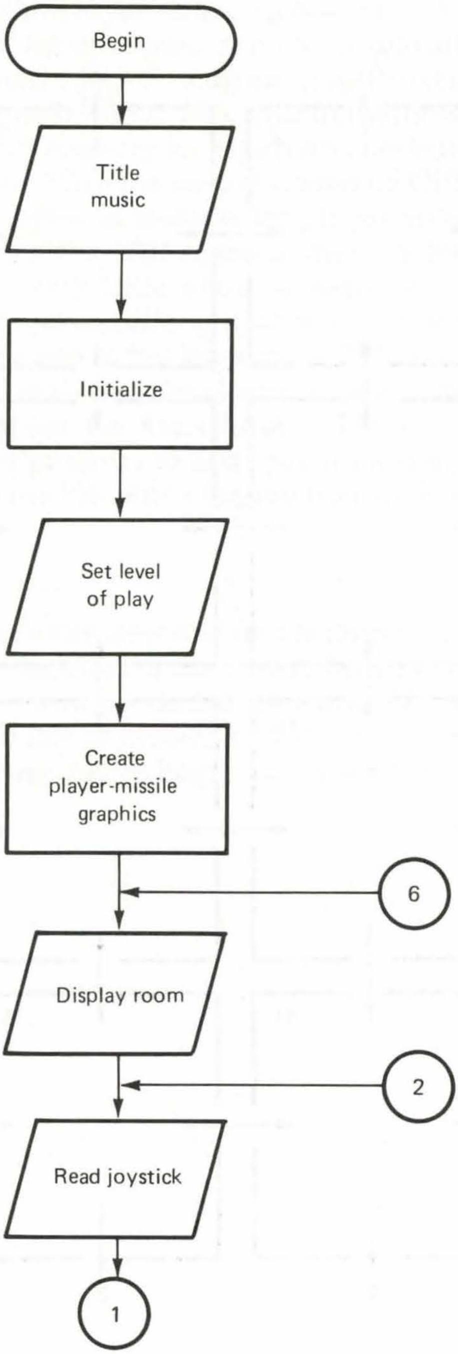
THE OBJECTIVE

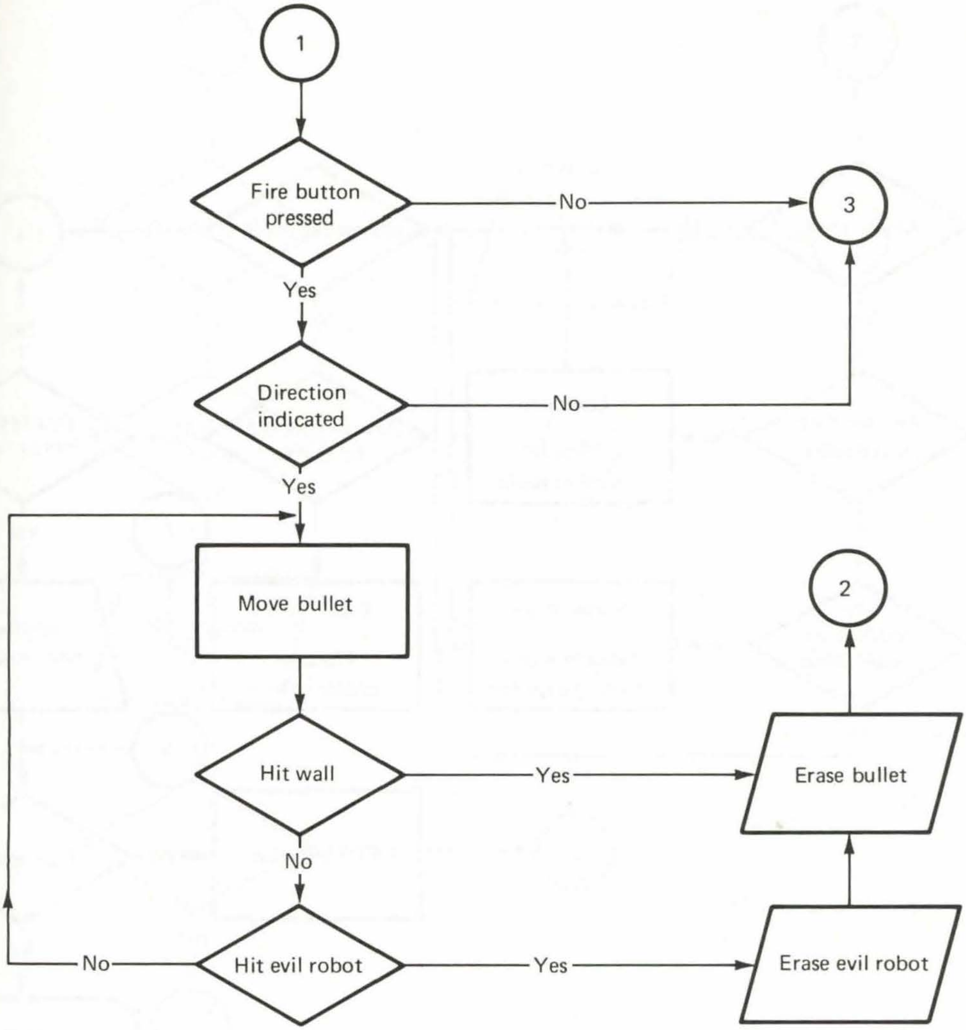
To search the dungeon's 16 rooms and locate the magic key which will reveal the hidden location of the crown. Only after the magic key is found will the crown be visible. However, if you allow an evil robot to touch the key or crown, he will steal it and place it in another room, forcing you to begin your search again.

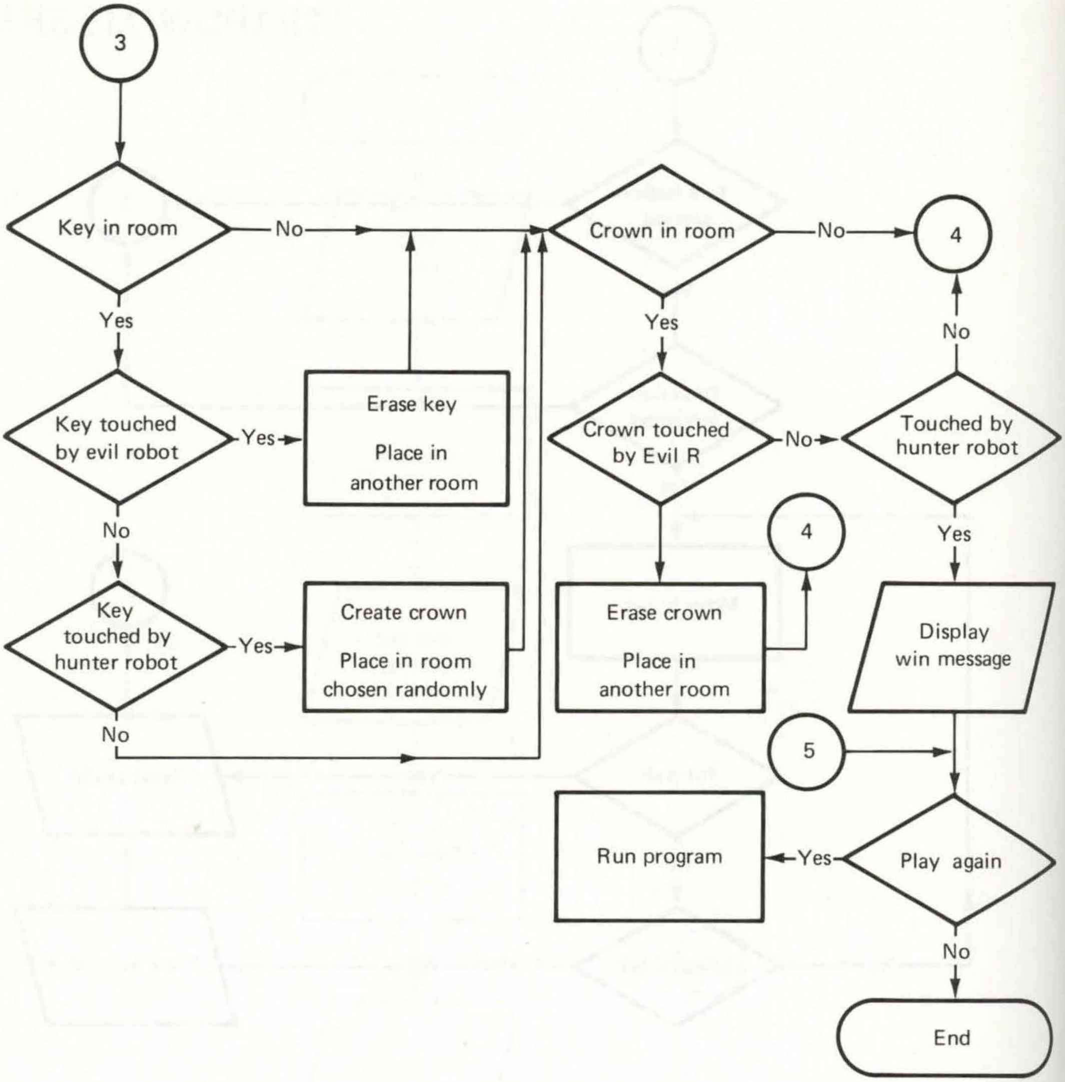
THE MAP

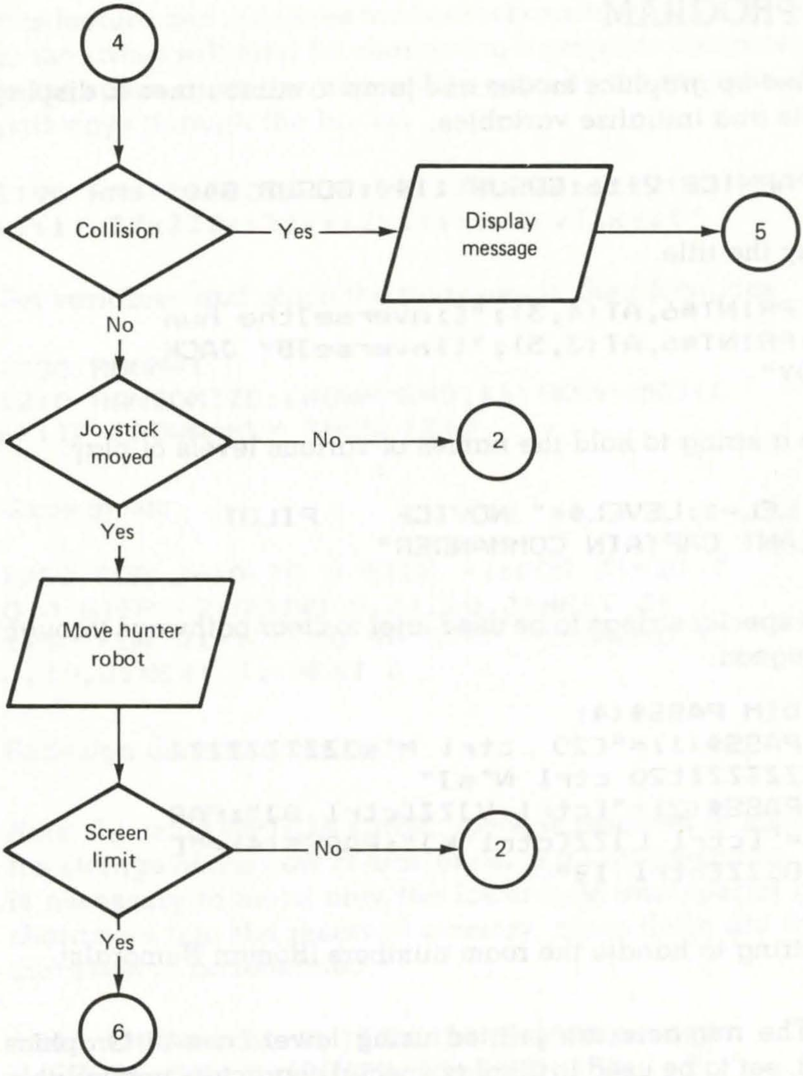

13
14
15
16
1
2
3
4
to
16
to
5
5
6
7
8
4
9
9
10
11
12
8
13
13
14
15
16
12
1
1
2
3
4

THE FLOWCHART




1
Fire button pressed
No
3
Yes
Direction indicated
No
Yes
Move bullet
2
Hit wall
Yes
Erase bullet
No
No
Hit evil robot
Yes
Erase evil robot


3
Key in room
No
Yes
Key touched by evil robot
Yes
Erase key
Place in another room
No
Key touched by hunter robot
Yes
Create crown
Place in room chosen randomly
No
Crown in room
No
4
Yes
Crown touched by Evil R
No
Touched by hunter robot
No
Erase crown
Place in another room
4
Yes
Display win message
5
Run program
Yes
Play again
No
End


4
Collision
Yes
Display message
5
No
Joystick moved
No
2
Yes
Move hunter robot
Screen limit
No
2
Yes
6

THE PROGRAM

First, set up graphics modes and jump to subroutines to display the title and initialize variables.

```
10 GRAPHICS 2+16:GOSUB 1140:GOSUB 840
```

Display the title.

```
1140 PRINT#6,AT(4,3);"[inverse]the hun
ter":PRINT#6,AT(3,5);"[inverse]BY JACK
 HARDY"
```

Create a string to hold the names of various levels of play.

```
1150 LEL=1:LEVEL$=" NOVICE    PILOT
SERGEANT CAPTAIN COMMANDER"
```

Create special strings to be used later to clear pathways through the dungeon.

```
1160 DIM PASS$(4)
1170 PASS$(1)="[20  ctrl M's]ZZZZZZZZZ
ZZZZZZZZZZZ[20 ctrl N's]"
1180 PASS$(2)="[ctrl V]ZZ[ctrl B]":PAS
S$(3)="[ctrl L]ZZ[ctrl K]":PASS$(4)="[
ctrl O]ZZ[ctrl I]"
```

Set a string to handle the room numbers (Roman Numerals).

Note: The numbers are printed using lower case in Graphics mode 2, set to be used to display special characters unavailable by using a normal Graphics mode 2. There are two character sets for modes 1 and 2. The standard character set contains the usual upper-case letters, numbers, and punctuation. For this program, the alternate character set which contains special graphics characters and lower-case letters is used. To see the effects of this, type in direct mode GRAPHICS 2:POKE 756,226, hit RETURN and there will be a screen full of hearts. The program uses this feature and redefines the heart character into bricks. The Z's in the string will later be changed to a graphics character equal to the space character and will be used by the program to open pathways through the bricks.

```
1190 RM$="iZZZiiZZiiiZivZZvZZZviZZviiZ
viiiixZZxZZZxiZZxiiZxiiixivZxvZZxviZ"
```

Set variables and place the treasures in their locations.

```
1200 ROOM=1
1210 RANDOMIZE:CROWN=RND(16):KEY=RND(1
6):IF CROWN=KEY THEN 1210
```

Some music:

```
1220 FOR J=10 TO 0 STEP -1:FOR J1=20 T
O 0 STEP -2:SOUND 0,J1,10,J:NEXT J1
1230 FOR J1=120 TO 90 STEP -2:SOUND 1,
J,10,J:NEXT J1:NEXT J
```

Redesign the character set.

Note: By using OPTION CHR2, space is reserved in memory for the storage of the new character set (e.g., bricks-space). Then it is necessary to move only the lower-case and special graphic characters into the reserved memory, since these are the ones that need to be redefined.

```
1240 CHBAS=756:OPTION CHR2:ADDR%=VARPT
R(CHR2):PAGE%=(ADDR%/256)AND 255
1250 MOVE 57856,ADDR%,512:RESTORE 1390
```

Change the heart character into a set of bricks.

```
1260 OF=0*8:FOR I=0 TO 7:READ CH:POKE
ADDR%+OF+I,CH:NEXT
1390 DATA 251,251,251,0,223,223,223,0
```

Change the graphic character produced by using ctrl C into a large ┘-shaped character to be used later in forming the corner of a passage.

```
1270 OF=3*8:FOR I=0 TO 7:READ CH:POKE
ADDR%+OF+I,CH:NEXT
1400 DATA 3,3,3,3,3,3,255,255
```

Change the graphic character produced by using ctrl E into a large ┐-shaped corner.

```
1280 OF=5*8:FOR I=0 TO 7:READ CH:POKE
ADDR%+OF+I,CH:NEXT
1410 DATA 255,255,3,3,3,3,3,3
```

Change the graphic character produced by using ctrl I into a lower right corner square.

```
1290 OF=9*8:FOR I=0 TO 7:READ CH:POKE
ADDR%+OF+I,CH:NEXT
1420 DATA 0,0,0,0,0,0,3,3
```

Change the graphic character produced by using ctrl K into upper right and ctrl L character into upper left corner squares.

```
1300 OF=11*8:FOR I=0 TO 15:READ CH:POK
E ADDR%+OF+I,CH:NEXT
1430 DATA 3,3,0,0,0,0,0,0,192,192,0,0,
0,0,0,0
```

Change the graphic character produced by using ctrl O into lower left corner square.

```
1310 OF=15*8:FOR I=0 TO 7:READ CH:POKE
 ADDR%+OF+I,CH:NEXT
1440 DATA 0,0,0,0,0,0,192,192
```

Change the graphic character produced by using ctrl Q into large ┌-shaped corner.

```
1320 OF=17*8:FOR I=0 TO 7:READ CH:POKE
 ADDR%+OF+I,CH:NEXT
1450 DATA 255,255,192,192,192,192,192,
192
```

Change the graphic character produced by using ctrl Z into large L-shaped corner.

```
1330 OF=26*8:FOR I=0 TO 7:READ CH:POKE
 ADDR%+OF+I,CH:NEXT
1460 DATA 192,192,192,192,192,192,255,
255
```

Change lower-case i into capital I for use in the Roman numeral. (*Note:* Since the graphic mode used will not display capital letters, the lower-case letter used must be changed into upper case for display.)

```
1340 OF=41*8:FOR I=0 TO 7:READ CH:POKE
 ADDR%+OF+I,CH:NEXT
1470 DATA 0,126,24,24,24,24,126,0
```

Change lower-case r to upper-case R.

```
1350 OF=50*8:FOR I=0 TO 7:READ CH:POKE
 ADDR%+OF+I,CH:NEXT
1480 DATA 0,124,102,102,124,108,102,0
```

Change lower-case v to upper-case V.

```
1360 OF=54*8:FOR I=0 TO 7:READ CH:POKE
 ADDR%+OF+I,CH:NEXT
1490 DATA 0,102,102,102,102,60,24,0
```

Change lower-case x to upper-case X.

```
1370 OF=56*8:FOR I=0 TO 7:READ CH:POKE
 ADDR%+OF+I,CH:NEXT
1500 DATA 0,102,102,60,60,102,102,0
```

Finally, the lower-case z is erased to be used as the blank character.

```
1380 OF=58*8:FOR I=0 TO 7:READ CH:POKE
 ADDR%+OF+I,CH:NEXT
1510 DATA 0,0,0,0,0,0,0,0
```

Let player select level he wishes to play.

```
1520 PRINT#6,AT(4,8);"select LEVEL":PR
INT#6,AT(4,11);"start TO PLAY";
1530 PRINT#6,AT(5,9);MID$(LEVEL$,LEL*9
-8,9):AMO=36-(LEL*4)
1540 IF PEEK(53279)=6 THEN 1570
1550 IF PEEK(53279)=5 THEN LEL=LEL+1:S
OUND 0,150,10,10,10:IF LEL>5 THEN LEL=
1
1560 FOR J=1 TO 100:NEXT:GOTO 1530
```

Set beginning position of the Hunter Robot. Reserve memory for double-line player-missile graphic character resolution. Tell the computer that the program is using double-line resolution by POKing a 46 into location 559. (POKE with a 62 for single-line resolution.) Turn on player-missile graphics by POKing a 3 into 53277 (&D01D hex). Set horizontal position. Double the width of player and missile by POKing a 1 into locations 53256 (&D008 hex) and 53260 (&D00C hex). Set priority over playfields. Set color of hunter.

```
1570 X=120:Y=82
1580 OPTION PLM2:POKE 559,46:POKE &D00
C,1:POKE &D01D,3:POKE &D000,X:POKE &D0
08,1:POKE &26F,1
1590 SETCOLOR 0,9,8
```

The program goes to 1610 to define the Hunter Robot, returns, points to the alternate graphics mode 2 character set, and clears the screen.

```
1600 GOSUB 1610:POKE CHBAS,PAGE%:PRINT
#6,"[esc ctrl clear]":RETURN
```

Design the shape of the Hunter Robot.

```
1610 RESTORE 1630:FOR J=VARPTR(PLM2)+1
28+Y TO VARPTR(PLM2)+137+Y:READ CH:POK
E J,CH
1620 NEXT J
```

The shape table of the Hunter Robot:

```
1630 DATA 0,16,16,16,56,40,124,254,56,
0
1640 POKE &D01D,3:RETURN
```

Find initial room and jump to proper graphics display.

```
840 ON ROOM GOSUB 870,1030,970,900,103
0,970,900,870,970,900,870,1030,900,870
,1030,970
```

The routines to print the shapes of the rooms:

Shape of rooms #1, #8, #11, #14:

```
870 PRINT #6,AT(0,4);PASS$(1);
880 FOR CL=0 TO 3:PRINT #6,AT(8,CL);PA
SS$(2):NEXT:PRINT#6,AT(8,4);PASS$(3)
890 PRINT#6,AT(8,6);PASS$(4):FOR CL=7
TO 11:PRINT#6,AT(8,CL);PASS$(2);:NEXT:
RETURN
```

Shape of rooms #4, #7, #10, #13:

```
900 FOR CL=0 TO 11:PRINT#6,AT(8,CL);PA
SS$(2):NEXT
```

```
910 PRINT#6,AT(0,4);PASS$(1)
920 PRINT#6,AT(4,1);"[ctrl Q/3 ctrl M'
s/ctrl L]ZZ[ctrl K/3 ctrl M's/ctrl E]"
:FOR CL=2 TO 3:PRINT#6,AT(4,CL);"[ctrl
 V]ZZZZZZZZZZ[ctrl B]":NEXT
930 PRINT#6,AT(4,4);"[ctrl L]ZZ[6 ctrl
 N's]ZZ[ctrl K]":PRINT#6,AT(4,5);"ZZZ
     ZZZ":PRINT#6,AT(4,6);"[ctrl O]ZZ[
6 ctrl M's]ZZ[ctrl I]"
940 FOR CL=7 TO 8:PRINT#6,AT(4,CL);"[c
trl V]ZZZZZZZZZZ[ctrl B]":NEXT
950 PRINT#6,AT(4,9);"[ctrl Z/3 ctrl N'
s/ctrl O]ZZ[ctrl I/3 ctrl N's/ctrl C]"
960 RETURN
```

Shape of rooms #3, #6, #9, #16:

```
970 PRINT#6,AT(0,4);PASS$(1):FOR CL=0
TO 2:PRINT#6,AT(8,CL);PASS$(2):PRINT#6
,AT(8,CL+9);PASS$(2);:NEXT
980 PRINT#6,AT(3,2);"[ctrl Q/4 ctrl M'
s/ctrl L]ZZ[ctrl K/4 ctrl M's/ctrl E]"
990 PRINT#6,AT(3,3);"[ctrl V]ZZZZZZZZZ
ZZZ[ctrl B]"
1000 PRINT#6,AT(3,4);"[ctrl L]ZZZZZZZZ
ZZZZ[ctrl K]":PRINT#6,AT(3,5);"ZZZZZZZ
ZZZZZZZ":PRINT#6,AT(3,6);"[ctrl O]ZZZZ
ZZZZZZZZ[ctrl I]"
1010 FOR CL=7 TO 8:PRINT#6,AT(3,CL);"[
ctrl V]ZZZZZZZZZZZZ[ctrl B]":NEXT
1020 PRINT#6,AT(3,9);"[ctrl Z/4 ctrl N
's/ctrl O]ZZ[ctrl I/4 ctrl N's/ctrl C]
":RETURN
```

Shape of rooms #2, #5, #12, #15:

```
1030 FOR CL=0 TO 11:PRINT#6,AT(8,CL);P
ASS$(2);:NEXT
1040 PRINT#6,AT(0,4);PASS$(1):PRINT#6,
AT(3,7);"[ctrl V]ZZZZZZZZZZZZ[ctrl B]"
:PRINT#6,AT(3,8);"[ctrl Z/4 ctrl N's/c
trl O]ZZ[ctrl I/4 ctrl N's/ctrl C]"
```

```
1050 PRINT#6,AT(5,4);"[ctrl E/2 spaces
/ctrl V]ZZ[ctrl B/2 spaces/ctrl Q]":PR
INT#6,AT(5,5);"[ctrl B/2 spaces/ctrl V
]ZZ[ctrl B/2 spaces/ctrl V]"
1060 PRINT#6,AT(3,6);"[ctrl O]Z[ctrl K
/2 ctrl M's/ctrl L]ZZ[ctrl K/2 ctrl M'
s/ctrl L]Z[ctrl I]"
1070 RETURN
```

After printing the rooms, the program returns to line 850, where it prints the room number.

```
850 PRINT#6,AT(0,10);"[inverse]room":P
RINT#6,AT(0,11);MID$(RM$,ROOM*4-3,4);
860 RETURN
```

Here the main loop begins. First, set the value of S from the joystick. Second, reset collision register and, finally, check to see if the joystick button has been pressed.

```
20 S=PEEK(&278):POKE &D01E,0:IF PEEK(&
D010)=0 THEN 290
```

Check for Key, Crown, and Monster in this room.

```
30 IF KEY=ROOM THEN GOSUB 650
40 IF CROWN=ROOM AND KEY=0 THEN GOSUB
680
50 IF MON THEN GOSUB 200
```

Check for collision with wall or Monster.

```
60 IF PEEK(&D004)=3 THEN 1650
70 IF PEEK(&D00E)=1 THEN 1650
```

Compare S (Joystick value) to see if it is set to a value other than the center.

```
80 IF S=15 THEN 20
```

If other than the center, find the new value

```
90 CX=(S=9 OR S=10 OR S=11)-(S=5 OR S=
6 OR S=7)
100 CY=(S=6 OR S=10 OR S=14)-(S=5 OR S
=9 OR S=13)
```

and move accordingly.

```
110 X=X+CX*3:POKE &D000,X
120 IF CY=+1 THEN MOVE VARPTR(PLM2)+12
8+(Y-2),VARPTR(PLM2)+128+Y,12:Y=Y+2
130 IF CY=-1 THEN MOVE VARPTR(PLM2)+12
8+Y,VARPTR(PLM2)+128+(Y-2),12:Y=Y-2
```

Check to see if Hunter Robot left the screen.

```
140 IF X>200 THEN X=44:GOSUB 250:PRINT
#6,AT(0,0);"[esc ctrl clear]":POKE &D0
00,X:GOSUB 720
150 IF X<44 THEN X=200:GOSUB 250:PRINT
#6,AT(0,0);"[esc ctrl clear]":POKE &D0
00,X:GOSUB 740
160 IF Y>106 THEN GOSUB 190:GOSUB 250:
PRINT#6,AT(0,0);"[esc ctrl clear]":Y=1
2:GOSUB 780:GOS
UB 1610
170 IF Y<12 THEN GOSUB 190:GOSUB 250:P
RINT#6,AT(0,0);"[esc ctrl clear]":Y=10
6:GOSUB 760:GOS
UB 1610
```

If not, send the program back to the main loop.

```
180 GOTO 20
```

Joystick button pushed:

If no direction, return to main loop.

```
290 IF PEEK(&278)=15 THEN 50
```

If direction, subtract ammo, sound on, set bullet location (BX and BY).

```
300 AMO=AMO-1:IF AMO<1 THEN SOUND 0,10
,10,10,2:GOTO 50
310 SOUND 0,100,6,10,5:BX=X:BY=Y
```

Find direction in which joystick was pushed, which is the way the bullet will travel, and jump to proper routine.

```
320 IF PEEK(&278)=14 THEN 420
330 IF PEEK(&278)=11 THEN 540
340 IF PEEK(&278)=13 OR PEEK(&278)=9 O
R PEEK(&278)=5 THEN 480
```

Joystick pushed to the right (bullet will travel to the right):

```
350 BX=BX+6
360 POKE VARPTR(PLM2)+4+BY,3
370 POKE &D004,BX
380 BX=BX+4
390 IF PEEK(&D000)<>0 THEN 640
400 IF PEEK(&D008)>1 THEN 610
410 IF BX>240 THEN POKE VARPTR(PLM2)+4
+BY,0:GOTO 20 ELSE 370
```

Joystick pushed forward (bullet will travel up the screen):

```
420 POKE VARPTR(PLM2)+4+BY,1
430 POKE &D004,BX+4
440 MOVE VARPTR(PLM2)+4+BY,VARPTR(PLM2
)+4+(BY-4),8:BY=BY-4
450 IF PEEK(&D000)<>0 THEN 640
460 IF PEEK(&D008)>1 THEN 610
470 IF BY<15 THEN POKE VARPTR(PLM2)+4+
BY,0:GOTO 20 ELSE 440
```

Joystick pulled back (bullet will travel down the screen):

```
480 POKE VARPTR(PLM2)+4+BY,1
490 POKE &D004,BX+4
500 MOVE VARPTR(PLM2)+4+(BY-4),VARPTR(
PLM2)+4+BY,8:BY=BY+4
510 IF PEEK(&D000)<>0 THEN 640
520 IF PEEK(&D008)>1 THEN 610
530 IF BY>124 THEN POKE VARPTR(PLM2)+4
+BY,0:GOTO 20 ELSE 500
```

Joystick pushed to the left (bullet will travel to the left):

```
540 POKE VARPTR(PLM2)+4+BY,3
550 POKE VARPTR(PLM2)+4+Y,3
560 POKE &D004,BX
570 BX=BX-4
580 IF PEEK(&D000)<>0 THEN 640
590 IF PEEK(&D008)>1 THEN 610
600 IF BX<44 THEN POKE VARPTR(PLM2)+4+
BY,0:GOTO 20 ELSE 560
```

If Evil Robot was hit by a Missile, turn off the Missile, change color of the Robot, create the appropriate sound, erase shape table of Evil Robot, and set variable (MON) to zero.

```
610 POKE VARPTR(PLM2)+4+BY,0
620 FOR CL=12 TO 0 STEP -1:SETCOLOR 2,
4,CL:SOUND 0,20*CL,6,8:FOR T=1 TO 10:N
EXT T:NEXT CL:SOUND 0,0,0,0
630 FOR J=VARPTR(PLM2)+384+RY TO VARPT
R(PLM2)+392+RY:POKE J,0:NEXT:MON=0:GOT
O 20
```

If Evil Robot was not hit, turn off the Missile.

```
640 POKE VARPTR(PLM2)+4+BY,0:GOTO 20
```

If the Key is in the room, the program checks to see if the Evil Robots or the Hunter Robot have touched it.

```
650 IF PEEK(&DOOD)>1 THEN GOSUB 710:KE
Y=RND(16)
660 IF PEEK(&DOOD)=1 THEN GOSUB 710:KE
Y=0
670 RETURN
```

If the Crown is in room, the program checks to see if the Evil Robots or the Hunter Robot have touched it.

```
680 IF PEEK(&DOOD)>1 THEN GOSUB 710:CR
OWN=RND(16)
690 IF PEEK(&DOOD)=1 THEN GOTO 1830
700 RETURN
```

If either the Key or the Crown is touched, the program clears the shape table to have it disappear.

```
710 SOUND 0,200,10,8,10:FOR J=VARPTR(P
LM2)+256+TY TO VARPTR(PLM2)+256+CS+TY:
POKE J,0:NEXT:RETURN
```

When the Hunter Robot touches the Crown, the mission is accomplished (a win) and the program comes here:

```
1830 PRINT#6,"[esc ctrl clear]":POKE C
HBAS,224:PRINT#6,AT(2,3);"congratulati
ons":PRINT#6,AT(9,4)"a"
1840 PRINT#6,AT(1,5);"[inverse]SUCCESS
FUL MISSION"
1850 FOR J=15 TO 0 STEP -0.5:FOR J1=10
 TO 19:SOUND 0,J1,2,J:SOUND 1,J1*5,10,
J:NEXT J1:NEXT J
1860 FOR J=1 TO 500:NEXT:GOSUB 710
1870 PRINT#6,AT(0,10);"PRESS start  TO
 play":IF PEEK(53279)=6 THEN GOSUB 190
:RUN
1880 FOR J=1 TO 250:NEXT:PRINT#6,AT(6,
10);"[inverse]option[inverse off] TO [
inverse]quit":IF PEEK(53279)=3 THEN EN
D
1890 FOR J=1 TO 250:NEXT:GOTO 1870
```

If the Evil Robot is in the room, move it.

```
200 IF X>RX THEN RX=RX+1
210 IF X<RX THEN RX=RX-1
220 IF Y<RY THEN MOVE VARPTR(PLM2)+384
+RY,VARPTR(PLM2)+384+(RY-1),10:RY=RY-1
230 IF Y>RY THEN MOVE VARPTR(PLM2)+384
+(RY-1),VARPTR(PLM2)+384+RY,10:RY=RY+1
240 POKE &D002,RX:SOUND 1,200,2,4,1:RE
TURN
```

If the Hunter Robot gets "zapped," change its shape.

```
1650 RESTORE 1720:SETCOLOR 0,4,12:FOR
J=VARPTR(PLM2)+128+Y TO VARPTR(PLM2)+1
37+Y:READ CH:POKE J,CH:NEXT
1720 DATA 0,16,0,16,0,40,124,0,56,0
```

Then change its color and create sound effects.

```
1660 FOR CL=12 TO 0 STEP -1:SETCOLOR 0
,4,CL:SOUND 0,20*CL,6,8:FOR T=1 TO 10:
NEXT T:NEXT CL:SOUND 0,0,0,0
```

Clear player-missile graphics images of the Hunter Robot and the other robot.

```
1670 FOR J=VARPTR(PLM2)+128+Y TO VARPT
R(PLM2)+137+Y:POKE J,0:NEXT:FOR TI=1 T
O 200:NEXT
1680 FOR J=VARPTR(PLM2)+384+RY TO VARP
TR(PLM2)+392+RY:POKE J,0:NEXT:POKE &DO
1D,0
```

Tell player he was "zapped"

```
1690 GRAPHICS 2+16:PRINT#6,AT(2,4);"YOU
GOT ZAPPED!!"
```

and create sound effects.

```
1700 FOR J=15 TO 0 STEP -0.5:SOUND 0,6
0,12,J:SOUND 1,120,12,J:SOUND 2,60,12,
J:SOUND 3,120,12,J:NEXT
1710 GOTO 1870
```

In the routine above, the program is sent to line 1870 which gives the player an opportunity to play again.

Subroutines to clear out shape table when Hunter Robot leaves room:

Hunter Robot shape (only necessary when leaving room from top or bottom):

```
190 FOR J=VARPTR(PLM2)+128+Y TO VARPTR
(PLM2)+137+Y:POKE J,0:NEXT:POKE &D01D,
0:RETURN
```

This routine clears other player-missile shapes when Hunter Robot leaves a room.

```
250 FOR J=VARPTR(PLM2)+384+RY TO VARPT
R(PLM2)+392+RY:POKE J,0:NEXT:MON=0
260 IF ROOM=KEY THEN FOR J=VARPTR(PLM2
)+256+TY TO VARPTR(PLM2)+260+TY:POKE J
,0:NEXT
270 IF ROOM=CROWN AND KEY=0 THEN FOR J
=VARPTR(PLM2)+256+TY TO VARPTR(PLM2)+2
65+TY:POKE J,0:NEXT
280 RETURN
```

Subroutine to change variable ROOM when Hunter Robot reaches limits of screen area:

```
720 IF ROOM=16 THEN ROOM=1:GOTO 800
730 ROOM=ROOM+1:GOTO 800
740 IF ROOM=1 THEN ROOM=16:GOTO 800
```

```
750 ROOM=ROOM-1:GOTO 800
760 IF ROOM <5 THEN ROOM=ROOM+12:GOTO
800
770 ROOM=ROOM-4:GOTO 800
780 IF ROOM>12 THEN ROOM=ROOM-12:GOTO
800
790 ROOM=ROOM+4:GOTO 800
```

Reset screen color. Find X,Y location for next Evil Robot.

```
800 POKE 77,0:RX=RND(140)+50:RY=RND(70
)+40
```

Jump to random Monster.

```
810 MONTYPE=RND(5):ON MONTYPE GOSUB 17
30,1740,1750,1760,1780
```

Subroutines to create different Evil Robots (color, shape, and size):

```
1730 SETCOLOR 2,6,8:RESTORE 1790:POKE
&D00A,1:GOTO 1770
1740 SETCOLOR 2,4,8:RESTORE 1800:POKE
&D00A,1:GOTO 1770
1750 SETCOLOR 2,3,8:RESTORE 1810:POKE
&D00A,0:GOTO 1770
1760 SETCOLOR 2,5,8:RESTORE 1820:POKE
&D00A,3
```

Actual routine to create Robot (chosen from above by RESTORE function, which restores a particular line of data and then jumps here to read in shape of Robot:

```
1770 POKE &D002,RX:FOR J=VARPTR(PLM2)+
384+RY TO VARPTR(PLM2)+392+RY:READ CH:
POKE J,CH:NEXT:MON=1
1780 RETURN
1790 DATA 0,66,66,90,255,219,255,231,0
1800 DATA 0,60,126,90,255,255,66,231,0
```

```
1810 DATA 0,60,126,90,255,255,195,0,0
1820 DATA 0,60,36,60,24,255,60,102,0
```

This routine returns here to see if the shape of the Key or the Crown is needed:

```
820 IF KEY=ROOM THEN GOSUB 1080
830 IF CROWN=ROOM AND KEY=0 THEN GOSUB
 1110
```

If so, create the shape:

The Key

```
1080 TX=120:TY=70:CS=4:RESTORE 1100:PO
KE &D001,TX:POKE &D009,1:SETCOLOR 1,12
,8
1090 FOR J=VARPTR(PLM2)+256+TY TO VARP
TR(PLM2)+260+TY:READ CH:POKE J,CH:NEXT
:RETURN
1100 DATA 0,255,165,229,0
```

The Crown

```
1110 TX=114:TY=70:CS=9:RESTORE 1130:PO
KE &D001,TX:POKE &D009,3:SETCOLOR 1,1,
8
1120 FOR J=VARPTR(PLM2)+256+TY TO VARP
TR(PLM2)+265+TY:READ CH:POKE J,CH:NEXT
:RETURN
1130 DATA 0,16,56,16,254,170,214,170,2
54,0
```

The program is now complete. Be sure to save a copy to tape with the CSAVE command or to disk with SAVE "D:filename."

A GRAPHIC ADVENTURE
IN ATARI BASIC:

TIME CRIME

Creating graphic displays with Atari BASIC has been made easy for the user by including many keywords in the language. Three of these are COLOR, PLOT, and DRAWTO. With these keywords the user can draw pictures in graphic modes 3 through 11. When using these modes, the screen is considered to be made of dots or blocks called pixels. The higher the graphic mode, the smaller the pixel. When a color has been assigned and the X,Y coordinate has been plotted, the pixel is lighted with that color. By using DRAWTO, a complete line of pixels can be lighted. Some really beautiful pictures can be drawn in the higher resolution modes (7-11).

Time Crime illustrates some of the ways in which Atari BASIC can be used to create graphics for adventures, as well as for other uses. The program requires 16K when used with cassette or 24K with disk.

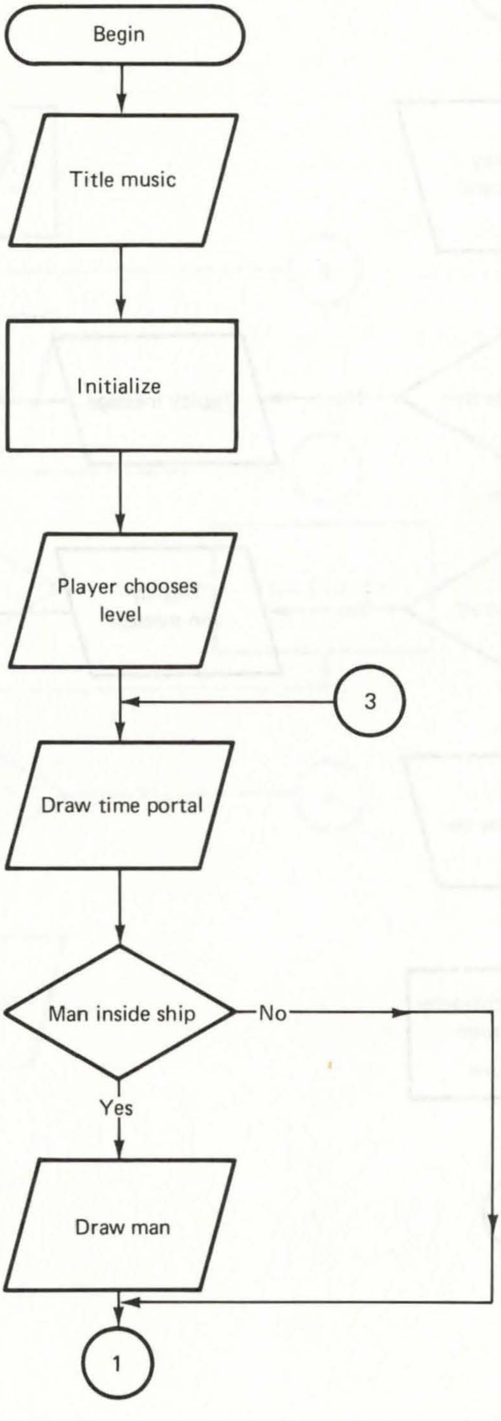
THE OBJECTIVE

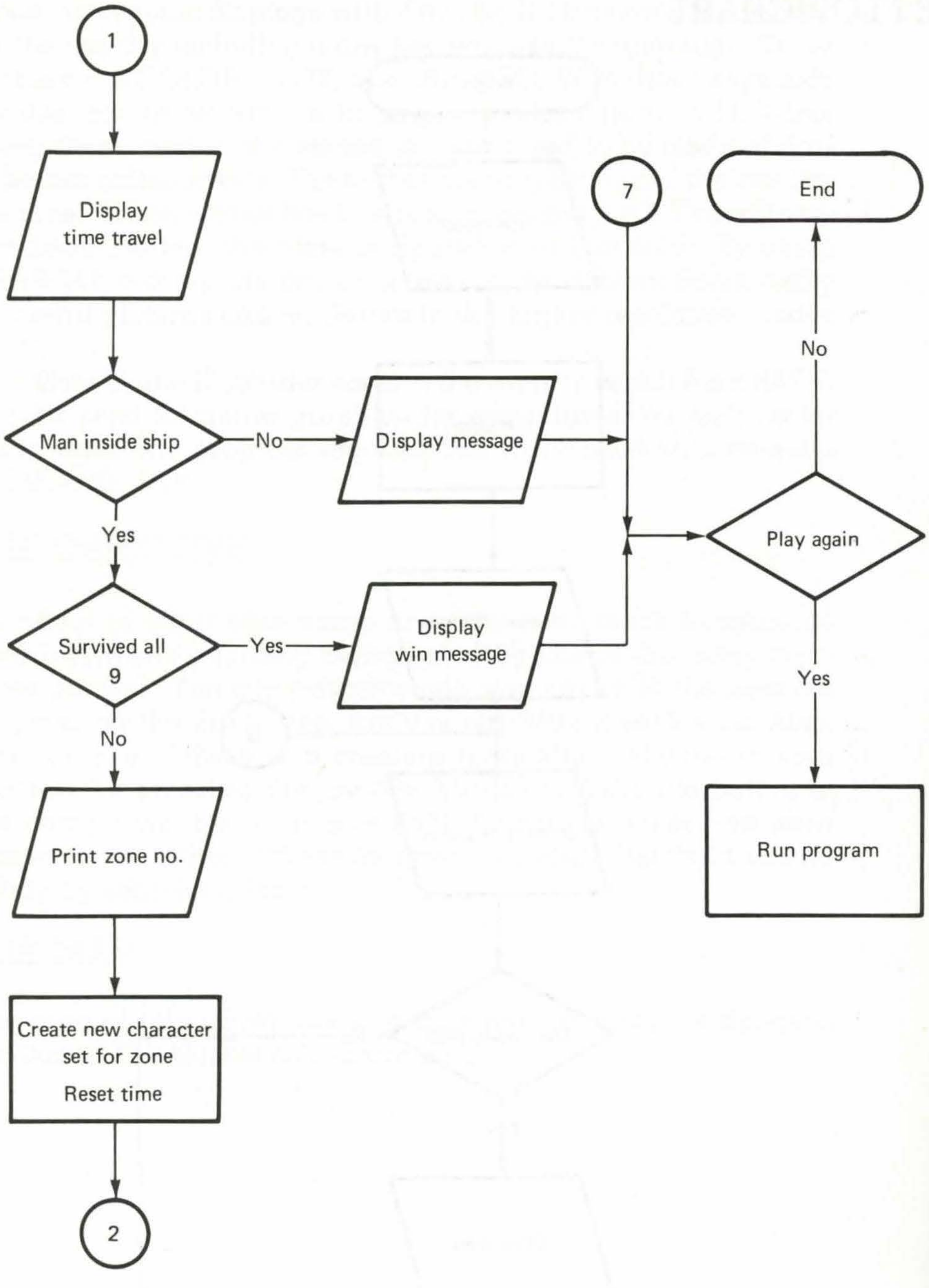
To travel to other time zones and collect as much treasure as possible, while avoiding being eaten by one of the many creatures present. You can advance one step closer to the treasure by pressing the fire button, but you pay $50 for each step. Also, you can see if there is a creature (normally invisible) in your location by pressing the joystick handle and the fire button at the same time, for a price of $200. Of course, since you need money to use either of these features, you must first find treasure solely by searching for it.

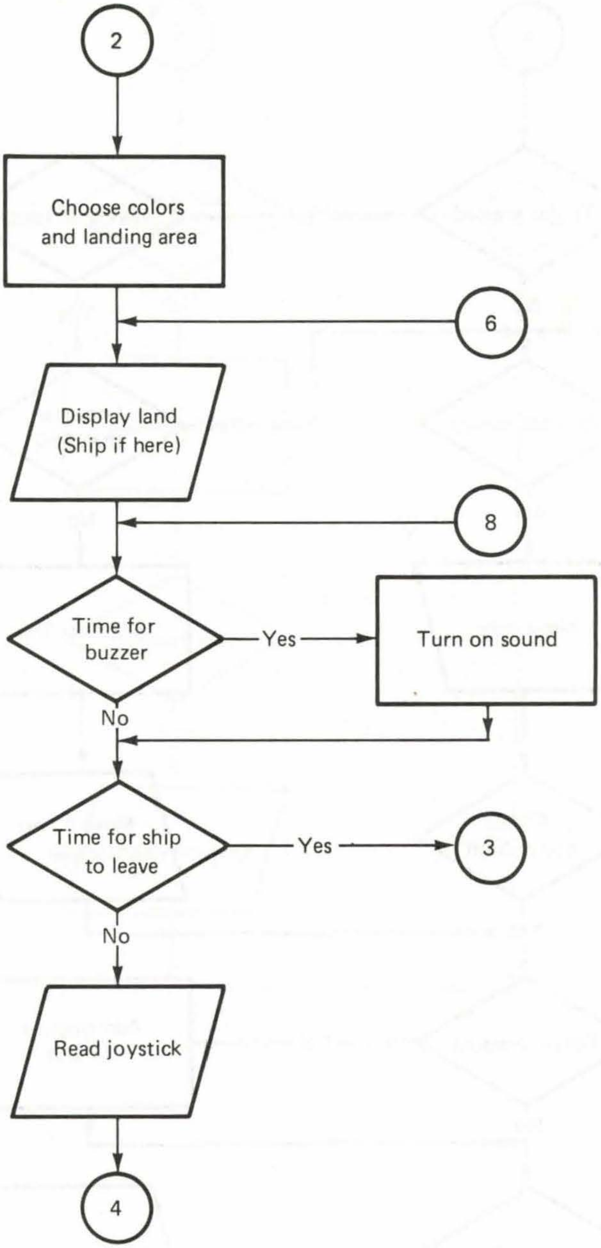
THE MAP

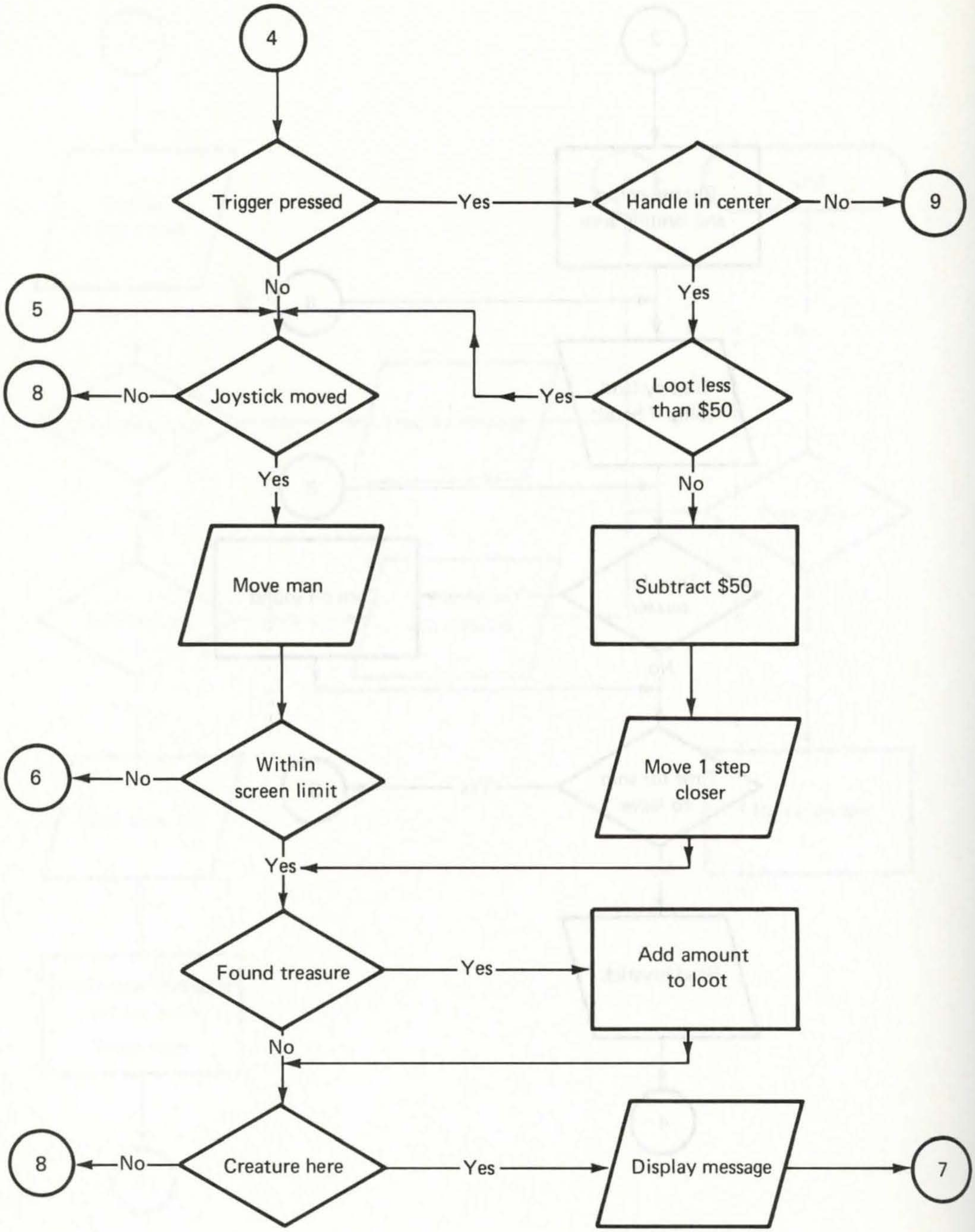
The map of this adventure is a nine-room continuous dungeon similar to **The Hunter** adventure map.

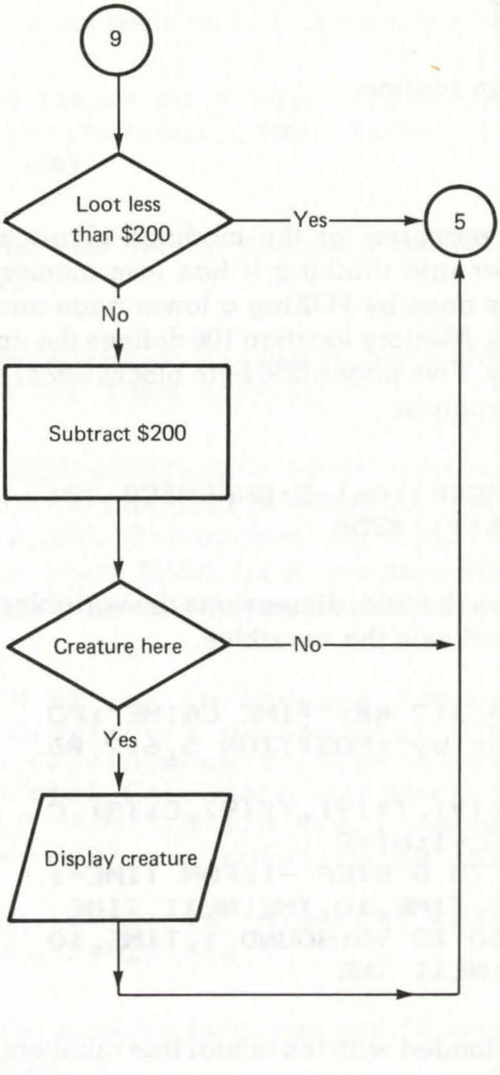
THE FLOWCHART




1
Display time travel
Man inside ship
No
Display message
Yes
Survived all 9
Yes
Display win message
No
Print zone no.
Create new character set for zone
Reset time
2
7
End
No
Play again
Yes
Run program


2
Choose colors and landing area
6
Display land (Ship if here)
8
Time for buzzer
Yes
Turn on sound
No
Time for ship to leave
Yes
3
No
Read joystick
4


4
Trigger pressed
Yes
Handle in center
No
9
No
Yes
5
8
No
Joystick moved
Yes
Loot less than $50
Yes
No
Move man
Subtract $50
Move 1 step closer
6
No
Within screen limit
Yes
Found treasure
Yes
Add amount to loot
No
8
No
Creature here
Yes
Display message
7


9
Loot less than $200
Yes
5
No
Subtract $200
Creature here
No
Yes
Display creature

THE PROGRAM

Jump to initialization routine.

```
10 GOSUB 3000
```

First, the program prepares for the modified character set by fooling the computer into thinking it has less memory than it actually has. This is done by POKing a lower page number into memory location 106. Memory location 106 defines the upper limit of available memory. Five pages (256 byte blocks each) are being subtracted by the program.

```
3000 POKE 106,PEEK(106)-5:GRAPHICS 18:
START=(PEEK(106)+1)*256
```

The program displays the title, dimensions the variables, creates the sound effects, and sets the variables.

```
3010 POSITION 5,3:? #6;"TIME CRIME":PO
SITION 9,5:? #6;"by":POSITION 5,6:? #6
;"JACK HARDY"
3020 DIM L(9),Z(9),TX(9),TY(9),CX(9),C
Y(9):SET=0:LEVEL=1:HI=0
3030 FOR TME=9 TO 0 STEP -1:FOR TIME=1
0 TO 40:SOUND 0,TIME,10,TME:NEXT TIME
3040 FOR TIME=60 TO 90:SOUND 1,TIME,10
,TME:NEXT TIME:NEXT TME
```

These variables are loaded with the actual line numbers for Land [L(n)] and Zone [Z(n)] routines in the program.

```
3060 L(1)=800:L(2)=840:L(3)=850:L(4)=8
70:L(5)=890:L(6)=900:L(7)=930:L(8)=950
:L(9)=955
3070 Z(1)=2060:Z(2)=2090:Z(3)=2130:Z(4
)=2160:Z(5)=2200:Z(6)=2240:Z(7)=2280:Z
(8)=2320:Z(9)=2380
```

Shuffle the Zone routines so that the game is always different.

```
3080 FOR SHU=9 TO 2 STEP -1:NW=INT(RND
(0)*SHU)+1:T=Z(NW):Z(NW)=Z(SHU):Z(SHU)
=T:NEXT SHU
```

Set Variables for man location (MX and MY) and ship location (SX and SY).

```
3090 ZONE=0:MX=9:MY=6:SX=MX:SY=MY:LOOT
=0:IF SET=1 THEN 3180
```

Add the machine language routine to move the character set from ROM into RAM (the area prepared for it with line 3000). This routine is exactly 38 characters long. If you have difficulty keying in the line, check Appendix A, program #6, line 3110 and you will be able to see the line as it should appear in the program.

```
3099 REM ** X$="[h/inverse )/@/ctrl E/
K/ctrl E/M/)/ctrl period/ctrl E/N/%/in
verse off/j/inverse X/inverse off/i/in
verse A/ctrl E/l/space/@/1/M/ctrl Q/K/
H/P/y/f/L/f/N/%/N/I/d/P/m/inverse off/
ctrl period]" - a total of 38 characte
rs.
3110 DIM X$(38):X$="h)@EKEM).EN%jXiAEL
 @1MQKHPyfLfN%NIdPm."
```

Jump to the machine language routine stored in X$ and move the character set.

```
3120 Z=USR(ADR(X$)):RESTORE 3150
```

Change the ! character to a man figure and the] character to a time machine.

```
3130 FOR TIME=1 TO 2:READ X
3140 FOR Y=0 TO 7:READ Z:POKE X+Y+STAR
```

```
T,Z:NEXT Y:NEXT TIME:SET=1
3150 DATA 8,56,56,16,124,124,56,40,40
3160 DATA 472,56,56,16,56,108,254,186,
130
```

Let player choose level of difficulty and when to start game.

```
3180 POSITION 3,9:? #6;"[inverse]SELEC
T[inverse off] LEVEL - ";LEVEL:POSITIO
N 4,10:? #6;"[inverse]start[inverse of
f] TO PLAY"
3185 IF PEEK(53279)=6 THEN 3200
3190 IF PEEK(53279)=5 THEN LEVEL=LEVEL
+1:SOUND 0,30,10,4:IF LEVEL>3 THEN LEV
EL=1
3192 MAXMIN=4-LEVEL:POKE 710,INT(RND(0
)*16)*16+8:POKE 711,INT(RND(0)*16)*16+
8
3195 FOR TM=1 TO 30:NEXT TM:SOUND 0,0,
0,0:GOTO 3180
3200 GOSUB 1010:RETURN
```

When start is pressed, change graphics mode to 5+16 (no text window, Graphics 21), pick a random color from 0 to 15, and set the color registers 0 through 2. (*Note:* One register is set darker than the rest.)

```
1010 GRAPHICS 21:C=INT(RND(0)*16):SETC
OLOR 0,C,8:SETCOLOR 1,C,8:SETCOLOR 2,C
,4:COL=1
```

Start with color register 1 and draw squares in ever-decreasing increments. The PLOT and DRAWTO allow you to plot points and draw lines in the graphic modes 3, 4, 5, 6, 7, 8 (also 9, 10, and 11 if you have the new GITA chip). The PLOT statement illuminates a single point on the screen. DRAWTO indicates the column and row to draw to and must be used with the PLOT statement. PLOT must also be preceded by a COLOR(n) statement to inform it as to which color register to use; otherwise it will PLOT and DRAWTO in the background color of black.

```
1020 FOR I=0 TO 20:COLOR COL
1030 X=I*2:Y=I:SOUND 1,I*5,10,8
1040 PLOT X,Y
1050 DRAWTO 79-X,Y:DRAWTO 79-X,47-Y:PL
OT 79-X-1,Y
1060 DRAWTO 79-X-1,47-Y:DRAWTO X,47-Y
1070 DRAWTO X,Y:PLOT X+1,47-Y:DRAWTO X
+1,Y
```

Shift color register every other time around for darker color in some squares.

```
1080 COL=COL+0.5:IF COL+0.5>=4 THEN CO
L=COL-3
1090 NEXT I:IF (MX<>SX OR MY<>SY) THEN
 2000
```

Note: The above IF/THEN statement makes sure the man made it back to the ship location. If he didn't make it the program skips the part below which draws the man inside the ship.

Draw a man inside the time machine.

```
1100 COLOR 0:PLOT 44,21:DRAWTO 36,21:P
LOT 39,19:DRAWTO 39,17:DRAWTO 41,17:DR
AWTO 41,19:PLOT 40,18:DRAWTO 40,23
1110 PLOT 39,22:DRAWTO 39,23:DRAWTO 37
,25:PLOT 41,22:DRAWTO 41,23:DRAWTO 43,25
```

Transfer the colors from one register to another and create sound for effect of travel through time.

```
2000 FOR TME=1 TO 50:TEMP=PEEK(708):PO
KE 708,PEEK(709):POKE 709,PEEK(710):PO
KE 710,TEMP
2010 SOUND 0,TEMP*15,10,4:SOUND 1,PEEK
(708),10,4:FOR TIME=1 TO 10:NEXT TIME:
NEXT TME
```

Turn off sound and check to see if the man is in the ship again.

```
2020 SOUND 0,0,0,0:SOUND 1,0,0,0:GRAPH
ICS 18:IF (MX<>SX OR MY<>SY) THEN 500
```

If the man didn't make it back to ship:

```
500 POSITION 2,4:? #6;"YOUR TIME MACHI
NE":? #6;"  LEFT WITHOUT YOU!"
510 POSITION 4,6:? #6;"LOOT $";LOOT
520 IF LOOT>HI THEN HI=LOOT
530 POSITION 2,7:? #6;"MOST LOOT $";HI
550 POSITION 3,9:? #6;"[inverse]PLAY A
GAIN? Y/N"
560 OPEN #1,4,0,"K:":GET #1,K:CLOSE #1
570 IF K<>78 AND K<>89 THEN 560
580 IF K=78 THEN POSITION 1,10:? #6;"T
HANKS FOR PLAYING":FOR TM=1 TO 500:NEX
T TM:END
590 GOSUB 3080:GOTO 20
```

If the man was in the ship the program increments to next time zone and checks to see if he's survived all 9 time zones.

```
2025 ZONE=ZONE+1:IF ZONE=10 THEN 4000
```

The man made it through all nine zones.

```
4000 POSITION 2,1:? #6;"[inverse]CONGR
ATULATIONS!":POSITION 2,3:? #6;"you pi
lliaged all"
4010 POSITION 3,4:? #6;"nine time zone
s";:PUT #6,1:POSITION 1,6:? #6;"LOOT T
AKEN $";LOOT
4020 IF LOOT>HI THEN HI=LOOT
4030 POSITION 2,7:? #6;"MOST LOOT $";H
I
4040 FOR TIME=15 TO 0 STEP -1:FOR TME=
10 TO 15:SOUND 0,TME,10,TIME:SOUND 1,T
ME*5,10,TIME:NEXT TME:NEXT TIME
4050 GOTO 550
```

If there are still more zones left, the program informs the player.

```
2027 POSITION 4,4:? #6;"now entering":
POSITION 4,5:? #6;"TIME ZONE #";ZONE
```

Shuffle the land line numbers from line 3060 so layout of the land will not be in the same order each time the ship lands.

```
2030 FOR SHU=9 TO 2 STEP -1:NL=INT(RND
(0)*SHU)+1:T=L(NL):L(NL)=L(SHU):L(SHU)
=T:NEXT SHU
```

Reset the creature and treasure locations.

```
2032 FOR CRE=1 TO 9:CY(CRE)=0:CX(CRE)=
0:NEXT CRE
2035 FOR TRE=1 TO 9:TX(TRE)=INT(RND(0)
*13)+3:TY(TRE)=INT(RND(0)*6)+4:NEXT TR
E
```

The program creates more creatures as the player makes it through each time zone.

```
2036 FOR CRE=1 TO ZONE:CX(CRE)=INT(RND
(0)*13)+3:CY(CRE)=INT(RND(0)*6)+4:NEXT
 CRE
2040 ON ZONE GOSUB Z(1),Z(2),Z(3),Z(4)
,Z(5),Z(6),Z(7),Z(8),Z(9)
```

Note: Above are the shuffled zone line numbers from line 3080.

The shapes of trees, rocks, walls, and creatures are created below. Each is a different shape in each of the nine time zones the player will enter.

Note: These are being changed by the program as the player is reading the message ENTERING TIME ZONE n.

```
2060 RESTORE 2081:FOR TIME=1 TO 7:READ
 X
2070 FOR Y=0 TO 7:READ Z:POKE X+Y+STAR
T,Z:NEXT Y:NEXT TIME:RETURN
2081 DATA 24,239,239,239,0,254,254,254
,0
2082 DATA 40,60,126,255,126,60,24,24,2
4
2083 DATA 48,24,24,24,24,24,24,24,60
2084 DATA 56,255,219,129,129,129,129,1
65,255
2085 DATA 64,16,40,68,130,8,20,34,65
2086 DATA 72,16,8,4,2,1,255,255,255
2087 DATA 256,0,0,255,255,255,255,0,0
2090 RESTORE 2111:FOR TIME=1 TO 7:READ
 X
2100 FOR Y=0 TO 7:READ Z:POKE X+Y+STAR
T,Z:NEXT Y:NEXT TIME:RETURN
2111 DATA 24,224,238,238,238,14,238,23
8,224
2112 DATA 40,189,90,189,24,24,24,24,24
2113 DATA 48,24,24,24,24,24,60,66,129
2114 DATA 56,170,255,213,129,129,171,2
55,85
2115 DATA 64,170,68,170,0,170,68,170,0
2116 DATA 72,0,0,60,126,126,126,60,0
2117 DATA 256,0,0,170,255,255,85,0,0
2130 RESTORE 2151:FOR TIME=1 TO 7:READ
 X
2140 FOR Y=0 TO 7:READ Z:POKE X+Y+STAR
T,Z:NEXT Y:NEXT TIME:RETURN
2151 DATA 24,255,255,255,255,255,255,2
55,255
2152 DATA 40,24,60,126,255,255,126,60,
24
2153 DATA 48,24,24,24,24,24,255,60,255
2154 DATA 56,24,36,66,129,129,66,36,24
2155 DATA 64,130,68,40,16,65,34,20,8
2156 DATA 72,0,32,112,36,14,68,224,64
2157 DATA 256,0,0,0,255,255,0,0,0
2160 RESTORE 2181:FOR TIME=1 TO 7:READ
 X
2170 FOR Y=0 TO 7:READ Z:POKE X+Y+STAR
```

```
T,Z:NEXT Y:NEXT TIME:RETURN
2181 DATA 24,102,255,255,126,126,255,2
55,102
2182 DATA 40,60,255,126,24,126,255,60,
24
2183 DATA 48,60,255,126,24,126,255,60,
24
2184 DATA 56,255,255,129,129,129,129,2
55,255
2185 DATA 64,56,238,0,28,119,0,56,238
2186 DATA 72,112,56,28,0,28,56,112,224
2187 DATA 256,0,0,0,255,255,0,0,0
2200 RESTORE 2221:FOR TIME=1 TO 7:READ
 X
2210 FOR Y=0 TO 7:READ Z:POKE X+Y+STAR
T,Z:NEXT Y:NEXT TIME:RETURN
2221 DATA 24,170,85,170,85,170,85,170,
85
2222 DATA 40,126,255,90,102,90,102,255
,24
2223 DATA 48,24,90,189,90,24,24,24,24
2224 DATA 56,60,102,195,129,129,195,10
2,60
2225 DATA 64,137,157,137,195,195,145,1
85,145
2226 DATA 72,0,120,120,0,60,60,0,0
2227 DATA 256,0,0,0,255,255,0,0,0
2240 RESTORE 2261:FOR TIME=1 TO 7:READ
 X
2250 FOR Y=0 TO 7:READ Z:POKE X+Y+STAR
T,Z:NEXT Y:NEXT TIME:RETURN
2261 DATA 24,170,170,170,170,170,170,1
70,170
2262 DATA 40,129,66,60,153,90,60,24,24
2263 DATA 48,24,60,90,153,24,60,66,129
2264 DATA 56,255,129,231,129,231,129,1
29,255
2265 DATA 64,73,0,0,146,0,0,73,0
2266 DATA 72,153,62,96,60,6,124,24,129
2267 DATA 256,255,129,129,231,129,231,
129,255
2280 RESTORE 2301:FOR TIME=1 TO 7:READ
 X
```

```
2290 FOR Y=0 TO 7:READ Z:POKE X+Y+STAR
T,Z:NEXT Y:NEXT TIME:RETURN
2301 DATA 24,255,0,255,0,255,0,255,0
2302 DATA 40,126,255,255,255,255,255,1
26,60
2303 DATA 48,60,60,60,60,60,60,255,255
2304 DATA 56,255,129,135,225,135,225,1
29,255
2305 DATA 64,198,198,0,24,24,0,198,198
2306 DATA 72,0,16,56,124,56,16,0,0
2307 DATA 256,255,129,225,135,225,135,
129,255
2320 RESTORE 2331:FOR TIME=1 TO 7:READ
 X
2330 FOR Y=0 TO 7:READ Z:POKE X+Y+STAR
T,Z:NEXT Y:NEXT TIME:RETURN
2331 DATA 24,85,85,85,85,85,85,85,85
2332 DATA 40,192,224,240,120,28,14,7,7
2333 DATA 48,7,7,7,7,15,30,62,60
2334 DATA 56,102,60,60,255,60,100,4,6
2335 DATA 64,68,146,41,68,0,36,82,137
2336 DATA 72,6,6,0,48,48,0,192,192
2337 DATA 256,102,60,60,255,60,38,32,9
6
2380 RESTORE 2391:FOR TIME=1 TO 7:READ
 X
2390 FOR Y=0 TO 7:READ Z:POKE X+Y+STAR
T,Z:NEXT Y:NEXT TIME:RETURN
2391 DATA 24,219,219,0,219,219,0,219,2
19
2392 DATA 40,63,126,252,248,240,224,96
,96
2393 DATA 48,96,96,96,96,96,96,96,96
2394 DATA 56,171,171,255,255,255,255,2
13,213
2395 DATA 64,16,40,2,37,80,4,74,160
2396 DATA 72,0,24,36,66,129,255,255,25
5
2397 DATA 256,213,213,255,255,255,255,
171,171
```

The program returns here to pick random colors for the display of each time zone.

```
2045 POKE 708,INT(RND(0)*16)*16+8:POKE
 709,INT(RND(0)*16)*16+8:POKE 710,INT(
RND(0)*16)*16+8
2046 POKE 711,INT(RND(0)*16)*16+8
```

The program picks a random ship location, sets a background color, sets the timing registers to zero, lets the computer know where to find the new character set, and returns to main loop.

```
2050 LOC=INT(RND(0)*9)+1:POKE 712,INT(
RND(0)*16)*16+12:POKE 20,0:POKE 19,0:S
HIP=LOC:POKE 756,START/256:RETURN
```

The main program loop:

First, clear screen and display loot.

```
20 ? #6;"[esc ctrl clear]":POSITION 1,
0:? #6;"$";LOOT
```

Then jump to the correct routines (shuffled in line 2030) to draw the landscape.

```
30 ON LOC GOSUB L(1),L(2),L(3),L(4),L(
5),L(6),L(7),L(8),L(9)
```

Note: The program uses a PUT statement instead of a PRINT statement. This allows the man character to be displayed in a usually unavailable color. *Example:* In direct mode type GRAPHICS 2:PRINT #6;"![inverse]!" hit RETURN. Now displayed are two colored exclamation marks. Since there is no way to display the exclamation marks other than normal or inverse, there are only two colors available to the program. Hit SYSTEM RESET and again in direct mode type GRAPHICS 2:PUT #6,1:PUT #6,33:PUT #6,129:PUT #6,161. There are now four colored quotation marks. For other character values see Appendix B.

```
40 POSITION MX,MY:PUT #6,1
```

If ship variable equals (SHIP) location variable (LOC), a PUT statement places it on the screen.

```
50 IF LOC=SHIP THEN POSITION SX,SY:PUT
 #6,251
```

Set time variable and reset color shift register.

```
60 TM=PEEK(19)/14:POKE 77,0
```

Print the amount of time remaining.

```
70 POSITION 14,0:? #6;MAXMIN-INT(TM);"
 MIN."
```

If time has expired, time machine leaves.

```
80 IF MAXMIN=TM THEN GOSUB 1000:GOTO 2
0
```

Sound a 30-second warning before time machine leaves.

```
90 IF TM=(MAXMIN-0.5) THEN SOUND 1,0,4
,1
```

Program checks for joystick trigger being pressed

```
100 IF STRIG(0)=0 THEN GOSUB 210
```

then checks to see if player wants to move (which way is joystick being pressed?).

```
110 S=STICK(0):NX=(S=5 OR S=6 OR S=7)-
(S=9 OR S=10 OR S=11)
120 NY=(S=5 OR S=9 OR S=13)-(S=6 OR S=
10 OR S=14):IF  NOT (NX OR NY) THEN 60
125 OX=MX:OY=MY:MX=MX+NX:MY=MY+NY
```

If player's man is at the limits of the screen, the program changes locations.

```
130 IF MX>19 THEN MX=0:GOTO 300
140 IF MX<0 THEN MX=19:GOTO 320
150 IF MY>11 THEN MY=1:GOTO 350
160 IF MY<1 THEN MY=11:GOTO 370
```

The program checks to see if player's man is at the ship location

```
170 LOCATE MX,MY,Z:IF Z=251 THEN POSIT
ION OX,OY:? #6;" ";:GOSUB 1000:GOTO 60
```

then it checks to see if player's man is trying to go through a tree, rock, or wall.

```
175 IF Z<>32 THEN MX=OX:MY=OY:GOTO 60
```

Next it checks to see if player's man is on the same space with the treasure.

```
180 IF MX=TX(LOC) AND MY=TY(LOC) THEN
GOSUB 250
185 POSITION OX,OY:? #6;" ";:POSITION
MX,MY:PUT #6,1:SOUND 0,150,6,4
```

Finally, it checks to see if player's man is on the space with a creature.

```
190 SOUND 0,0,0,0:IF CX(LOC)=MX AND CY
(LOC)=MY THEN 460
200 GOTO 40
```

These subroutines display each location accessed from line 30:

```
800 POSITION 2,2:? #6;"[inverse]### ##
# ### ###":FOR BY=3 TO 5:POSITION 2,BY
:? #6;"[inverse]#     # #     #":NEXT BY
```

```
810 POSITION 2,6:? #6;"[inverse]######
# #######":FOR BY=7 TO 9:POSITION 2,BY
:? #6;"[inverse]#     # #     #":NEXT
BY
820 POSITION 2,10:? #6;"[inverse]### #
## ### ###":RETURN
840 POSITION 2,2:? #6;"[inverse]######
#   ######":GOSUB 970:GOSUB 980:GOSUB
990:RETURN
850 FOR BY=3 TO 4:POSITION 2,BY:? #6;"
[inverse]#":NEXT BY:FOR BY=8 TO 9:POSI
TION 2,BY:? #6;"[inverse]#":NEXT BY
860 GOSUB 960:GOSUB 970:GOSUB 980:RETU
RN
870 FOR BY=3 TO 4:POSITION 17,BY:? #6;
"[inverse]#":NEXT BY:FOR BY=8 TO 9:POS
ITION 17,BY:? #6;"[inverse]#":NEXT BY
880 GOSUB 960:GOSUB 980:GOSUB 990:RETU
RN
890 POSITION 2,10:? #6;"[inverse]#####
#   #######":GOSUB 960:GOSUB 970:GOSUB
 990:RETURN
900 POSITION 2,2:? #6;"[inverse]######
   #######":FOR BY=3 TO 5:POSITION 7,B
Y:? #6;"[inverse]#   #":NEXT BY
910 FOR BY=7 TO 9:POSITION 7,BY:? #6;"
[inverse]#   #":NEXT BY:POSITION 2,10:
? #6;"[inverse]######   #######"
920 GOSUB 970:GOSUB 990:RETURN
930 FOR TREE=1 TO 10:TX=INT(RND(0)*18)
+1:TY=INT((RND(0)*4)+2)*2:POSITION TX,
TY:PUT #6,134
940 POSITION TX,TY-1:PUT #6,37:NEXT TR
EE:RETURN
950 FOR SWAMP=1 TO 10:TX=INT(RND(0)*17
)+1:TY=INT(RND(0)*9)+2:POSITION TX,TY:
? #6;"[inverse]((":NEXT SWAMP:RETURN
955 GOSUB 950:GOSUB 930:RETURN
960 POSITION 2,2:? #6;"[inverse]######
##########":RETURN
970 FOR BY=3 TO 9:POSITION 17,BY:? #6;
"[inverse]#":NEXT BY:RETURN
980 POSITION 2,10:? #6;"[inverse]#####
###########":RETURN
```

```
990 FOR BY=3 TO 9:POSITION 2,BY:? #6;"
[inverse]#":NEXT BY:RETURN
```

When time is depleted, the program comes here from line 80, or if the man returns to the ship, it comes here to check time variable (TM). If time has not expired, the program returns to the main loop.

```
1000 IF MAXMIN<>TM THEN RETURN
```

If joystick trigger is pressed, the program comes here from line 100.

```
210 IF STICK(0)<>15 THEN 600
```

If the player has less than $50 the treasure finder will not work and the program returns to the main loop.

```
215 IF LOOT<50 THEN RETURN
```

Subtract the money from his loot and display the new amount.

```
217 LOOT=LOOT-50:POSITION 1,0:? #6;"$"
;LOOT;" "
```

If the treasure at this location has already been found, the program sounds the buzzer and returns to the main loop.

```
218 IF TY(LOC)=25 THEN SOUND 3,10,2,8:
FOR T=1 TO 20:NEXT T:SOUND 3,0,0,0:GOT
O 245
```

If the player has $50 the program comes here to find the treasure location and move the player one step closer to the treasure.

```
220 OX=MX:OY=MY:OC=PEEK(709):POKE 709,
INT(RND(0)*16)*16+12:SOUND 3,200,10,8
222 IF MX>TX(LOC) THEN MX=MX-1
```

```
225 IF MY>TY(LOC) THEN MY=MY-1
230 IF MX<TX(LOC) THEN MX=MX+1
235 IF MY<TY(LOC) THEN MY=MY+1
240 POKE 709,OC:SOUND 3,0,0,0:POP :GOT
O 130
245 IF STRIG(0)=0 THEN 245
247 RETURN
```

If the joystick handle is moved when the trigger pressed, the program comes here to check for a creature in this location.

If the player has less than $200, the program returns to the main loop.

```
600 IF LOOT<200 THEN RETURN
```

If a creature is present, it is shown to the player.

```
610 SOUND 0,200,10,8:IF CX(LOC) THEN G
OSUB 400
```

Subtract money and display the new amount.

```
630 LOOT=LOOT-200:POSITION 1,0:? #6;"$
";LOOT;"  ":SOUND 0,0,0,0
640 IF STICK(0)<>15 THEN 640
650 RETURN
```

When the player's man leaves the screen boundaries, the program comes here to change locations from lines 222 through 235.

```
300 LOC=LOC+1:IF LOC>9 THEN LOC=1
310 GOTO 20
320 LOC=LOC-1:IF LOC<1 THEN LOC=9
330 GOTO 20
350 LOC=LOC+3:IF LOC>9 THEN LOC=LOC-9
360 GOTO 20
370 LOC=LOC-3:IF LOC<1 THEN LOC=LOC+9
380 GOTO 20
```

If man's location and treasure location are the same, the program comes here from line 180. The program puts the treasure on the screen, makes the appropriate sound, and adds the value of the treasure to the player's loot.

```
250 POSITION TX(LOC),TY(LOC):PUT #6,13
7:FOR TME=1 TO 5:SOUND 0,10,10,6:FOR T
IME=1 TO 5:NEXT TIME
255 SOUND 0,0,0,0:FOR TIME=1 TO 20:NEX
T TIME:NEXT TME
260 TY(LOC)=25:LOOT=LOOT+INT(RND(0)*10
00*ZONE)+100:POSITION 1,0:? #6;"$";LOO
T:RETURN
```

If the player's man location and the creature location are the same, the program comes here from line 190. The program shows the monster chewing, creates sound, and displays the appropriate message.

```
400 FOR TME=1 TO 5:POSITION CX(LOC),CY
(LOC):? #6;"@";:SOUND 2,100,8,4
410 FOR TIME=1 TO 9:NEXT TIME:POSITION
 CX(LOC),CY(LOC):? #6;"'";
420 SOUND 2,0,0,0:FOR TIME=1 TO 9:NEXT
 TIME:NEXT TME:RETURN
460 GOSUB 400:POSITION 1,4:? #6;"A CRE
ATURE GOBBLED         YOU UP":GOTO 520
```

The **Time Crime** program is complete. Be sure to save a copy onto cassette with the CSAVE command, or onto diskette with SAVE "D:filename."

PART 4

Creating Your Own Computer Adventure

Now that you have become familiar with the basic components of a computer adventure, and the various programming capabilities of Atari PILOT, Microsoft BASIC, and BASIC, you are ready to create your own original adventure. In this section you will find **The Creator,** a program used to develop a disk-based adventure, along with some sample data you can enter to become familiar with how the program works, and **The Interpreter,** the program used to play the adventure you create.

Remember that your programming can be as simple or as complex as you wish to make it. Simple, straightforward programs can do the same job or task as one organized into various specialized subroutines. Although the job may take longer and use more memory with a simple program, it will still be accomplished. However, as your programming skill improves, you may wish to take advantage of the advanced techniques which follow. These rules usually apply to BASIC, although some are valid for PILOT; the applicable language is noted for each.

MEMORY SAVINGS

- Whenever possible, place multiple statements within program lines to eliminate line numbers. With each new line number, 6 bytes of memory are required, but when a statement is added to an existing line, only 3 bytes of additional memory are required. This applies to Microsoft BASIC as well as Atari BASIC. PILOT can use multiple statements only in its graphics routines (GR:) to draw pictures.

- In Atari BASIC, replace much-used constants with variables. Each occurrence of a numeric constant in a statement is replaced by a one-byte pointer to the memory location where the value of that constant is stored. This value in memory is stored in internal binary form and occupies an additional 6 bytes regardless of the size of the constant. This means that each occurrence of a constant requires 7 bytes of memory, since Atari BASIC does not check for multiple use of any constant.

When a variable is used it is placed in a table in memory called the Variable Name Table (VNT). Six bytes of memory are allocated to store the value of the variable and an additional two bytes are stored in the VNT to point to the value of its variable. Only one byte is placed in the BASIC statement in place of the variable name. This byte points to the VNT. Therefore, replacement of much-used constants will save six bytes each time the constant is replaced.

In order to be able to remember that the variable is being used as a constant, assign simple numeric variables to the constant in the following manner: U0=0:U10=10:U500=500, etc.

- Use short variable names, because the more characters there are in a variable name, the more memory will be used to store the variable name in the VNT. Since you can have only 128 variables in BASIC, significant memory savings can be achieved by reusing variables to perform the same task (i.e., FOR/NEXT loops).

- Replace IF N<>0 with IF N. In PILOT, the conditional command would be changed from (#N<>0) to (#N). This technique will save 3 bytes for each conversion. The use of an IF/THEN in BASIC or a conditional in PILOT is to check for true or false; if the variable equals zero, the statement is false and will not execute; anything other than zero is considered true and the statement executes.

- Use subroutines to eliminate duplicate statements. When a statement is duplicated many times in a program, it is better to convert it to a subroutine and GOSUB each time the statement is required.
- When jumping out of a FOR/NEXT loop or from within a subroutine, use POP to clear the stack. When a GOSUB or FOR/NEXT is executed, BASIC puts the return address into a computer memory location called the stack. When the RETURN or NEXT command is executed, the top address is popped off the stack and the computer uses it to return control to the program address. Therefore, the stack is constantly changing and, ideally, it is being cleared after each NEXT or RETURN command. If the program jumps out of the FOR/NEXT loop before completion, or the RETURN is never executed, the stack retains the program address in memory. When this occurs, the stack can quickly eat up memory. The POP command clears these unused addresses.

SPEED OF EXECUTION

- Organize your program so that seldom-used subroutines will be at the end. BASIC starts at the beginning of the program and searches downward by line numbers to find the requested GOTO, GOSUB, or FOR/NEXT loop. Placing much-used routines as close to the beginning as possible will greatly speed up your program.

USER-FRIENDLINESS

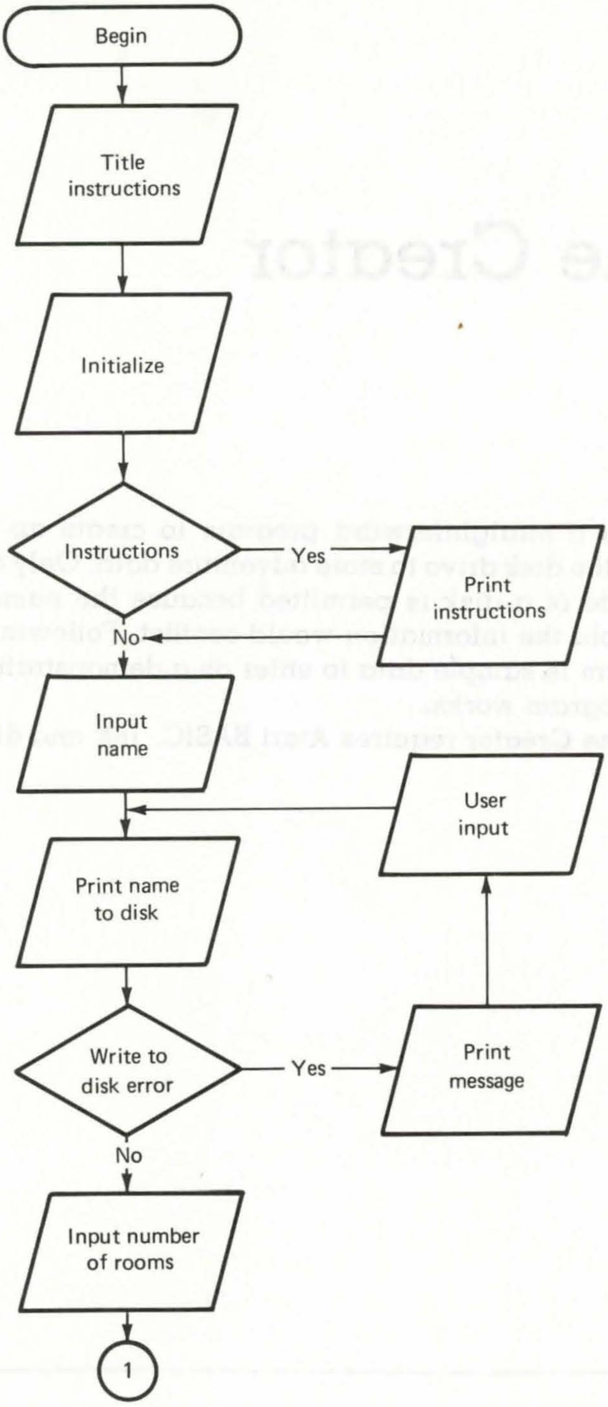
- Always make your program user-friendly by the judicious use of TRAPs so that unwanted input can be ignored. Prompt the user for the proper input and explain the use of the program when necessary.

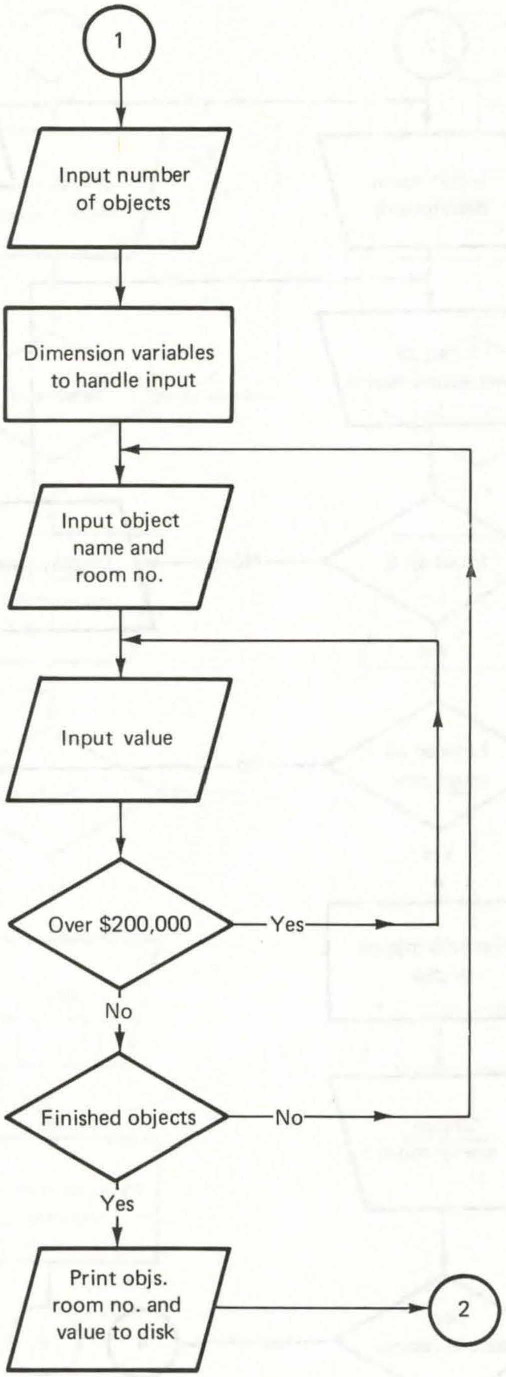
The Creator

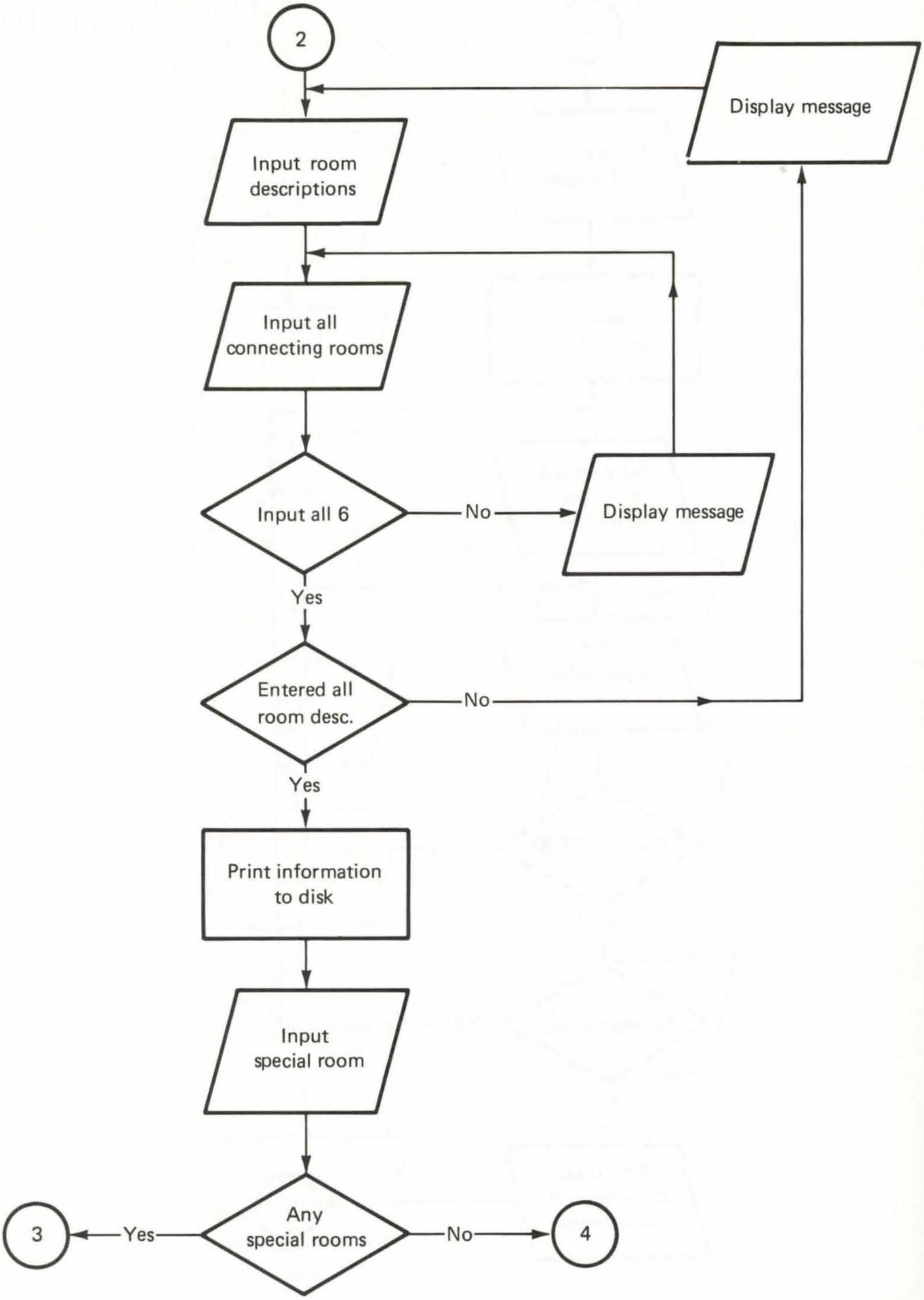
This is a straightforward program to create an adventure by using the disk drive to store adventure data. Only one adventure per side of a disk is permitted because the names of the files that hold the information would conflict. Following **The Creator** program is sample data to enter as a demonstration of the way the program works.

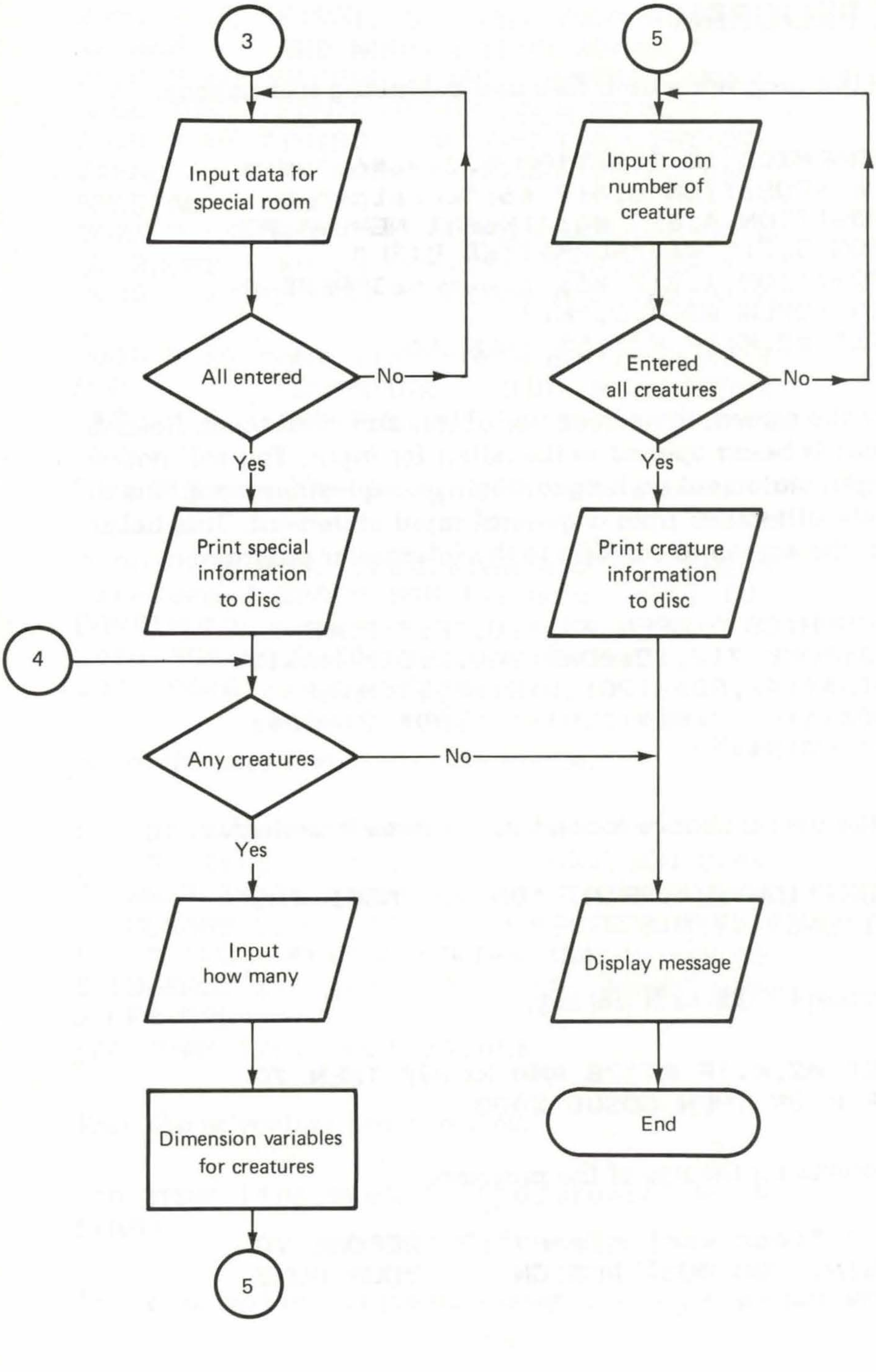
The Creator requires Atari BASIC, 16K and disk drive.

THE FLOWCHART




1
Input number of objects
Dimension variables to handle input
Input object name and room no.
Input value
Over $200,000
Yes
No
Finished objects
No
Yes
Print objs. room no. and value to disk
2


2
Display message
Input room descriptions
Input all connecting rooms
Input all 6
No
Display message
Yes
Entered all room desc.
No
Yes
Print information to disk
Input special room
3
Yes
Any special rooms
No
4


3
Input data for special room
All entered
No
Yes
Print special information to disc
4
Any creatures
No
Yes
Input how many
Dimension variables for creatures
5
5
Input room number of creature
Entered all creatures
No
Yes
Print creature information to disc
Display message
End

THE PROGRAM

Open the program with a title and beginning instructions.

```
10 GRAPHICS 18:POSITION 5,3:? #6;"adve
nture":POSITION 6,4:? #6;"creator"
20 POSITION 4,6:? #6;"INSERT NEWLY":PO
SITION 3,7:? #6;"FORMATTED DISK"
30 POSITION 4,9:? #6;"[inverse]PRESS R
ETURN":OPEN #2,4,0,"K:"
40 GET #2,K:IF K<>155 THEN 40
```

Set up the screen, dimension variables, and clear them. *Note:* A channel is being opened to the editor for input. You will notice the input statement no longer displays a question mark; this is the only difference from a general input statement. This helps format the screen to conform to the interpreter program.

```
50 GRAPHICS 0:OPEN #3,4,0,"E:":POKE 7
10,12:POKE 712,12:POKE 709,2:DIM NA$(2
0),DIS$(14),RD$(120),DIR(6),ITEM(6)
55 RD$(1)=" ":RD$(120)=" ":RD$(2)=RD$:
NA$="":DIS$=""
```

Give the user a chance to read instructions if desired.

```
60 POSITION 2,4:PRINT "DO YOU NEED INS
TRUCTIONS? (Y/N)"
```

Only accept Y (89) or N (78) key.

```
70 GET #2,K:IF K<>78 AND K<>89 THEN 70
80 IF K=89 THEN GOSUB 2000
```

Instructions for the use of the program:

```
2000 ? "[esc ctrl clear]":? "BEFORE YO
U BEGIN, YOU MUST DESIGN      YOUR DUNG
EON ON PAPER."
```

```
2005 ? :? "FIRST, LAY OUT YOUR ADVENTU
RE MAP     AND NUMBER YOUR ROOMS."
2010 ? :? "SECOND, LIST, NUMBER, AND V
ALUE THE   CONTENTS OF EACH ROOM."
2020 ? :? "THIRD, CONSTRUCT A LIST OF
ITEMS      BY NUMBER NEEDED TO ACCESS
SPECIAL    ROOMS."
2025 ? :? "FOURTH, PREPARE A LIST OF A
LL ROOMS   WHICH CONTAIN CREATURES."
2030 ? :? "FINALLY, ENTER ALL THE DATA
."
2040 ? :? "ANY TIME YOU WANT TO PLAY Y
OUR        ADVENTURE, LOAD IN INTERPRE
TER        AND RUN."
```

Turn off cursor. Wait for space bar to be pressed.

```
2050 POKE 752,1:POSITION 4,23:? "PRESS
 [inverse] SPACE BAR [inverse off] TO
CONTINUE";
2060 GET #2,K:IF K<>32 THEN 2060
2070 POKE 752,0:RETURN
```

Allow the user to name the adventure.

```
100 DIS$="D:ADVENAME":CLOSE #1
110 ? "[esc ctrl clear]":POSITION 2,4:
? "WHAT IS THE NAME OF THIS ADVENTURE
   PLEASE LIMIT TO 20 CHARACTERS.":?
115 ? "FOR SPECIAL COLORS USE [inverse
] INVERSE [inverse off] AND  lower cas
e LETTERS!!":?
120 TRAP 120:INPUT #3,NA$
```

Print the adventure name to disk.

```
130 TRAP 1100:OPEN #1,8,0,DIS$:PRINT #
1;NA$
```

Trap error and inform the user what is wrong.

```
1100 TRAP 40000:CLOSE #1:? :? "[esc ct
rl 2] UNLOCK DISK AND [inverse] PRESS
RETURN ."
```

Accept only the RETURN key.

```
1110 GET #2,K:IF K<>155 THEN 1110
1120 GOTO 130
```

Allow the user to input his or her name.

```
140 NA$="":? "INPUT YOUR NAME - LIMIT
OF 20 LETTERS":?
150 TRAP 150:INPUT #3,NA$
```

Print author name to disk.

```
160 PRINT #1;NA$:CLOSE #1
```

Let the user know the task is completed, jump to delay loop, and prepare for the next phase.

```
180 ? "[esc ctrl clear]":POSITION 2,4:
? "PHASE ONE COMPLETED!":GOSUB 1000:DI
S$="D:ADVEN.DAT"
```

Delay loop.

```
1000 FOR WAIT=1 TO 400:NEXT WAIT:RETUR
N
```

Clear screen, format output, print message and wait for input.

```
190 PRINT "[esc ctrl clear]":POSITION
2,4:PRINT "INPUT NUMBER OF ROOMS ":? :
TRAP 190:INPUT #3,NR
```

Only accept input between 5 and 50.

```
200 IF NR<5 THEN ? "A ";NR;"-ROOM DUNG
EON IS VERY SMALL.":? "MAKE A BIGGER O
NE!":GOSUB 1000:GOTO 190
210 IF NR>50 THEN ? "YOU CAN ONLY CREA
TE A 50-ROOM DUNGEON ON ONE SIDE OF TH
E DISK.":GOSUB 1000:GOTO 190
```

Input accepted; print number rooms to disk.

```
215 OPEN #1,8,0,DIS$:PRINT #1;NR
```

More user input:

```
220 ? :? "HOW MANY MOVABLE OBJECTS ARE
 IN YOUR  ADVENTURE":TRAP 220:INPUT #3
,OBS:IF OBS<=0 THEN 220
```

Dimension more variables (only those which are needed by user); clear the string.

```
225 DIM IRN(OBS),VI(OBS),ITEM$(OBS*20)
:ITEM$(1)=" ":ITEM$(OBS*20)=" ":ITEM$(
2)=ITEM$
```

Clear screen and relay more information to the user.

```
230 ? "[esc ctrl clear]":POSITION 2,4:
? "OBJECT NAMES MUST BE MORE THAN 3 AN
D  LESS THAN 20 LETTERS."
232 ? :? "YOU MUST ALSO REMEMBER TO MA
KE THE     FIRST FOUR LETTERS OF EACH W
ORD        DISTINCTLY DIFFERENT."
234 ? :? "[inverse] EXAMPLE: [inverse
off] BOOKCASE and BOOKEND WOULD BY CON
SIDERED THE SAME BY THE PROGRAM."
236 ? "YOU COULD SAY 'DUSTY BOOKCASE'
or      'HEAVY BOOKEND' THEN THE INTERP
RETER  WOULD UNDERSTAND."
237 ? :? "PRESS [inverse] SPACE BAR [i
nverse off] TO CONTINUE"
238 GET #2,K:IF K<>32 THEN 238
239 ? "[esc ctrl clear]":POSITION 2,4:
? "[esc ctrl 2]USE UPPER CASE LETTERS
FOR REST OF    PROGRAM.":GOSUB 1000
```

The program loops based on the number of objects in the user's adventure:

```
240 FOR LP=1 TO OBS:? "[esc ctrl clear
]":POSITION 2,4
250 TRAP 250:? "INPUT NAME OF OBJECT #
";LP:? :INPUT #3,NA$:IF LEN(NA$)<4 THE
N 250
255 ITEM$(LP*20-19,LP*20)=NA$
260 TRAP 260:? :? "INPUT ROOM NUMBER O
F ";NA$:? :INPUT #3,RM:IF RM<1 OR RM>N
R THEN 260
262 IRN(LP)=RM
265 TRAP 265:? :? "INPUT VALUE OF ";NA
$:? :INPUT #3,V
270 IF V>200000 THEN ? "SORRY, MAXIMUM
 VALUE CAN ONLY BE       $200000.":GOTO
 265
275 VI(LP)=V:NA$="":NEXT LP
```

When all objects and data are entered, print the information to the disk.

```
280 PRINT #1;OBS:PRINT #1;ITEM$:FOR LP
=1 TO OBS:PRINT #1;IRN(LP):PRINT #1;VI
(LP):NEXT LP:CLOSE #1
```

Inform the user that the operation is completed, and prepare for the next phase.

```
290 ? "[esc ctrl clear]":POSITION 2,4:
? "PHASE TWO COMPLETED!!":GOSUB 1000:D
IS$="D:ROOM"
```

Loop to input information about each room ard direction data.

```
300 FOR LP=1 TO NR:POSITION 2,4
310 IF LP<10 THEN DIS$(7,7)=STR$(LP):G
OTO 330
320 DIS$(7,8)=STR$(LP)
330 ? "INPUT DESCRIPTION OF ROOM #";LP
:? "(MAXIMUM OF THREE LINES)"
332 ? "FORMAT SCREEN SO WORDS ARE NOT
SPLIT BETWEEN LINES.":?
335 TRAP 335:INPUT #3,RD$
340 ? :? "INPUT ROOM NUMBER FOR EACH D
IRECTION  NORTH,SOUTH,EAST,WEST,UP,DOW
N":?
350 ? "IF NO ROOM IN THAT DIRECTION, E
NTER 0":?
360 TRAP 1200:INPUT #3,N,S,E,W,U,D
370 OPEN #1,8,0,DIS$:PRINT #1;RD$:PRIN
T #1,N:PRINT #1,S:PRINT #1,E:PRINT #1,
W:PRINT #1,U:PRINT #1,D:CLOSE #1
380 NEXT LP
```

Trap any errors in user input.

```
1200 TRAP 40000:PRINT "YOU MUST ENTER
A NUMBER FOR EACH      DIRECTION.":GOS
UB 1000:GOTO 340
```

Another section completed.

```
390 ? "[esc ctrl clear]":POSITION 2,4:
? "PHASE THREE COMPLETED!":GOSUB 1000
```

Prepare for special information.

```
400 ? "[esc ctrl clear]":POSITION 2,4:
? "ARE THERE ANY ROOMS THAT NEED A
   SPECIAL ITEM TO GAIN ADMITTANCE.
   (Y/N)"
410 GET #2,K:IF K<>89 AND K<>78 THEN 4
10
420 IF K=78 THEN 800
430 ? :? "PLEASE LIMIT ROOMS TO A TOTA
L OF 24.  (MAXIMUM OF 4 IN EACH DIRECT
ION)."
435 ? :? "HOW MANY ROOMS ARE NORTH?":G
OSUB 1250
```

Subroutine to handle input:

```
1250 ? "ENTER [inverse] 0 [inverse off
] IF NONE!"
1260 GET #2,K:K=K-48:IF K>4 THEN ? "PL
EASE LIMIT TO 4 OR LESS":GOTO 1260
1270 RETURN
```

Back to the main program.

```
450 IF K=0 THEN 500
460 OPEN #1,8,0,"D:SPNORTH.DAT":PRINT
#1;K
470 GOSUB 1300:GOSUB 1400
```

Routine to remind the user of an important fact.

```
1300 ? "[esc ctrl clear]":POSITION 2,4
:? "REMEMBER, YOU MUST ENTER THE NUMBE
R OFTHE ROOM WHICH LETS YOU GAIN ACCES
S"
1310 ? "TO THE SECRET ROOM.":GOSUB 100
0:GOSUB 1000:RETURN
```

Subroutine for user input (used for all directions).

```
1400 FOR LP=1 TO K:? "[esc ctrl clear]
":POSITION 2,4
1410 ? "INPUT #";LP:? :? " - FROM ROOM
 NUMBER ":TRAP 1410:? :INPUT #3,RM:DIR
(LP)=RM
1420 ? :? "INPUT ITEM [inverse](NUMBER
 ONLY)[inverse off] NEEDED TO    GAIN
ACCESS FROM ROOM #";RM:TRAP 1420:? :IN
PUT #3,IT:ITEM(LP)=IT
```

```
1430 NEXT LP:FOR LP=1 TO K:PRINT #1;DI
R(LP):PRINT #1;ITEM(LP):NEXT LP:CLOSE
#1:RETURN
```

Return again to main program.

```
500 ? :? "HOW MANY ROOMS ARE SOUTH?":G
OSUB 1250
520 IF K=0 THEN 550
525 OPEN #1,8,0,"D:SPSOUTH.DAT":PRINT
#1;K
530 GOSUB 1300:GOSUB 1400
550 ? :? "HOW MANY ROOMS ARE EAST?":GO
SUB 1250
560 IF K=0 THEN 600
570 OPEN #1,8,0,"D:SPEAST.DAT":PRINT #
1;K
575 GOSUB 1300:GOSUB 1400
600 ? :? "HOW MANY ROOMS ARE WEST?":GO
SUB 1250
610 IF K=0 THEN 650
615 OPEN #1,8,0,"D:SPWEST.DAT":PRINT #
1;K
620 GOSUB 1300:GOSUB 1400
650 ? :? "HOW MANY ROOMS ARE UP?":GOSU
B 1250
660 IF K=0 THEN 700
665 OPEN #1,8,0,"D:SPUP.DAT":PRINT #1;
K
670 GOSUB 1300:GOSUB 1400
700 ? :? "HOW MANY ROOMS ARE DOWN?":GO
SUB 1250
710 IF K=0 THEN 800
720 OPEN #1,8,0,"D:SPDOWN.DAT":PRINT #
1;K
730 GOSUB 1300:GOSUB 1400
```

Another phase completed.

```
800 ? "[esc ctrl clear]":POSITION 2,4:
? "PHASE FOUR COMPLETED!":GOSUB 1000
```

Create monsters.

```
810 ? "[esc ctrl clear]":POSITION 2,4:
? "ARE THERE ANY CREATURES TO FIGHT?
  (Y/N)":?
820 GET #2,K:IF K<>78 AND K<>89 THEN 8
20
830 IF K=78 THEN 900
840 ? :? "HOW MANY CREATURES":TRAP 840
:INPUT #3,NC
850 IF NC>NR THEN ? "THERE ARE MORE MO
NSTERS THAN ROOMS IN YOUR DUNGEON":GOS
UB 1000:GOTO 840
860 OPEN #1,8,0,"D:CREATURE.DAT":PRINT
 #1,NC
870 DIM MON(NC):FOR LP=1 TO NC:? "[esc
 ctrl clear]":POSITION 2,4
880 ? "INPUT ROOM OF CREATURE #";LP:TR
AP 880:INPUT #3,RM
885 MON(LP)=RM:NEXT LP
890 FOR LP=1 TO NC:PRINT #1;MON(LP):NE
XT LP:CLOSE #1
```

Finally, all information loaded.

```
895 ? "[esc ctrl clear]":POSITION 2,4:
? "PHASE FIVE COMPLETED!":CLOSE #3:GOS
UB 1000
```

Allow user the opportunity to try the new dungeon.

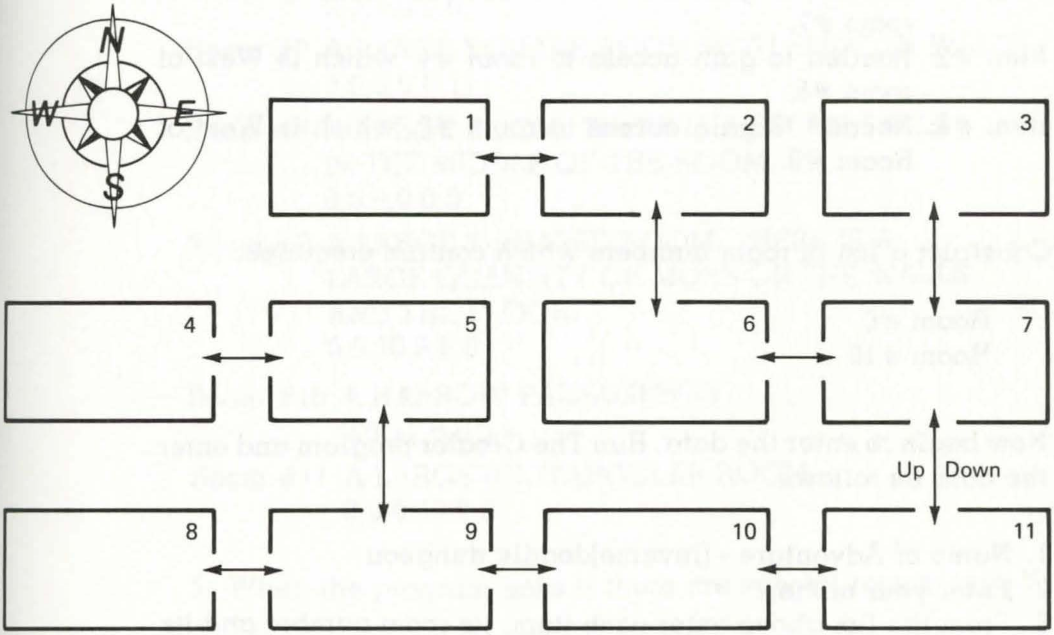
```
900 ? "ARE YOU READY TO TRY YOUR DUNGE
ON?    (Y/N)"
910 GET #2,K:IF K<>89 AND K<>78 THEN 4
10
920 IF K=78 THEN END
930 TRAP 970:? "INSERT DISK WITH INTER
PRETER           [inverse] PRESS [inver
se off]RETURN."
940 GET #2,K:IF K<>155 THEN 940
950 CLOSE #2:RUN "D:INTERPRE.TER"
```

```
960 END

970 TRAP 40000:OPEN #2,4,0,"K:":GOTO 9
30
```

The Creator program is now complete. Be sure to save a copy to disk with SAVE "D:filename."

SAMPLE DATA FOR THE CREATOR PROGRAM

As always, begin with the construction of a map. Always make sure the player will begin the adventure from Room #1.



Next, number, name, assign a location number, and declare a value for each object in the adventure:

ITEM	NAME	LOCATION #	VALUE
1	Glass skull	8	20000
2	Magic ball	3	10000
3	Scroll	10	500
4	Ruby ring	4	100000
5	Brass key	5	5
6	Golden crown	11	200000
7	Dusty rag	7	0

Using the number of the item above, construct a list of items needed to help gain access to special rooms:

Item #5: Needed to gain access to room #3, which is North of room #7.
Item #2: Needed to gain access to room #4, which is West of room #5.
Item #4: Needed to gain access to room #8, which is West of Room #9.

Construct a list of room numbers which contain creatures:

Room #6
Room #10

Now begin to enter the data. Run **The Creator** program and enter the data as follows.

1. Name of Adventure - [inverse]deadly dungeon
2. Enter your name.
3. From the list above enter each item, its room number and its value.
4. The description of each room and the number of the connecting room. (NOTE: The descriptions are formatted as they should appear when entered.)

Room #1 THE ENTRANCE OF A CAVERN.
0,0,2,0,0,0

Room #2 A VERY LARGE CAVERN. THE FLOOR SLANTS DOWNWARD AND THE WALLS ARE VERY DAMP.
0,6,0,1,0,0

Room #3 A VERY SMALL ROOM. THERE IS MOLD ON THE WALLS. THERE IS A VERY PUNGENT ODOR HERE.
0,7,0,0,0,0

Room #4 A ROOM WITH A LOW CEILING. THE ROOF IS HELD UP BY ROTTING TIMBERS.
0,0,5,0,0,0

Room #5 AN L-SHAPED ROOM WITH A DAMP FLOOR.
0,9,0,4,0,0

Room #6 A LARGE PASSAGEWAY.
2,0,7,0,0,0

Room #7 A LARGE SQUARE ROOM WITH SMOOTH WALLS.
3,0,0,6,0,11

Room #8 A LARGE ROOM. THERE IS A STONE TABLE IN THE MIDDLE OF THE ROOM.
0,0,9,0,0,0

Room #9 A LARGE T-SHAPED ROOM. THERE IS A LARGE QUANTITY OF MOSS ON THE WALLS AND THE FLOOR.
5,0,10,8,0,0

Room #10 A NARROW PASSAGEWAY.
0,0,11,9,0,0

Room #11 A LARGE RECTANGULAR ROOM.
0,0,0,10,7,0

5. When the program asks if there are special rooms, type Y.

How many are North - type 1.
Room #7 - Item #5 (Refer to the list from above)
How many are South - type 0.
How many are East - type 0.

How many are West - type 2.
Room #5 - Item #2
Room #9 - Item #4
How many are Up - type 0.
How many are Down - type 0.

6. When the program asks if there are any creatures, type Y.

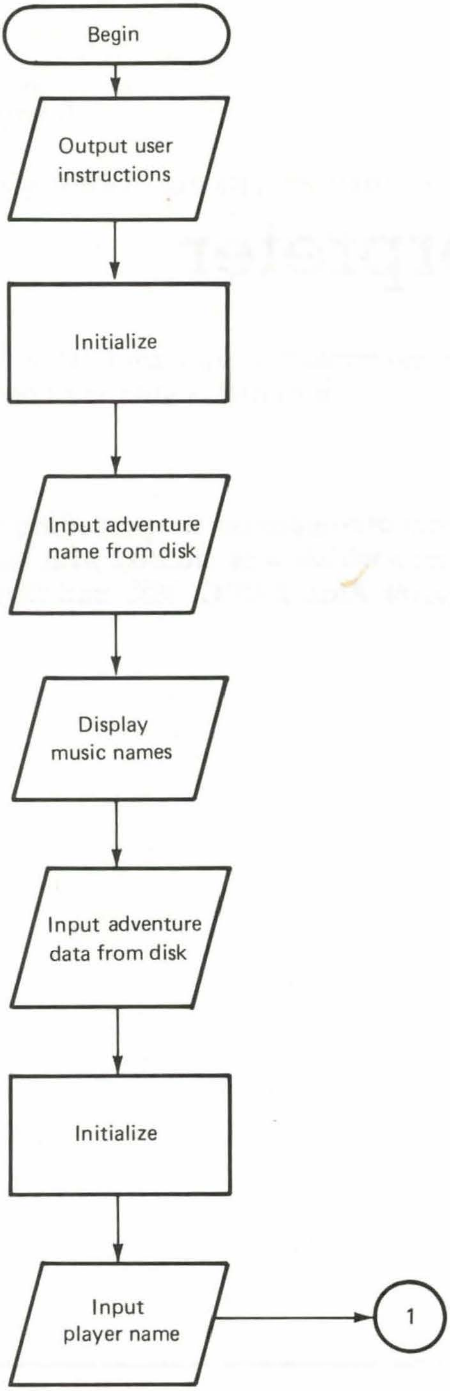
How many - type 2.
Room #2 - Room #10

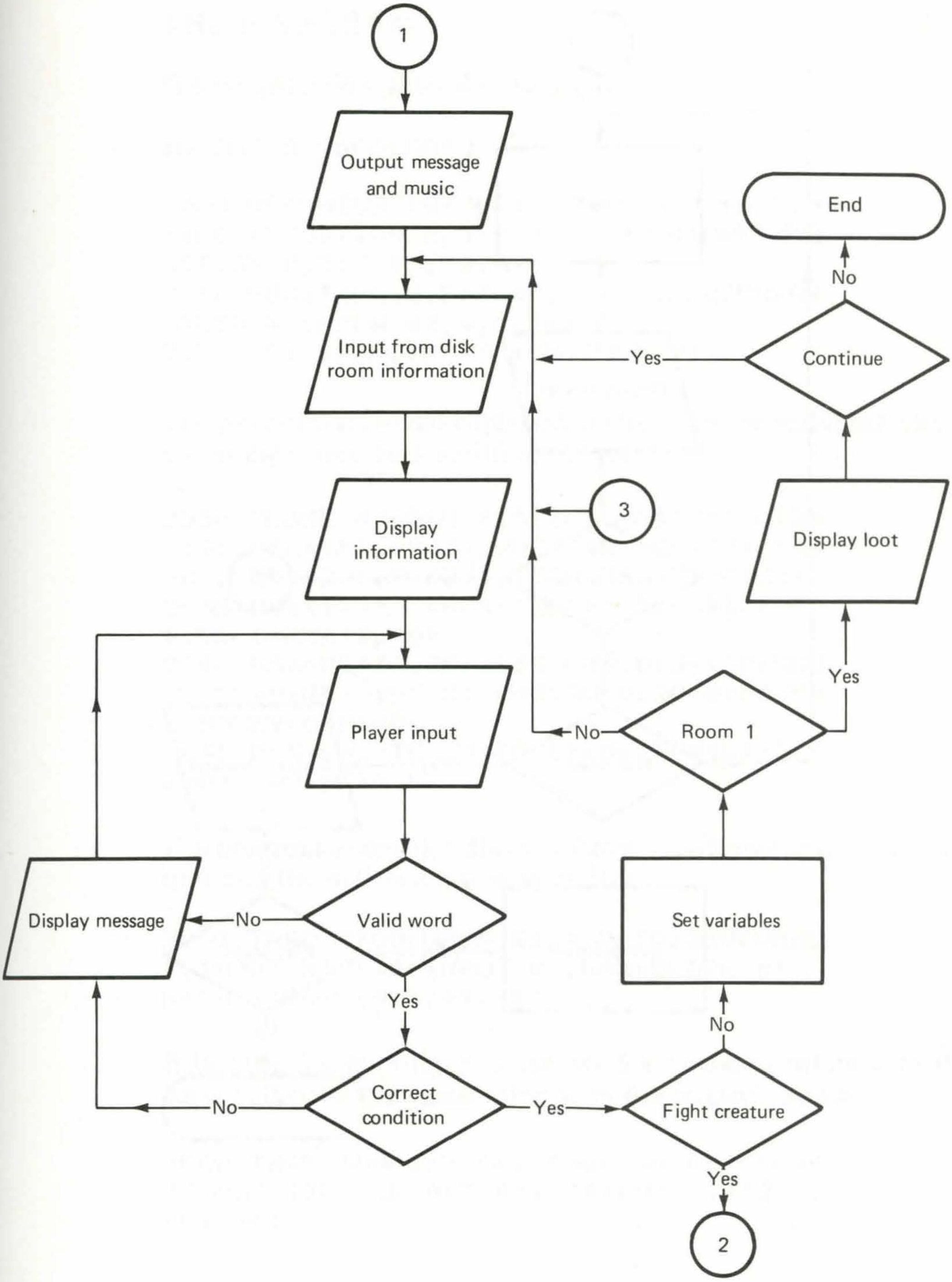
This completes the entry of all data. Key in **Interpreter** and RUN using the disk just created to supply information.

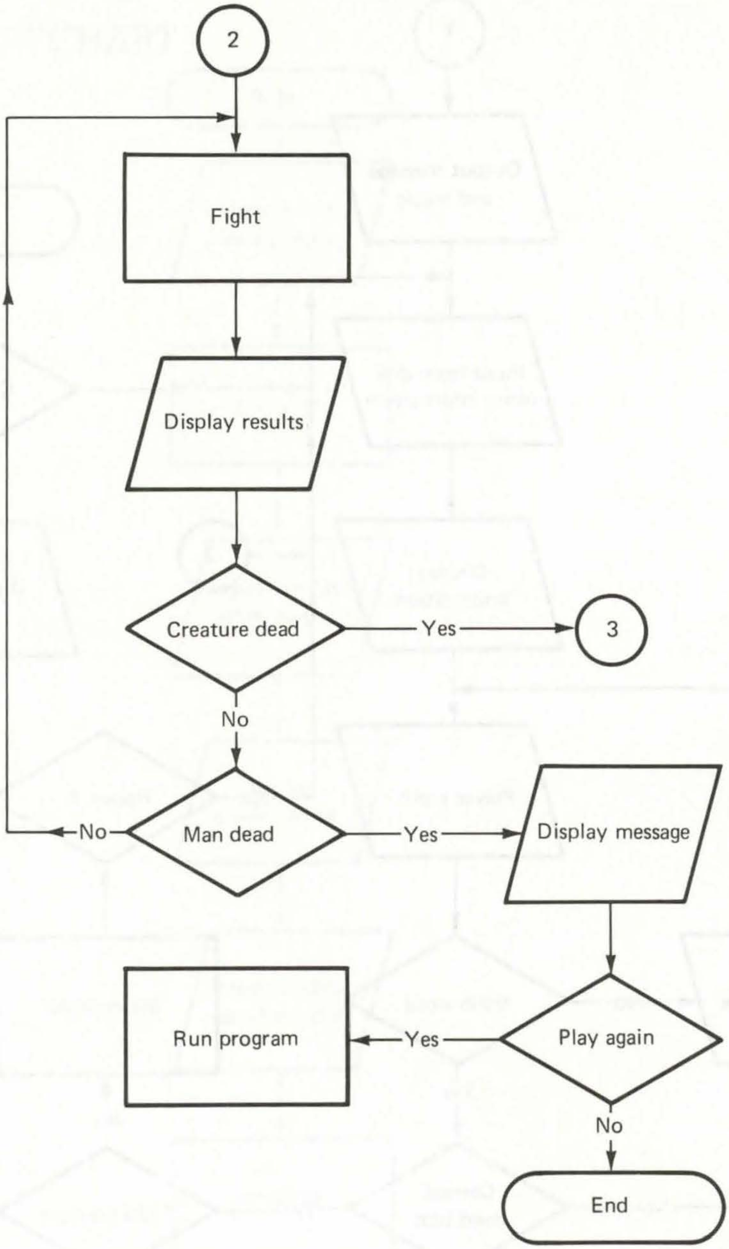
Interpreter

This program accompanies the preceding one and is used to play the adventure which was created with that program. The **Interpreter** requires Atari BASIC, 16K, and disk drive.

THE FLOWCHART




1
Output message and music
Input from disk room information
Display information
Player input
Valid word
No
Display message
Yes
Correct condition
No
Yes
Fight creature
Yes
2
No
Set variables
Room 1
No
Yes
Display loot
Continue
Yes
No
End
3


2
Fight
Display results
Creature dead
Yes
3
No
Man dead
No
Yes
Display message
Play again
Yes
Run program
No
End

THE PROGRAM

The program flow should be familiar now. Start with Instructions.

```
10 GOSUB 2000:GOTO 30
```

```
2000 GRAPHICS 18:POSITION 7,3:? #6;"in
sert":POSITION 6,4:? #6;"ADVENTURE":PO
SITION 8,5:? #6;"DISK"
2010 POSITION 5,8:? #6;"[inverse]PRESS
 RETURN":OPEN #2,4,0,"K:"
2020 GET #2,K:IF K<>155 THEN 2020
```

The program closes the keyboard, dimensions the variables, clears the strings, and sets variable values.

```
2030 CLOSE #2:DIM TI$(20),NA$(20),DIS$
(14),DIR$(6),V$(3),VOC$(48),M$(30),I$(
20),RD$(120),SR(24),ITEM(24),TEMP(24)
2035 RD$(1)=" ":RD$(120)=" ":RD$(2)=RD
$:VOC$=RD$(1,48)
2040 NA$=RD$(1,20):TI$=NA$:DIS$="D:ROO
M":SN=0:SS=0:SE=0:SW=0:SU=0:SD=0:ROOM=
1:MONEY=0:ST=0
2045 FOR L=1 TO 24:SR(L)=0:ITEM(L)=0:T
EMP(L)=0:NEXT L
```

The program opens the disk file Adventure Name and inputs the title and the author's name from disk.

```
2050 TRAP 2700:OPEN #1,4,0,"D:ADVENAME
":INPUT #1,TI$:INPUT #1,NA$:CLOSE #1:L
N=INT((20-LEN(TI$))/2)
```

It locates the starting position for the title and prints it to the screen. It does the same thing with the author's name.

```
2060 TRAP 40000:? #6;"[esc shift clear
]":POSITION LN,4:? #6;TI$:LN=INT((20-L
EN(NA$))/2)
```

```
2065 POSITION 9,7:PRINT #6;"BY":POSITI
ON LN,8:PRINT #6;NA$
```

It creates a sound effect.

```
2070 FOR V=0 TO 15 STEP 0.5:SOUND 0,60
,12,V:SOUND 1,120,12,V:SOUND 2,60,12,V
:SOUND 3,120,12,V:NEXT V
2080 FOR V=15 TO 0 STEP -0.5:SOUND 0,6
0,12,V:SOUND 1,120,12,V:SOUND 2,60,12,
V:SOUND 3,120,12,V:NEXT V
```

If the user did not have the proper disk in the drive the program would trap (line 2050) the error and send control below.

```
2700 ? "[esc ctrl 2]":POKE 710,66:POKE
 712,66:FOR T=1 TO 30:NEXT T:RUN
```

The program reads the rest of the information that the creator stored on disk.

Note: The program uses the first of the inputs to dimension other variables. Remember OBS from Creator was the number of objects in the adventure.

```
2090 OPEN #1,4,0,"D:ADVEN.DAT":INPUT #
1,NR:INPUT #1,OBS:DIM IRN(OBS),VI(OBS)
,ITEM$(20*OBS)
```

After dimensioning the string and variables to hold room numbers and value of the object, the program begins to input information from the disk. This loads the variables with the desired information.

```
2100 INPUT #1,ITEM$:FOR LP=1 TO OBS:IN
PUT #1,RM:INPUT #1,VA:IRN(LP)=RM:VI(LP
)=VA:NEXT LP:CLOSE #1
```

Now it opens the special files for information concerning a certain direction.

Note: The trap statement sends execution of the program to the next line for special direction data. Remember, the Creator program only created files for which there was special information for that direction. If no file was made, the Interpreter program would crash and tell you that no such file existed. By trapping the program to the next line it will just jump to that line and try to open the next disk file.

```
2110 TRAP 2130:OPEN #1,4,0,"D:SPNORTH.
DAT":INPUT #1,SN
2120 FOR LP=1 TO SN:INPUT #1,RM:INPUT
#1,I:SR(LP)=RM:ITEM(LP)=I:NEXT LP
2130 CLOSE #1:TRAP 40000:TRAP 2150:OPE
N #1,4,0,"D:SPSOUTH.DAT":INPUT #1,SS
2140 FOR LP=SN+1 TO SN+SS:INPUT #1,RM:
INPUT #1,I:SR(LP)=RM:ITEM(LP)=I:NEXT L
P
2150 CLOSE #1:TRAP 40000:TRAP 2170:OPE
N #1,4,0,"D:SPEAST.DAT":INPUT #1,SE
2160 FOR LP=SN+SS+1 TO SN+SS+SE:INPUT
#1,RM:INPUT #1,I:SR(LP)=RM:ITEM(LP)=I:
NEXT LP
2170 CLOSE #1:TRAP 40000:TRAP 2190:OPE
N #1,4,0,"D:SPWEST.DAT":INPUT #1,SW
2180 FOR LP=SN+SS+SE+1 TO SN+SS+SE+SW:
INPUT #1,RM:INPUT #1,I:SR(LP)=RM:ITEM(
LP)=I:NEXT LP
2190 CLOSE #1:TRAP 40000:TRAP 2210:OPE
N #1,4,0,"D:SPUP.DAT":INPUT #1,SU
2200 FOR LP=SN+SS+SE+SW+1 TO SN+SS+SE+
SW+SU:INPUT #1,RM:INPUT #1,I:SR(LP)=RM
:ITEM(LP)=I:NEXT LP
2210 CLOSE #1:TRAP 40000:TRAP 2230:OPE
N #1,4,0,"D:SPDOWN.DAT":INPUT #1,SD
2220 FOR LP=SN+SS+SE+SW+SU+1 TO SN+SS+
SE+SW+SU+SD:INPUT #1,RM:INPUT #1,I:SR(
LP)=RM:ITEM(LP)=I:NEXT LP
2230 CLOSE #1:TRAP 40000:TRAP 2250:OPE
```

```
N #1,4,0,"D:CREATURE.DAT":INPUT #1,CRE
2240 DIM MON(CRE):FOR LP=1 TO CRE:INPU
T #1,RM:MON(LP)=RM:NEXT LP
2250 CLOSE #1:TRAP 40000
```

Now the program reads in the data common to every adventure, the vocabulary and one-letter direction commands.

```
2260 FOR LP=1 TO 16:READ V$:VOC$(LP*3-
2,LP*3)=V$:NEXT LP
2270 FOR LP=1 TO 6:READ V$:DIR$(LP,LP)
=V$:NEXT LP
3000 DATA WAL,RUN,GO ,GET,TAK,DRO,GIV,
LOO,EXA,TOU,FEE,OPE,UNL,FIG,KIL,INV,N,
S,E,W,U,D
```

The program now checks the number of objects in this adventure and sets the player's initial strength by the value of each object. Thus, a valuable object will cost more strength points to pick up and, after fighting a dungeon full of creatures, the player may not have the strength to pick up such an object.

```
2280 FOR LP=1 TO OBS:ST=ST+INT(VI(LP)/
1000):NEXT LP
```

The program requests the player to input his or her name and then assigns a message addressed to the player.

```
2290 GRAPHICS 0:POKE 710,0:POSITION 14
,8:? "[inverse]   WELCOME!!   [inverse
 off]":? "ENTER YOUR NAME INTO THE BOO
K OF THE  DEAD."
2300 TRAP 2300:INPUT NA$
2310 POKE 752,1:PRINT :PRINT "MAY THE
BLESSINGS OF THE GODS BE      WITH YOU
 ";NA$;"!"
2320 GOSUB 2400:RETURN
```

The subroutine to play the death march:

```
2400 RESTORE 3100:FOR L=1 TO 9:READ SO
,DELAY:SOUND 0,SO,10,8:FOR T=1 TO DELA
Y:NEXT T:SOUND 0,0,0,0
2410 FOR T=1 TO 8:NEXT T:NEXT L:RETURN
3100 DATA 100,60,100,40,100,10,100,60,
85,80,90,60,100,40,105,20,100,60
```

The program returns to beginning and sets variables to zero, clears the end of DIS$, and resets to new room number.

```
30 M=0:N=0:S=0:E=0:W=0:U=0:D=0:SPD=0:D
IS$(7,8)="  ":DIS$(7,8)=STR$(ROOM):TRA
P 1750
```

The program opens the room file and inputs description and information about what number room lies in each direction.

```
40 OPEN #1,4,0,DIS$:INPUT #1,RD$:INPUT
 #1,N:INPUT #1,S:INPUT #1,E:INPUT #1,W
:INPUT #1,U:INPUT #1,D:CLOSE #1
```

If no such room exists, the program will come to the trap statement from line 30.

Note: This should never happen unless the disk is destroyed or if data was entered incorrectly into the Creator program.

```
1750 ? "[esc shift clear]":POSITION 2,
4:? "[esc ctrl 2]CHECK DISK DIRECTORY
FOR ROOM #";ROOM:STOP
```

The program colors the room by using the value of the variable, ROOM.

```
45 POKE 710,INT(ROOM/3.5)*16+2:POKE 71
2,PEEK(710)
```

It prints the information.

```
50 PRINT "[esc shift clear]":POSITION
2,0:? "[inverse]$$ WEALTH=>[inverse of
f]$";MONEY:POSITION 21,0:? "[inverse/2
 ctrl period’s/space]STRENGTH=>";ST
60 PRINT "[inverse/37 ctrl T’S]"
70 PRINT "[inverse/ 14 spaces]LOCATION
:[inverse/14 spaces]":PRINT :PRINT RD$
80 PRINT :PRINT "[inverse/ctrl Q/8 ctr
l W’s/19 ctrl R’s/8 ctrl W’s/ctrl E]"
90 PRINT "[inverse/ctrl Z/7 ctrl X’s/c
trl C/space]THERE ARE EXITS:[2 spaces/
ctrl Z/7 ctrl X’s/ctrl C]"
```

Before printing the exits, it checks to see if there are any special rooms with pathways to them blocked.

```
100 PRINT :IF SN THEN GOSUB 1500
110 IF N THEN PRINT "NORTH, ";
120 IF SS THEN GOSUB 1520
130 IF S THEN PRINT "SOUTH, ";
140 IF SE THEN GOSUB 1540
150 IF E THEN PRINT "EAST, ";
160 IF SW THEN GOSUB 1560
170 IF W THEN PRINT "WEST, ";
180 IF SU THEN GOSUB 1580
190 IF U THEN PRINT "UP, ";
200 IF SD THEN GOSUB 1600
210 IF D THEN PRINT "DOWN, ";
220 PRINT "[2 esc ctrl +’s/space]"
```

Special directions? Yes! Print these directions in inverse.

```
1500 FOR LP=1 TO SN:IF SR(LP)=ROOM AND
 ITEM(LP) THEN N=0:PRINT "[inverse] NO
RTH [inverse off], ";:SPD=1
1510 NEXT LP:RETURN
1520 FOR LP=SN+1 TO SN+SS:IF SR(LP)=RO
OM AND ITEM(LP) THEN S=0:PRINT "[inver
se] SOUTH [inverse off], ";:SPD=1
1530 NEXT LP:RETURN
1540 FOR LP=SN+SS+1 TO SN+SS+SE:IF SR(
LP)=ROOM AND ITEM(LP) THEN E=0:PRINT "
```

```
[inverse] EAST [inverse off], ";:SPD=1
1550 NEXT LP:RETURN
1560 FOR LP=SN+SS+SE+1 TO SN+SS+SE+SW:
IF SR(LP)=ROOM AND ITEM(LP) THEN W=0:P
RINT "[inverse] WEST [inverse off], ";
:SPD=1
1570 NEXT LP:RETURN
1580 FOR LP=SN+SS+SE+SW+1 TO SN+SS+SE+
SW+SU:IF SR(LP)=ROOM AND ITEM(LP) THEN
 U=0:PRINT "[inverse] UP [inverse off]
, ";:SPD=1
1590 NEXT LP:RETURN
1600 FOR LP=SN+SS+SE+SW+SU+1 TO SN+SS+
SE+SW+SU+SD:IF SR(LP)=ROOM AND ITEM(LP
) THEN D=0:PRINT "[inverse] DOWN [inve
rse off], ";:SPD=1
1610 NEXT LP:RETURN
```

The program checks for items in the room and prints them.

```
230 PRINT :PRINT "[inverse/10 ctrl R's
/space]ITEMS YOU SEE:[space/11 ctrl R'
s]"
240 PRINT :FOR LP=1 TO OBS
250 IF IRN(LP)=ROOM THEN I$=ITEM$(LP*2
0-19,LP*20):GOSUB 1650:PRINT I$;", ";:
I$=""
260 NEXT LP:PRINT "[2 esc ctrl +'s/spa
ce]"
```

Subroutines delete any spaces at the end of an object name.

```
1650 IF I$(LEN(I$),LEN(I$))=" " THEN I
$=I$(1,LEN(I$)-1):GOTO 1650
1660 RETURN
```

The program checks to see if there are any creatures here.

```
270 FOR L=1 TO CRE:IF MON(L)=ROOM THEN
 PRINT "THERE IS A MONSTER HERE!":M=L
275 NEXT L:PRINT
```

It asks for user input.

```
280 PRINT "[inverse 12 spaces]INSTRUCT
IONS:[12 spaces]":PRINT
290 POKE 764,255:POKE 752,0:TRAP 1700:
M$="":TI$=""
300 INPUT M$:POKE 752,1:IF LEN(M$)>1 T
HEN 400
```

The program determines whether input was a valid one-letter command.

```
310 FOR LP=1 TO 6:IF DIR$(LP,LP)=M$ TH
EN DIR=LP:POP :GOTO 330
320 NEXT LP:PRINT "I DIDN'T UNDERSTAND
 THAT!":GOTO 1710
```

Then it checks to see if the player can go in that direction.

Note: There may be several reasons why a player can't go in a direction: the monster won't let him, a special condition of that room, or that there is nothing in that direction.

```
330 IF M THEN PRINT "THE MONSTER WON'T
 LET YOU!":GOTO 1710
335 ON DIR GOTO 340,350,360,370,380,39 0
340 IF N THEN ROOM=N:GOTO 20
345 GOTO 395
350 IF S THEN ROOM=S:GOTO 20
355 GOTO 395
360 IF E THEN ROOM=E:GOTO 20
365 GOTO 395
370 IF W THEN ROOM=W:GOTO 20
375 GOTO 395
380 IF U THEN ROOM=U:GOTO 20
385 GOTO 395
390 IF D THEN ROOM=D:GOTO 20
395 IF SPD THEN 1720
397 PRINT "YOU CANT'T GO IN THAT DIREC
TION!":GOTO 1710
```

A special message:

```
1720 PRINT "[inverse/esc ctrl 2/space]
EXITS LISTED IN INVERSE REQUIRE[7 spac
es]SPECIAL OBJECTS TO USE!![inverse of
f/esc ctrl -]":GOTO 1710
```

Now the routine to wait for a message to be read before returning for more input. (This is also a part of the trap statement in line 290.)

```
1700 PRINT "[esc ctrl 2]PLEASE SAY THA
T AGAIN!":TRAP 40000
1710 FOR LP=1 TO 250:NEXT LP:PRINT "[3
 exc ctrl -’s/9 esc inserts]":GOTO 290
```

In order to enable the player to move in that direction, add the following:

```
20 IF ROOM=1 THEN GOSUB 1900
1900 GRAPHICS 18:POSITION 1,1:PRINT #6
;"[inverse]YOU ARE IN ROOM #1"
1910 POSITION 1,3:PRINT #6;"YOU HAVE $
";MONEY:POSITION 4,4:PRINT #6;"IN TREA
SURE"
1920 LN=((20-LEN(NA$))/2):POSITION LN,
6:PRINT #6;NA$:GOSUB 2500
1930 POSITION 5,8:PRINT #6;"do you wan
t":POSITION 1,9:PRINT #6;"to continue
(Y/N)":OPEN #2,4,0,"K:"
1940 GET #2,K:IF K<>78 AND K<>89 THEN
1940
1950 CLOSE #2:IF K=78 THEN POSITION 0,
7:PRINT #6;"[inverse]LIVE LONG & PROSP
ER!":FOR T=1 TO 250:NEXT T:END
1960 GRAPHICS 0:POKE 710,0:RETURN
```

If the input was more than one letter, the program looks for a space and separates the words.

```
400 FOR LP=1 TO LEN(M$):IF M$(LP,LP)="
 " THEN TI$=M$(LP+1,LEN(M$)):POP :GOTO
 420
410 NEXT LP
```

It checks for a verb that is part of the vocabulary and jumps to the proper rountine.

```
420 FOR L=1 TO 16:IF M$(1,3)=VOC$(L*3-
2,L*3) THEN VOC=L:POP :GOTO 440
430 NEXT L:PRINT "THAT IS NOT PART OF
MY VOCABULARY!":GOTO 1710
440 ON VOC GOTO 450,450,450,470,470,53
0,530,570,570,610,610,700,700,750,750,
1000
```

The GO, WALK, RUN routine sends control back to the one-letter direction routine to see if player can go in that direction.

```
450 FOR L=1 TO 6:IF TI$(1,1)=DIR$(L,L)
 THEN DIR=L:POP :GOTO 330
460 NEXT L:PRINT "YOU CAN'T GO THERE!"
:GOTO 1710
```

In TAKE and GET routines, the program checks to see if the object requested is one of the objects loaded into the Item string, then checks to see if player has already picked it up [IRN(n)=0 no room number], checks to see if it is in the same room as the player, checks to see if the player is strong enough, checks to see if the object is needed to set special directions, then lets the object be picked up and adds the value to the player's wealth.

```
470 FOR L=1 TO OBS:IF TI$(1,4)=ITEM$(L
*20-19,L*20-16) THEN POP :GOTO 490
480 NEXT L:PRINT "YOU CAN'T TAKE A ";T
I$:GOTO 1710
490 IF IRN(L)=0 THEN PRINT "YOU ALREAD
Y HAVE IT!":GOTO 1710
500 IF IRN(L)<>ROOM THEN PRINT "THAT I
```

```
TEM IS NOT HERE!":GOTO 1710
510 IF ST-INT(VI(L)/1000)<0 THEN PRINT
 "YOU ARE TOO WEAK TO CARRY THAT ITEM!
":GOTO 1710
512 FOR LP=1 TO SN+SS+SE+SW+SU+SD:IF I
TEM(LP)=L THEN TEMP(LP)=L:ITEM(LP)=0
515 NEXT LP
520 ST=ST-INT(VI(L)/1000):MONEY=MONEY+
VI(L):IRN(L)=0:GOTO 30
```

In the routine for DROP and GIVE, the program checks first to see if it is a valid item.

```
530 FOR L=1 TO OBS:IF TI$(1,3)=ITEM$(L
*20-19,L*20-17) THEN POP :GOTO 545
540 NEXT L:PRINT "DROP WHAT!!!":GOTO 1
710
```

After ascertaining validity, the program checks to see if there is a monster in the room. If there is, there is a probability that it will attack the player if he or she begins dropping items in order to regain strength.

```
545 IF M AND RND(0)>0.5 THEN PRINT "YO
U'VE BEEN ATTACKED!":GOSUB 2500:GOTO 7
70
```

If the player has the object, the program places it in the room, resets any special directions associated with the object, and subtracts the value from player's wealth.

```
547 FOR LP=1 TO SN+SS+SE+SW+SU+SD:IF T
EMP(LP)=L THEN ITEM(LP)=L:TEMP(LP)=0
548 NEXT LP
550 IF IRN(L)=0 THEN IRN(L)=ROOM:ST=ST
+INT(VI(L)/1000):MONEY=MONEY-VI(L):GOT
O 50
560 PRINT "YOU'RE NOT CARRYING IT!!":G
OTO 1710
```

In the routine to LOOK and EXAMINE objects, the program checks to see if it is a valid object and displays a message if the object is in the room. If in your adventure you wish the player to see something special, additional program lines would have to be added here.

```
570 FOR L=1 TO OBS:IF TI$(1,3)=ITEM$(L
*20-19,L*20-17) THEN POP :GOTO 590
580 NEXT L:PRINT "IT LOOKS LIKE A ";TI
$;"!!":GOTO 1710
590 IF IRN(L)<>ROOM AND IRN(L)<>0 THEN
 PRINT "IT'S NOT HERE TO LOOK AT!":GOT
O 1710
600 PRINT "I SEE NOTHING SPECIAL!":GOT
O 1710
```

The routine to TOUCH and FEEL:

```
610 FOR L=1 TO OBS:IF TI$(1,3)=ITEM$(L
*20-19,L*20-17) THEN POP :GOTO 630
620 NEXT L:PRINT "IT FEELS LIKE A ";TI
$;"!":GOTO 1710
630 PRINT "IT FEELS ";:F=INT(RND(0)*6)
+1:ON F GOTO 640,650,660,670,680,690
640 PRINT "ROUGH!!":GOTO 1710
650 PRINT "SMOOTH!!":GOTO 1710
660 PRINT "WET!!":GOTO 1710
670 PRINT "DRY!!":GOTO 1710
680 PRINT "COLD!":GOTO 1710
690 PRINT "WARM!":GOTO 1710
```

The routine to OPEN and UNLOCK:

```
700 PRINT "IF YOU HAVE THE CORRECT OBJ
ECT,         THE ITEM OPENS AUTOMATICALL
Y!":GOTO 1710
```

In the routine to FIGHT or KILL, the program checks to see if the monster is in the room, clears the screen, sets the color to red, and then randomly chooses a monster for the player to fight.

```
750 IF M THEN 770
760 PRINT "THERE IS NOTHING HERE TO FI
GHT!":GOTO 1710
770 PRINT "[esc ctrl clear]":POSITION
2,4:PRINT "THE MONSTER IS A ";:POKE 71
0,66:POKE 712,66:POKE 752,1:F=INT(RND(
0)*4)+1
775 ON F GOTO 780,790,800,810
780 PRINT "HOBGOBLIN!!":MST=INT(RND(0)
*80)+5:GOTO 850
790 PRINT "DRAGON!!":MST=INT(RND(0)*10
0)+30:GOTO 850
800 PRINT "ZOMBIE!!":MST=INT(RND(0)*40
)+10:GOTO 850
810 PRINT "GIANT SPIDER!":MST=INT(RND(
0)*80)+20:GOTO 850
```

The program displays monster's and player's strengths, allows the player to fight by using the space bar, creates sound, displays results, subtracts the hit points, and checks to see if the monster or the player is dead.

```
850 POSITION 1,8:PRINT "MONSTER STRENG
TH ";MST;"  ":POSITION 1,10:PRINT "YOU
R STRENGTH ";ST;"  ";:POKE 764,255
860 POSITION 12,19:PRINT "[inverse] PU
SH SPACE BAR ":ML=INT(RND(0)*10):POSIT
ION 5,14:PRINT "MONSTER LOSES ";ML
870 YL=INT(RND(0)*6):POSITION 24,14:PR
INT "YOU LOSE ";YL
880 IF PEEK(764)<>255 THEN 900
890 POSITION 12,19:PRINT " PUSH SPACE
BAR ":FOR T=1 TO 10:NEXT T:GOTO 860
900 COL=PEEK(710):POKE 710,0:POKE 712,
0:MST=MST-ML:ST=ST-YL:SOUND 0,100,2,10
:FOR T=1 TO 50:NEXT T:SOUND 0,0,0,0
```

If the monster died, the program plays a song.

```
910 IF MST<=0 THEN PRINT "    YOU KILL
ED THE CREATURE!!";:MON(M)=0:M=0:GOSUB
 2500:POKE 752,0:GOTO 45
2500 RESTORE 3200:FOR L=1 TO 6:READ SO
,DELAY:SOUND 0,SO,10,10:FOR T=1 TO DEL
AY:NEXT T:NEXT L:SOUND 0,0,0,0:RETURN
3200 DATA 96,10,72,10,57,10,48,20,57,1
0,48,40
```

If the player died, it displays wealth collected and gives the player a chance to play again.

```
920 IF ST<=0 THEN 1800
1800 GRAPHICS 18:POSITION 5,1:PRINT #6
;"[inverse]the monster":POSITION 5,2:P
RINT #6;"[inverse]killed you"
1810 LN=INT((20-LEN(NA$))/2):POSITION
LN,4:PRINT #6;NA$:GOSUB 2400
1815 POSITION 1,6:PRINT #6;"TREASURES
$";MONEY
1820 POSITION 2,8:PRINT #6;"[inverse]T
RY AGAIN? (Y/N)":OPEN #2,4,0,"K:"
1830 GET #2,K:IF K<>78 AND K<>89 THEN
1830
1840 CLOSE #2:IF K=89 THEN RUN
1850 POSITION 1,9:PRINT #6;"THANKS FOR
 PLAYING!":FOR L=1 TO 200:NEXT L:END
```

If nobody died, it loops through the fight routine again.

```
930 POKE 710,COL:POKE 712,COL:GOTO 850
```

The routine for INVENTORY:

```
1000 PRINT "[esc shift clear]":PRINT "
[inverse] YOU ARE CARRYING: - - VALUED
 AT:     "1010 PRINT :FOR L=1 TO OBS:IF
 IRN(L)=0 THEN PRINT ITEM$(L*20-19,L*2
0);:PRINT "[esc tab/3 spaces]$";VI(L)
1020 NEXT L:POSITION 5,22:PRINT "TOUCH
```

```
 [inverse] SPACE BAR [inverse off]TO C
ONTINUE":OPEN #2,4,0,"K:"
1030 GET #2,K:IF K<>32 THEN 1030
1040 CLOSE #2:GOTO 50
```

The last of the programs is finally complete. In the process of keying them in and playing them, hopefully you have enjoyed them and have learned something to help you in your efforts to make your own adventure programs.

```
(inverse SPACE BAR (shift-key 0))TO C
ONTINUE":PRINT #2,4,0,"RUN"
1030 GET #2,K:IF K<>32 THEN 1030
1040 CLOSE #2:GOTO 90
```

The last of the programs is finally completed. In the course of keying them in and playing them, hopefully, you have enjoyed them and have learned something to help you in your efforts to make your own adventure programs.

APPENDIX A

Program Listings

Following are the complete listings of each program presented. These listings contain characters not shown on the Atari keyboard. They are special characters which use the Control, Escape, and Atari Logo (inverse) keys. Below is a list of these characters and the keystrokes needed to obtain them.

```
♥ --- CTRL ,
├ --- CTRL A
▕ --- CTRL B
┘ --- CTRL C
┤ --- CTRL D
┐ --- CTRL E
╱ --- CTRL F
╲ --- CTRL G
◢ --- CTRL H
▗ --- CTRL I
◣ --- CTRL J
▝ --- CTRL K
▘ --- CTRL L
▔ --- CTRL M
▁ --- CTRL N
▖ --- CTRL O
♣ --- CTRL P
┌ --- CTRL Q
─ --- CTRL R
┼ --- CTRL S
● --- CTRL T
▄ --- CTRL U
▏ --- CTRL V
┬ --- CTRL W
┴ --- CTRL X
▌ --- CTRL Y
└ --- CTRL Z
♦ --- CTRL .
♠ --- CTRL ;
␛ --- ESC ESC
↑ --- ESC CTRL UP-ARROW
↓ --- ESC CTRL DOWN-ARROW
← --- ESC CTRL LEFT-ARROW
→ --- ESC CTRL RIGHT-ARROW
↖ --- ESC SHIFT CLEAR
◀ --- ESC BACK S
```

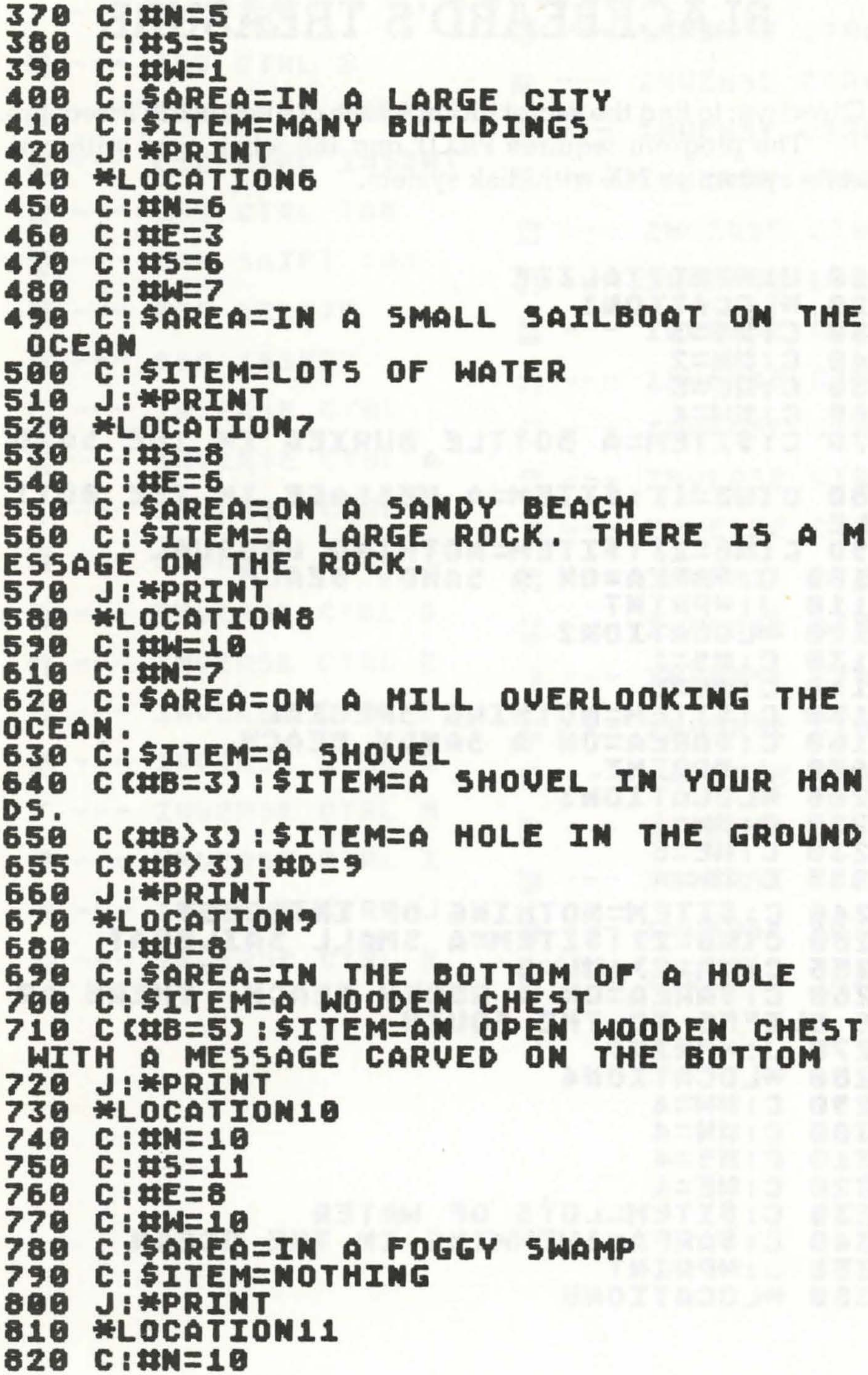
```
▸ --- ESC TAB
⬉ --- ESC CTRL 2
◀ --- ESC CTRL BACK-S
▶ --- ESC CTRL INSERT
⇤ --- ESC CTRL TAB
⇥ --- ESC SHIFT TAB
⇧ --- ESC DELETE
⇩ --- ESC INSERT
♥ --- INVERSE CTRL ,
┣ --- INVERSE CTRL A
┃ --- INVERSE CTRL B
┛ --- INVERSE CTRL C
┫ --- INVERSE CTRL D
┓ --- INVERSE CTRL E
╱ --- INVERSE CTRL F
╲ --- INVERSE CTRL G
◤ --- INVERSE CTRL H
▛ --- INVERSE CTRL I
◥ --- INVERSE CTRL J
▙ --- INVERSE CTRL K
▟ --- INVERSE CTRL L
▇ --- INVERSE CTRL M
▀ --- INVERSE CTRL N
▜ --- INVERSE CTRL O
♣ --- INVERSE CTRL P
┏ --- INVERSE CTRL Q
═ --- INVERSE CTRL R
╋ --- INVERSE CTRL S
◙ --- INVERSE CTRL T
▀ --- INVERSE CTRL U
▊ --- INVERSE CTRL V
┳ --- INVERSE CTRL W
┻ --- INVERSE CTRL X
▐ --- INVERSE CTRL Y
┗ --- INVERSE CTRL Z
♠ --- INVERSE CTRL .
♦ --- INVERSE CTRL ;
█ --- INVERSE SPACE
▆ --- INVERSE SHIFT -
‖ --- INVERSE SHIFT =
```

BLACKBEARD'S TREASURE

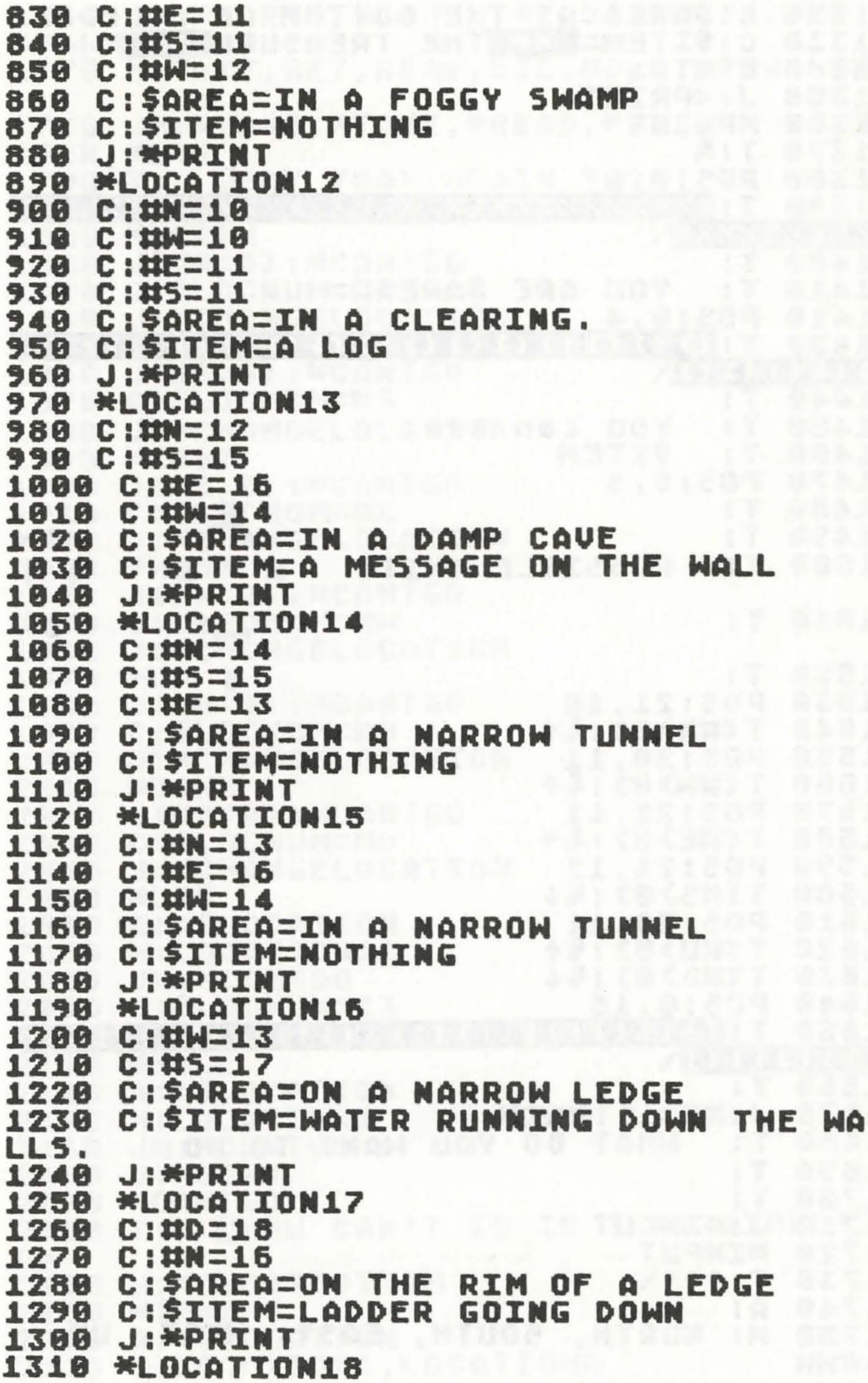
Objective: to find the secret hiding place of Blackbeard's treasure.

The program requires PILOT and 16K when used with cassette system or 24K with disk system.

```
10 U:*INITIALIZE
20 *LOCATION1
30 C:#S=3
40 C:#N=2
50 C:#E=5
60 C:#W=4
70 C:$ITEM=A BOTTLE BURIED IN THE SAND

80 C(#B=1):$ITEM=A MESSAGE IN THE BOTT
LE
90 C(#B=2):$ITEM=NOTHING UNUSUAL
100 C:$AREA=ON A SANDY BEACH
110 J:*PRINT
120 *LOCATION2
130 C:#S=1
140 C:#E=5
150 C:$ITEM=NOTHING SPECIAL
160 C:$AREA=ON A SANDY BEACH
170 J:*PRINT
200 *LOCATION3
220 C:#N=1
230 C:#E=5
235 C:#W=4
240 C:$ITEM=NOTHING OF INTEREST
250 C(#B=2):$ITEM=A SMALL SAILBOAT
255 C(#B=2):#W=6
260 C:$AREA=ON A ROCKY BEACH. THERE AR
E CLIFFS TO THE SOUTH
270 J:*PRINT
280 *LOCATION4
290 C:#W=4
300 C:#N=4
310 C:#S=4
320 C:#E=1
330 C:$ITEM=LOTS OF WATER
340 C:$AREA=SWIMMING IN THE OCEAN
350 J:*PRINT
360 *LOCATION5
```



```
370 C:#N=5
380 C:#S=5
390 C:#W=1
400 C:$AREA=IN A LARGE CITY
410 C:$ITEM=MANY BUILDINGS
420 J:*PRINT
440 *LOCATION6
450 C:#N=6
460 C:#E=3
470 C:#S=6
480 C:#W=7
490 C:$AREA=IN A SMALL SAILBOAT ON THE
 OCEAN
500 C:$ITEM=LOTS OF WATER
510 J:*PRINT
520 *LOCATION7
530 C:#S=8
540 C:#E=6
550 C:$AREA=ON A SANDY BEACH
560 C:$ITEM=A LARGE ROCK. THERE IS A M
ESSAGE ON THE ROCK.
570 J:*PRINT
580 *LOCATION8
590 C:#W=10
610 C:#N=7
620 C:$AREA=ON A HILL OVERLOOKING THE
OCEAN.
630 C:$ITEM=A SHOVEL
640 C(#B=3):$ITEM=A SHOVEL IN YOUR HAN
DS.
650 C(#B>3):$ITEM=A HOLE IN THE GROUND
655 C(#B>3):#D=9
660 J:*PRINT
670 *LOCATION9
680 C:#U=8
690 C:$AREA=IN THE BOTTOM OF A HOLE
700 C:$ITEM=A WOODEN CHEST
710 C(#B=5):$ITEM=AN OPEN WOODEN CHEST
 WITH A MESSAGE CARVED ON THE BOTTOM
720 J:*PRINT
730 *LOCATION10
740 C:#N=10
750 C:#S=11
760 C:#E=8
770 C:#W=10
780 C:$AREA=IN A FOGGY SWAMP
790 C:$ITEM=NOTHING
800 J:*PRINT
810 *LOCATION11
820 C:#N=10
```



```
830 C:#E=11
840 C:#S=11
850 C:#W=12
860 C:$AREA=IN A FOGGY SWAMP
870 C:$ITEM=NOTHING
880 J:*PRINT
890 *LOCATION12
900 C:#N=10
910 C:#W=10
920 C:#E=11
930 C:#S=11
940 C:$AREA=IN A CLEARING.
950 C:$ITEM=A LOG
960 J:*PRINT
970 *LOCATION13
980 C:#N=12
990 C:#S=15
1000 C:#E=16
1010 C:#W=14
1020 C:$AREA=IN A DAMP CAVE
1030 C:$ITEM=A MESSAGE ON THE WALL
1040 J:*PRINT
1050 *LOCATION14
1060 C:#N=14
1070 C:#S=15
1080 C:#E=13
1090 C:$AREA=IN A NARROW TUNNEL
1100 C:$ITEM=NOTHING
1110 J:*PRINT
1120 *LOCATION15
1130 C:#N=13
1140 C:#E=16
1150 C:#W=14
1160 C:$AREA=IN A NARROW TUNNEL
1170 C:$ITEM=NOTHING
1180 J:*PRINT
1190 *LOCATION16
1200 C:#W=13
1210 C:#S=17
1220 C:$AREA=ON A NARROW LEDGE
1230 C:$ITEM=WATER RUNNING DOWN THE WA
LLS.
1240 J:*PRINT
1250 *LOCATION17
1260 C:#D=18
1270 C:#N=16
1280 C:$AREA=ON THE RIM OF A LEDGE
1290 C:$ITEM=LADDER GOING DOWN
1300 J:*PRINT
1310 *LOCATION18
```

```
1320 C:$AREA=AT THE BOTTOM OF A LEDGE
1330 C:$ITEM= ** THE TREASURE **
1340 C:#T=1
1350 J:*PRINT
1360 *PRINT
1370 T:↖
1380 POS:0,0
1390 T:▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨
▨▨▨▨▨▨▨▨▨\
1400 T:
1410 T:  YOU ARE $AREA
1420 POS:0,4
1430 T:() () () () () () () () () () () () () () () (
) () () () ()\
1440 T:
1450 T:  You can see:
1460 T:  $ITEM
1470 POS:0,9
1480 T:                         N
1490 T:                        /‾\          ┌U┐
1500 T:  POSSIBLE EXITS      W     E        | |

1510 T:                        \_/          └D┘

1520 T:                         S
1530 POS:21,10
1540 T(#N>0):␛↑
1550 POS:20,11
1560 T(#W>0):␛←
1570 POS:22,11
1580 T(#E>0):␛→
1590 POS:21,12
1600 T(#S>0):␛↓
1610 POS:29,11
1620 T(#U>0):␛↑
1630 T(#D>0):␛↓
1640 POS:0,15
1650 T:[] [] [] [] [] [] [] [] [] [] [] [] [] [] [] [
] [] [] [] []\
1660 T:
1670 J(#T=1):*WIN
1680 T:  WHAT DO YOU WANT TO DO:
1690 T:
1700 T:
1710 J:*INPUT
1720 *INPUT
1730 T:↑→→\
1740 A:
1750 M: NORTH, SOUTH, EAST, WEST, UP,
DOWN
```

```
1760 JM:*NORTH,*SOUTH,*EAST,*WEST,*UP,
*DOWN
1770 M:TAKE,GET,READ,DIG,HOLE,OPEN,LOG

1780 JM:*TAKE,*TAKE,*READ,*DIG,*HOLE,*
OPEN,*LOG
1790 T:  ◘SAY THAT AGAIN ? ? ? ?
1800 J:*CLEARBOTTOM
1810 *NORTH
1820 J(#N=0):*CANTGO
1830 C:$LOCNUM=#N
1840 J:*CHANGELOCATION
1850 *SOUTH
1860 J(#S=0):*CANTGO
1870 C:$LOCNUM=#S
1880 J:*CHANGELOCATION
1890 *EAST
1900 J(#E=0):*CANTGO
1910 C:$LOCNUM=#E
1920 J:*CHANGELOCATION
1930 *WEST
1940 J(#W=0):*CANTGO
1950 C:$LOCNUM=#W
1960 J:*CHANGELOCATION
1970 *UP
1980 J(#U=0):*CANTGO
1990 C:$LOCNUM=#U
2000 J:*CHANGELOCATION
2010 *DOWN
2020 J(#D=0):*CANTGO
2030 C:$LOCNUM=#D
2040 J:*CHANGELOCATION
2050 *LOG
2060 A:=$LOCATION
2070 M:LOCATION12
2080 JN:*CANTDO
2090 C:$LOCNUM=13
2100 J:*CHANGELOCATION
2110 *HOLE
2120 A:=$LOCATION
2130 M:LOCATION8
2140 JN:*CANTDO
2150 J:*DOWN
2170 *CANTGO
2180 T:  ◘YOU CAN'T GO IN THAT DIRECTI
ON
2190 J:*CLEARBOTTOM
2200 *TAKE
2210 A:=$LOCATION
2220 M:LOCATION1,LOCATION8
```

```
2230 JM:*TAKE1,*TAKE2
2240 T:  ◘THERE IS NOTHING HERE TO TAK
E!
2250 J:*CLEARBOTTOM
2260 *TAKE1
2270 J(#B<>0):*HAVE
2280 C:#B=1
2290 U:*OK
2300 J:*LOCATION1
2310 *TAKE2
2320 J(#B<>2):*HAVE
2330 C:#B=3
2340 U:*OK
2350 J:*LOCATION8
2355 *READ
2360 J(#B=1):*MESSAGE1
2370 A:=$LOCATION
2380 M:ON7 ,ON9 ,N13 ,
2390 UY:*OK
2400 TY:↑↑↑
2410 JM:*MESSAGE2,*MESSAGE3,*MESSAGE4
2420 J:*CANTREAD
2430 *MESSAGE1
2440 U:*OK
2450 C:#B=2
2460 T:↑↑   FIND A BOAT
2470 T:      SAIL IT WEST
2480 T:       THERE'S A TREASURE THERE
2490 T:        ONE OF THE BEST
2495 PA:240
2500 J:*LOCATION1
2510 *MESSAGE2
2520 T:    FIND A SHOVEL
2530 T:      DIG IN THE GROUND
2540 T:       FOR THERE A CLUE
2550 T:        MAY BE FOUND
2560 J:*RESET
2570 *MESSAGE3
2580 J(#B<5):*CANTREAD
2590 T:    WHEN YOU CAN SEE
2600 T:     THROUGH THE FOG
2610 T:       CRAWL INTO
2620 T:        THE HOLLOW LOG
2630 J:*RESET
2640 *MESSAGE4
2650 T: LOOK AROUND THE CAVE
2660 T:     SO DIM
2670 T:       AND THEN CLIMB ONTO
2680 T:        THE RIM
2690 J:*RESET
```

```
2700 *OPEN
2710 A:=$LOCATION
2720 M:LOCATION9
2730 JN:*CANTDO
2740 J(#B<>4):*HAVE
2750 C:#B=5
2760 U:*OK
2770 J:*LOCATION9
2780 *DIG
2790 A:=$LOCATION
2800 M:LOCATION8
2810 JN:*CANTDO
2820 J(#B<>3):*CANTDO
2830 J(#B>=4):*HAVE
2840 C:#B=4
2850 U:*OK
2860 J:*LOCATION8
2870 *CANTREAD
2880 T:  ⬉THERE IS NOTHING HERE TO REA
D!
2885 T(#B=2):  THE NOTE TURNED TO DUST
!↑
2890 J:*CLEARBOTTOM
2900 *CANTDO
2910 T:  ⬉YOU CAN'T DO THAT!
2920 J:*CLEARBOTTOM
2930 *HAVE
2940 T:  YOU ALREADY HAVE!
2950 J:*CLEARBOTTOM
2960 *OK
2970 T:  OKAY!
2980 PA:60
2990 E:
3000 *CHANGELOCATION
3010 C:#N=0
3020 C:#S=0
3030 C:#E=0
3040 C:#W=0
3050 C:#U=0
3060 C:#D=0
3070 C:$LOCATION=LOCATION$LOCNUM
3080 A:=$LOCATION
3090 M:ON1 ,ON2 ,ON3 ,ON4 ,ON5 ,ON6 ,O
N7 ,
3100 JM:*LOCATION1,*LOCATION2,*LOCATIO
N3,*LOCATION4,*LOCATION5,*LOCATION6,*L
OCATION7
3120 M:ON8 ,ON9 ,N10 ,N11 ,N12 ,N13 ,
3130 JM:*LOCATION8,*LOCATION9,*LOCATIO
N10,*LOCATION11,*LOCATION12,*LOCATION1
```

```
3
3150 M:N14 ,N15 ,N16 ,N17 ,N18 ,
3160 JM:*LOCATION14,*LOCATION15,*LOCAT
ION16,*LOCATION17,*LOCATION18
3170 E:
3180 *RESET
3190 PA:240
3200 T:↑↑↑
3210 J:*CLEARBOTTOM
3220 *CLEARBOTTOM
3230 PA:120
3240 T:↑↑↓↓↓↓↓↓↓
3250 J:*INPUT
3260 *INITIALIZE
3270 C:@B1373=0
3280 C:@B1374=2
3290 WRITE:S,
3300 POS:4,3
3310 WRITE:S,blackbeard's
3320 POS:5,4
3330 WRITE:S,treasure
3340 POS:3,9
3350 WRITE:S,BY JACK HARDY
3360 SO:1,13
3370 PA:30
3380 SO:2,14
3390 PA:30
3400 SO:4,16
3410 PA:60
3420 SO:1,13
3430 PA:30
3440 SO:0
3450 PA:120
3460 C:#B=0
3470 C:#T=0  [TREASURE STATUS
3471 C:#N=0  [LOCATION POINTER
3472 C:#S=0
3473 C:#E=0
3474 C:#W=0
3475 C:#D=0
3476 C:#U=0
3480 C:$LOCATION=LOCATION1
3490 GR:QUIT
3500 C:@B710=0  [SETS SCREEN COLOR TO
BLACK
3510 C:@B82=0 [SET LEFT MARGIN
3520 E:
3540 *WIN
3550 POS:9,20
3560 T:***YOU FOUND IT!***
```

```
3570 C:#T=#T+1
3580 C:#A=?\30
3590 SO:#A
3600 PA:10
3610 POS:9,20
3620 T:***YOU FOUND IT!***
3630 C:#A=?\30
3640 SO:#A
3650 PA:10
3660 J(#T<10):*WIN
3670 E:
```

HALLS OF DEATH

Objective: to explore the Halls of Death, to find an invisible Golden Treasure Chest, and to return to room number 1. More complete instructions are included in the program's listing.

The program requires Atari Microsoft BASIC and 32K with cassette or 48K with disk drive.

```
10 GOTO 2960
20 FOR T=LEN(M$) TO 1 STEP-1:PRINT MID
$(M$,T,1);:SOUND 0,ASC(MID$(M$,T,1)),1
0,10:NEXT:SOUND 0,0,0,0:RETURN
30 FOR T=LEN(C3$) TO 1 STEP-1:C4$=C4$+
MID$(C3$,T,1):NEXT:RETURN
40 PRINT AT(2,11);"⇧⇧⇧⇧⇧⇧⇧⇧⇧":RETURN
50 L$=" A HALLWAY. THERE IS A METAL DO
OR       WEST AND AN IRON GATE EAST.":V
1$=" THE DOOR IS LOCKED."
60 V2$=" THE GATE IS LOCKED.":S=8:GOTO
 660
70 L$=" THE WEST END OF A FOUL SMELLIN
G       ROOM. AN IRON GATE IS WEST."
80 V1$=" MY WAY EAST IS BLOCKED BY A P
OOL OF    LIQUID.":V2$=" THE GATE IS LO
CKED.":GOTO 680
90 L$=" THE EAST END OF A FOUL SMELLIN
G       ROOM."
100 V1$=" MY WAY WEST IS BLOCKED BY A
POOL OF   LIQUID.":S=10:GOTO 690
110 L$=" THE BOTTOM OF A PIT. SOUTH I
SEE A     DOOR WITH NO HANDLE. ON A TAB
LE IS     A BOOK AND PEN.":GOTO 860
120 L$=" A SMALL ROOM.":V1$=" AN X IS
MARKED ON THE FLOOR.":E=8:S=14:GOTO 71
0
130 L$=" A HALLWAY. A MAN'S BODY IS ON
 THE         FLOOR. CHALK IS IN HIS HAND A
ND THERE IS WRITING ON THE WALL."
140 N=3:E=9:W=7:V1$=" A GIANT BLOCKS A
 DOORWAY SOUTH.":GOTO 770
150 L$=" A NARROW PASSAGE. FIRE BLOCKS
 MY       WAY SOUTH.":W=8:GOTO 870
160 L$=" A NARROW PASSAGE. A FOUL SMEL
L COMES  FROM THE NORTH.":N=5:E=11:GOT
O 870
170 L$=" A NARROW PASSAGE. AN IRON DOO
```

```
R LEADS  WEST.":V1$=" THE DOOR IS LOCK
ED.":N=6:S=17:GOTO 720
180 L$=" A LARGE ROOM. IN THE CORNER I
S A POOL OF WATER.":V1$=" THERE IS A D
RAGON IN THE ROOM.":E=13:S=18:GOTO 730

190 L$=" A SLOPING PASSAGEWAY.":W=12:E
=14:GOTO 870
200 L$=" A NARROW PASSAGEWAY. THERE IS
         GRAFITTI ON THE WALL.":W=13:N
=7:GOTO 870
210 L$=" A LARGE ROOM.":V1$=" I'M OUTS
IDE A GOLDEN CAGE. INSIDE I   SEE THE
MAGIC RING.":N=8:S=21:GOTO 650
220 L$=" A LARGE CAVERN.":N=9:GOTO 870

230 L$=" A HALLWAY WITH AN ARROW CARVE
D IN     THE WALL POINTING SOUTH.":N=1
1:S=24:GOTO 740
240 L$=" A LARGE PASSAGEWAY":N=12:E=19
:S=26:GOTO 870
250 L$=" A TORTURE CHAMBER. A SMALL CE
LL IS IN THE NORTH CORNER. IN THE CELL
 IS A    SIGN.":W=18:GOTO 870
260 L$=" A LARGE ROOM WITH ROUGH-CUT W
ALLS.    THERE ARE STAIRS GOING UP.":U
=28:E=21:GOTO 870
270 L$=" A T-SHAPED PASSAGE WITH A DOO
R        NORTH.":V1$=" THE DOORWAY IS
GUARDED BY A GIANT."
280 W=20:E=22:GOTO 760
290 L$=" A T-SHAPED PASSAGEWAY.":S=29:
W=21:E=23:GOTO 870
300 L$=" A CLOVER-SHAPED ROOM.":W=22:E
=24:GOTO 870
310 L$=" A STAR-SHAPED ROOM.":S=31:W=2
3:N=17:GOTO 870
320 L$=" A SMALL DUSTY ROOM. A DOOR IS
 ON THE  EAST WALL."
330 V1$=" THE DOOR IS CLOSED.":V2$=" T
HERE IS A SILVER DOOR IN THE FLOOR   W
ITH A LADDER GOING DOWN.":GOTO 780
340 L$=" A NARROW HALLWAY. THERE IS A
DOORWAY  SOUTH.":V1$=" THE DOOR IS CLO
SED.":N=18:E=27:GOTO 800
350 L$=" A LARGE LIBRARY. THERE ARE ST
AIRS     GOING DOWN.":W=26:D=28:GOTO 8
70
360 L$=" A STAIRWELL.":U=27:D=20:GOTO
870
```

```
370 L$=" A LABORATORY.":V1$=" THERE IS
 A WIZARD WITH A WAND.":N=22:E=30:GOTO
 810
380 L$=" A LARGE RECTANGULAR ROOM.":W=
29:E=31:GOTO 870
390 L$=" A LARGE CAVERN. WATER IS RUNN
ING       DOWN THE WALLS AND HAS FORMED
 A LARGE LAKE."
400 V1$=" THERE IS SOMETHING MOVING IN
 THE       LAKE.":W=30:N=24:GOTO 820
410 L$=" A LARGE LAKE. TO THE SOUTH I
SEE A     TUNNEL.":W=31:S=50:GOTO 870
420 L$=" A LARGE SQUARE ROOM WITH A LO
W         CEILING.":N=26:GOTO 870
430 L$=" A VERY SMALL ROOM.":W=17:GOTO
 870
440 L$=" A LARGE RED ROOM.":S=38:N=46:
E=36:W=47:IF TT>1 THEN V1$=" THERE ARE
 FOOTPRINTS IN THE DUST."
450 GOTO 870
460 L$=" A NARROW PASSAGE.":W=35:E=37:
GOTO 870
470 L$=" AN L-SHAPED PASSAGE.":S=40:W=
36:GOTO 870
480 L$=" AN L-SHAPED PASSAGE.":W=41:N=
35:GOTO 870
490 L$=" AN L-SHAPED PASSAGE.":E=40:S=
42:GOTO 870
500 L$=" AN L-SHAPED PASSAGE.":N=37:W=
39:GOTO 870
510 L$=" AN L-SHAPED PASSAGE.":S=42:E=
38:GOTO 870
520 L$=" AN L-SHAPED PASSAGE.":TT=TT+1
:E=43:N=41:GOTO 870
530 L$=" A NARROW PASSAGE.":TT=TT+1:W=
42:E=44:GOTO 870
540 L$=" A SMALL ROOM.":N=39:W=43:GOTO
 870
 550 L$=" A SMALL JAIL CELL. THERE IS A
  SIGN ON THE WALL. THROUGH THE BARS I
 SEE A     TORTURE CHAMBER.":GOTO 870
 560 L$=" A NARROW PASSAGE. THERE IS A
 PIT       NORTH.":S=35:GOTO 870
 570 T1=0:FOR T=1 TO 22:IF I(T)=0 THEN
 T1=T:T=22
 580 NEXT:IF T1>0 THEN R=45:GOTO 2880
 590 L$=" A VERY SMALL ROOM.":E=35:GOTO
  870
 600 L$=" THE HALLS OF DEATH. IN FRONT
 OF ME     IS A NARROW PIT. BEYOND THE P
```

```
IT IS   A PASSAGEWAY.":GOTO 870
610 L$=" A LARGE DRAB ROOM.":E=3:GOTO
870
620 L$=" A LONG WINDING TUNNEL. THE FL
OOR IS   DAMP. THERE IS GRAFITTI ON TH
E WALL.":W=51:N=32:GOTO 870
630 L$=" THE WEST END OF A TUNNEL.":V1
$=" THERE IS A LADDER LEADING UP TO A
  SILVER DOOR."
640 V2$=" THE SILVER DOOR IS LOCKED.":
E=50:GOTO 830
650 IF R=15 AND D8=1 THEN V1$=" I'M IN
 A GOLDEN CAGE."
660 IF R=3 AND D1=1 THEN V1$=" THE DOO
R IS OPEN.":W=2
670 IF R=3 AND G1=1 THEN V2$=" THE GAT
E IS OPEN.":E=4
680 IF R=4 AND G1=1 THEN V2$=" THE GAT
E IS OPEN.":W=3
690 IF R=5 AND I(21)=5 OR R=5 AND I(21
)=4 THEN V1$=" THERE IS A BOARD ACROSS
 THE LIQUID.":W=4
700 IF R=4 AND I(21)=5 OR R=4 AND I(21
)=4 THEN V1$=" THERE IS A BOARD ACROSS
 THE LIQUID.":E=5
710 IF R=7 AND D4=1 THEN V1$=" THERE I
S A HOLE IN THE FLOOR."
720 IF R=11 AND D2=1 THEN V1$=" THE IR
ON DOOR IS OPEN.":W=10
730 IF R=12 AND M(1)=0 THEN V1$=""
740 IF R=17 AND KH=1 THEN V1$=" THERE
IS A BRASS KEYHOLE IN THE      WALL."

750 IF R=17 AND KH=2 THEN V1$=" THERE
IS AN OPEN PANEL ON EAST WALL."
760 IF R=21 AND M(2)=0 THEN V1$=" THE
DOOR IS OPEN AND UNGUARDED.":N=15
770 IF R=8 AND M(2)=0 THEN V1$=" THE D
OOR IS OPEN AND UNGUARDED.":S=15
780 IF R=25 AND D3=1 THEN V1$=" THE DO
OR IS OPEN.":E=26
790 IF R=25 AND LD=1 THEN V2$=""
800 IF R=26 AND D5=1 THEN V1$=" THE DO
OR IS OPEN.":S=33
810 IF R=29 AND I(7)=99 THEN V1$=""
820 IF R=31 AND M(3)=0 THEN V1$=""
830 IF R=51 AND LD=1 THEN V1$=" THERE
IS A SILVER DOOR ABOVE ME.":U=0
840 IF R=51 AND D6=1 THEN V2$=" THE SI
LVER DOOR IS OPEN.":U=25
```

```
850 IF I(13)=0 THEN I$(18)="TSEHC ERUS
AERT NEDLOG":I(18)=44
860 IF R=6 AND D7=1 THEN V1$=" THE DOO
R IS OPEN":S=11
870 SETCOLOR 6,R,2:SETCOLOR 8,R,2:PRIN
T AT(1,0);"↖                   I AM IN: 
                    "
880 PRINT L$:PRINT V1$:PRINT V2$
890 PRINT AT(2,7);"                EXITS
 ARE:                    "
900 IF N<>0 THEN PRINT " NORTH";
910 IF S<>0 THEN PRINT " SOUTH";
920 IF E<>0 THEN PRINT " EAST";
930 IF W<>0 THEN PRINT " WEST";
940 IF U<>0 THEN PRINT " UP";
950 IF D<>0 THEN PRINT " DOWN";
960 PRINT AT(2,10);"               ITEMS
 I SEE:                    ":IT$=""
970 FOR T=1 TO 22:IF I(T)=R THEN IT$=I
T$+" ,"+I$(T)
980 NEXT:IF IT$<>"" THEN IT$=RIGHT$(IT
$,LEN(IT$)-2)
990 IF IT$="" THEN IT$="  GNIHTON"
1000 M$=IT$:PRINT AT(3,11);:GOSUB 20:G
OTO 1020
1010 GOSUB 20:GOSUB 3490
1020 POKE 752,0:PRINT AT(2,16);"     
     BY YOUR COMMAND:             ":INPUT
 CO$:POKE 752,1:C1$="":C2$="":D3=0
1030 IF LEN(CO$)>1 THEN 1210
1040 IF CO$="" THEN M$="!EM NODRAP◳":G
OTO 1010
1050 T1=0:FOR T=1 TO 6:IF CO$=DI$(T) T
HEN T1=T:T=6
1060 NEXT:IF T1<1 THEN M$="!!TAHT DNAT
SREDNU T'NDID I◳":GOTO 1010
1070 ON T1 GOTO 1080,1100,1120,1140,11
60,1180
1080 IF N=0 THEN 1200
1090 R=N:GOTO 2840
1100 IF S=0 THEN 1200
1110 R=S:GOTO 2840
1120 IF E=0 THEN 1200
1130 R=E:GOTO 2840
1140 IF W=0 THEN 1200
1150 R=W:GOTO 2840
1160 IF U=0 THEN 1200
1170 R=U:GOTO 2840
1180 IF D=0 THEN 1200
1190 R=D:GOTO 2840
```

```
1200 M$="!!!NOITCERID TAHT NI OG T'NAC
 I▣":GOTO 1010
1210 FOR T=1 TO LEN(CO$):IF MID$(CO$,T
,1)=" " THEN C2$=RIGHT$(CO$,LEN(CO$)-T
):T=LEN(CO$)
1220 NEXT:C1$=CO$:C4$="":C3$=C1$:GOSUB
 30:C1$=RIGHT$(C4$,3):C4$="":C3$=C2$:G
OSUB 30:C2$=RIGHT$(C4$,3)
1225 IF C1$=" OG" THEN C1$="OG"
1230 T1=0:FOR T=1 TO 31:IF C1$=VOC$(T)
 THEN T1=T:T=31
1240 NEXT:IF T1=0 THEN 1440
1250 IF T1<17 THEN ON T1 GOTO 1450,155
0,1550,1620,1680,1720,1720,1720,1850,1
850,1810,1810,1300,1950,2220,2280
1260 ON T1-16 GOTO 2330,2330,2460,2460
,2540,2540,2580,1310,1320,2630,2630,13
30,1340,2770,2770
1300 M$=".GNIDAER YB TOL A NRAEL ELPOE
P":GOTO 1010
1310 IF I(14)=0 AND DK=1 THEN DK=0:GOT
O 2840
1315 GOTO 1440
1320 IF I(1)=0 AND I(14)=0 AND DK=0 TH
EN DK=1:I$(14)="HCROT GNIBRUB":GOTO 28
40
1325 GOTO 1440
1330 IF RIGHT$(I$(10),8)="FO SSALG" TH
EN I$(10)="SSALG":GOTO 2680
1440 M$="!!TNAW UOY TAHW DNATSREDNU T'
NOD I▣":GOTO 1010
1450 IF C2$="DAL" AND I(19)=99 AND R=5
1 AND D6=1 THEN R=25:GOTO 2840
1460 IF C2$="KAL" AND R=31 AND M(3)=1
THEN M$="!!GEL YM TA GNILBBIN SI GNIHT
EMOS":GOSUB 20:GOSUB 3490:GOTO 2910
1470 IF C2$="NUT" AND R=32 THEN R=50:G
OTO 2840
1480 IF C2$="KAL" AND R=31 THEN R=32:D
K=0:GOTO 2840
1490 IF C2$="DAL" AND R=6 AND I(19)=R
THEN R=1:GOTO 2840
1500 IF C2$="WOD" AND R=1 OR C2$="WOD"
 AND R=46 THEN R=6:GOTO 2840
1510 IF C2$="RIF" AND I(18)=99 AND R=9
 THEN R=16:GOTO 2840
1520 T1=0:FOR T=1 TO 6:IF RIGHT$(C2$,1
)=DI$(T) THEN T1=T
1530 NEXT:IF T1>0 THEN 1070
1540 M$="? ? ? ? ? EREHW OG▣":GOTO 101
```

```
0
1550 IF C2$="OOD" AND R=11 AND I(4)=0
THEN D2=1:GOTO 2880
1560 IF (C2$="TAG" AND R=3 AND I(4)=0)
 OR (C2$="TAG" AND R=4 AND I(4)=0) THE
N G1=1:GOTO 2880
1570 IF C2$="GAC" AND R=15 AND I(15)=0
 THEN D8=1:I(13)=15:GOTO 2880
1580 IF C2$="OOD" AND R=26 THEN D5=1:G
OTO 2840
1590 IF C2$="OOD" AND R=25 THEN D3=1:E
=26:GOTO 2880
1600 IF C2$="LIS" AND I(16)=0 AND R=51
 THEN D6=1:GOTO 2840
1610 M$="!!T'NAC I":GOTO 1010
1620 IF R=51 OR R=6 THEN M$="!!!PU HGI
H OOT S'TI":GOTO 1010
1630 IF R=5 OR R=4 THEN M$="!!!SSORCA
RAF OOT S'TI":GOTO 1010
1640 IF C2$="WOD" AND R=1 OR C2$="WOD"
 AND R=46 THEN R=6:GOTO 2840
1650 IF LEFT$(C4$,3)="TIP" AND R=1 THE
N R=46:GOTO 2840
1660 IF LEFT$(C4$,3)="TIP" AND R=46 TH
EN R=1:GOTO 2840
1670 M$="? ? ? EREHW PMUJ":GOTO 1010
1680 IF C2$="LAW" THEN M$="!OT NO DLOH
 OT GNIHTON SI EREHT":GOTO 1010
1690 IF C2$="DAL" AND I(19)=R AND R=6
THEN R=1:GOTO 2840
1700 IF C2$="DAL" AND I(19)=R AND R=51
 THEN R=25:GOTO 2840
1710 M$="!!"+C4$+" BMILC T'NAC I":GOTO
 1010
1720 T1=0:FOR T=1 TO 22:IF C2$=RIGHT$(
I$(T),3) AND I(T)=0 THEN I(T)=R:T1=T:T
=22:IN=IN-1
1730 NEXT:IF R=45 THEN 2790
1740 IF T1=7 AND R=29 THEN I(11)=29:M$
=".SRAEPPASID DNA DNAW SPORD DRAZIW":I
(7)=99:GOSUB 20:GOSUB 3510:GOTO 2880
1750 IF T1=21 THEN 2840
1760 IF T1=22 AND R=31 THEN M(3)=0:I(T
1)=99:I(5)=32:GOTO 1800
1770 IF T1=19 AND R=25 OR T1=19 AND R=
51 THEN LD=0:GOTO 2840
1780 IF T1>0 THEN GOSUB 3500:GOSUB 40:
GOTO 960
1790 M$="!TAHT EVAH T'NOD I▣":GOTO 101
0
```

```
1800 M$=".SRAEPPASID NEHT DNA  ERUTAER
C EHT STAE - SRAEPPA NOGARD A":GOSUB 2
0:GOSUB 3510:GOTO 2880
1810 IF C2$="KAL" AND R=31 AND M(3)=1
THEN M$="!THGIRLA EREHT ERUTAERC A S'E
REHT":GOTO 1840
1820 IF C2$="QIL" AND R=4 OR C2$="QIL"
 AND R=5 THEN I(6)=4:I(8)=5:GOTO 2840
1830 M$=".LAICEPS GNIHTON EES I":GOTO
1010
1840 GOSUB 20:GOSUB 3490:M$="!YRGNUH S
KOOL TI":GOTO 1010
1850 IF C2$="DAL" AND R=25 OR C2$="DAL
" AND R=51 THEN I(19)=0:LD=1:D6=0:GOSU
B 3500:GOTO 2880
1860 IF C2$="OOB" AND R=6 OR C2$="NEP"
 AND R=6 THEN M$=".LLAW EHT OT DENIAHC
 SI TI":GOTO 1010
1870 IF IN>4 THEN M$=".EROM GNIHTYNA Y
RRAC T'NAC I▣":GOTO 1010
1880 T1=0:FOR T=1 TO 22:IF C2$=RIGHT$(
I$(T),3) AND I(T)=R THEN I(T)=0:T1=T:T
=22
1890 NEXT:IF T1=6 OR T1=8 THEN 1930
1900 IF R=31 AND I(10)=0 AND C2$="TAW"
 THEN I$(10)="RETAW FO SSALG":M$=I$(10
)+" A EVAH UOY":CN=1:GOTO 101
0
1910 IF T1>0 THEN IN=IN+1:GOSUB 3500:G
OSUB 40:GOTO 960
1920 M$="!EREH TI EES T'NOD I":GOTO 10
10
1930 IF I(10)=0 THEN I$(10)="DICA FO S
SALG":I(T1)=R:CN=2:GOSUB 3500:GOSUB 40
:GOTO 960
1940 I(T1)=R:M$="!RENIATNOC ON EVAH I"
:GOTO 1010
1950 IF C2$="RCS" THEN 1959
1952 IF C2$="CID" THEN 1970
1954 IF C2$="OOB" THEN 1980
1955 IF C2$="IRW" THEN 2160
1956 IF C2$="GIS" THEN 2155
1957 IF C2$="ARG" THEN 2170
1958 M$="!!!TAHT DAER T'NAC I":GOTO 10
10
1959 IF I(3)=0 AND I(17)=0 THEN M$="'W
ORRA HCUOT' :SYAS LLORCS":GOTO 1010
1960 IF I(3)=0 THEN M$=".EM OT KEERG S
'TI":GOTO 1010
1965 GOTO 2200
```

```
1970 IF I(17)=0 THEN M$="'YRANOITCID K
EERG' :SYAS REVOC":I$(17)=I$(17)+" KEE
RG":GOTO 1010
1975 GOTO 2200
1980 IF R=6 THEN 2140
1990 IF I(9)<>0 THEN 2200
2000 IF I(13)=0 THEN 2100
2010 IF R=9 THEN M$=".TOH NEHW SPLEH D
LOC GNIHTEMOS":GOTO 1010
2020 IF R=3 THEN M$=".LATEM TAE SEVISO
RROC":GOTO 1010
2030 IF R=4 THEN M$=".DOOW FO EDAM ERA
 SEGDIRB":GOTO 1010
2040 IF R=12 THEN M$=".CIGAM ERA SNOGA
RD":GOTO 1010
2050 IF R=29 THEN M$=".TNATROPMI SI NO
ITACINUMMOC":GOTO 1010
2060 IF R=5 THEN M$=".LUOF LLEMS YLLAU
SU SLACIMEHC":I(6)=4:I(8)=5:GOSUB 20:G
OSUB 3510:GOTO 2880
2070 IF R=30 THEN M$="'EGDELWONK FO KO
OB' :SDAER REVOC":GOTO 1010
2080 IF R=25 AND I(18)=0 THEN M$=".KNI
RD GNILOOC A":GOTO 1010
2090 M$=".TI NI NOITAMROFNI YNA DNIF T
'NAC I":GOTO 1010
2100 IF Q=0 THEN M$="!YTPME SAW MOOR E
HT":Q=1:GOTO 1010
2110 IF Q=1 THEN M$="!OG UOY DNUOR DNA
 DNUOR":Q=2:GOTO 1010
2120 M$="!!WOL DNA HGIH KOOL":GOTO 101
0
2140 IF I(18)=0 THEN M$=".GNIBMILC ROF
 PLEH DEEN ELPOEP":GOTO 1010
2150 M$="'.NI NGIS TSUM STSEUG LLA':SY
AS KOOB":GOTO 1010
2155 IF R=45 OR R=19 THEN M$=".LLEC NI
 DEWOLLA TON ERA SNOISSESSOP":GOTO 101
0
2156 GOTO 2210
2160 IF R=8 THEN C4$="":C3$=NA$:GOSUB
30:NB$=C4$:M$="...EHT NI OG T'NOD "+NB
$:GOTO 1010
2165 GOTO 2210
2170 IF R=50 THEN M$="!!DEID EW DNA DE
IRT EW":GOSUB 20:GOSUB 3490:M$="EVAD &
 KCAJ DENGIS":GOTO 1010
2180 IF R=14 THEN M$=".SPLEH EGDELWONK
 FO KOOB EHT":GOTO 1010
2190 GOTO 2210
```

```
2200 M$="!!TI EVAH T'NOD I◙":GOTO 1010

2210 M$="!EREH TON S'TI":GOTO 1010
2220 IF C2$="TAW" AND R=31 OR C2$="TAW
" AND R=32 THEN M$=".GNIHSERFER SAW TA
HT !HA":GOTO 1010
2230 IF C2$="TAW" AND I$(10)="RETAW FO
 SSALG" THEN I$(10)="SSALG":M$="!TOPS
EHT TIH TAHT":GOTO 1010
2240 IF C2$="TAW" THEN M$="!RETAW YNA
EES T'NOD I":GOTO 1010
2250 IF C2$="QIL" AND R=5 OR C2$="QIL"
 AND R=4 THEN M$="!EHTAERB T'NAC I - S
NRUB TI":GOSUB 20:GOSUB 3510:GOTO 2910

2260 IF C2$="TOP" AND I(18)=0 THEN M$=
".PU EM MRAW T'NOW ERIF A ,DLOC OS M'I
":I(18)=99:GOTO 1010
2270 M$="!!TAHT KNIRD T'NAC I◙":GOTO 1
010
2280 IF C2$<>"NAW" THEN M$="!!TAHT EVA
W OT TNAW I DLUOW YHW◙":GOTO 1010
2290 IF I(11)<>0 THEN M$="!!TAHT EVAH
T'NOD I":GOTO 1010
2300 SOUND 2,20,4,15:FOR T=1 TO 300:NE
XT:SOUND 2,0,0,0:M$="!!TLOB YGRENE NA
STIME DNAW EHT":GOSUB 20:GOSUB 3490
2310 IF R=12 AND M(1)=1 THEN M$=".HSIF
 A OTNI SNRUT NOGARD EHT":M(1)=0:I(22)
=12:GOSUB 20:GOSUB 3510:GOTO 2880
2320 M$="!!DEGNAHC SMEES GNIHTON ,EGNA
RTS":GOTO 1010
2330 IF C2$="AIG" AND M(2)=1 AND R=8 T
HEN 2400
2340 IF C2$="AIG" AND M(2)=1 AND R=21
THEN 2400
2350 IF C2$="ARD" AND M(1)=1 AND R=12
THEN 2420
2360 IF C2$="ERC" AND M(3)=1 AND R=31
THEN M$="!RETAW EHT NI TON M'I":GOTO 1
010
2370 IF C2$="ZIW" AND R=29 THEN 2430
2380 IF C2$="" THEN M$="?TAHW THGIF":G
OTO 1010
2390 M$="!THGIF OT EREH GNIHTON S'EREH
T◙":GOTO 1010
2400 IF I(5)=0 THEN GOSUB 3520:M$="!DE
HSIUQNAV NEEB SAH TNAIG EHT":M(2)=0:GO
SUB 20:GOSUB 3510:GOTO 2880
2410 GOTO 2430
```

```
2420 IF I(11)=0 THEN M$="??OD I DLUOHS
 TAHW":GOTO 1010
2430 M$="!!NOPAEW THGIR EHT EVAH T'NOD
 I"
2440 IF RND(0)<0.5 THEN C4$="!!DEKCATT
A NEEB EV'I       "+M$:M$=C4$:GOSUB 20
:GOSUB 3520:GOTO 2910
2450 GOTO 1010
2460 IF C4$="PLEH" THEN C1$="LEH":GOTO
 1300
2470 IF RIGHT$(C4$,1)="H" AND R=29 AND
 I(11)=99 THEN GOSUB 2530:M$="'?DNAW A
 YUB OT TNAW' :DRAZIW":AN=1:GOTO 1010
2480 IF AN=1 AND C2$="SEY" AND R=29 TH
EN GOSUB 2530:M$="'.DLOG FO GAB A EM E
VIG' SYAS DRAZIW":GOTO 1010
2490 IF RIGHT$(C4$,1)="H" AND R=8 OR R
IGHT$(C4$,1)="H" AND R=21 THEN GOSUB 2
530:GOTO 2520
2500 IF C2$="" THEN M$="?YAS OT EM TNA
W UOY OD TAHW":GOTO 1010
2510 GOSUB 2530:M$="!!SNEPPAH GNIHTON"
:GOTO 1010
2520 M$=".REWSNA TON SEOD TNAIG EHT":G
OTO 1010
2530 M$="!YAKO":GOSUB 20:GOSUB 3490:M$
=C4$:GOSUB 20:GOSUB 3490:RETURN
2540 IF C2$="RRA" AND R=17 THEN KH=1:M
$="!!SRAEPPA ELOHYEK SSARB A":GOSUB 20
:GOSUB 3490:GOTO 2880
2550 IF (C2$="QIL" AND R=4) OR (C2$="Q
IL" AND R=5) THEN I(6)=4:I(8)=5:GOTO 2
570
2560 M$=".EM OT "+C4$+" A EKIL SLEEF T
I":GOTO 1010
2570 M$="!!SNRUB REGNIF YM":GOSUB 20:G
OSUB 3490:M$="!GNISSIM EB OT SMEES REG
NIF YM":GOSUB 20:GOSUB 3510:GOTO 2880
2580 PRINT "↖           WARRIOR'S INVEN
TORY:       ":PRINT :PRINT
2590 FOR CH=1 TO 22:IF I(CH)=0 THEN M$
=I$(CH):GOSUB 20:PRINT
2600 NEXT:PRINT AT(7,22);"PRESS ANY KE
Y TO CONTINUE":POKE 764,255
2610 IF PEEK(764)=255 THEN 2610
2620 POKE 764,255:GOTO 2880
2630 IF C2$="MAN" AND R=6 OR C2$="OOB"
 AND R=6 THEN 2660
2640 IF C2$="" THEN M$="??TAHW ETIRW":
GOTO 1010
```

```
2650 M$="???"+C4$+" ETIRW I DLUOHS YHW
":GOTO 1010
2660 INPUT "ENTER YOUR NAME ␛→";NA$:IF
 LEN(NA$)<2 THEN PRINT "↑↑":GOTO 2660
2670 D7=1:GOTO 2880
2680 IF CN=1 THEN M$=".ROOLF EHT OTNI
SKAOS RETAW EHT":GOTO 1010
2690 IF R<>3 THEN M$="!!ROOLF EHT NI E
LOH A ETA DICA EHT":GOTO 1010
2700 IF I(12)=0 THEN M$="!KCOL EHT NI
ELOH A ETA DICA EHT":D1=1:GOSUB 20:GOS
UB 3490:GOTO 2880
2710 M$=".ROOD EHT NI SI ELOH A":PO=PO
+1:GOSUB 20:GOSUB 3490
2720 IF PO>0 THEN M$=".PLEH DLUOW LENN
UF EHT KNIHT I":GOTO 1010
2730 M$="!!PLEH T'NDID TAHT !EGNARTS":
GOTO 1010
2740 IF R<>7 THEN M$="!!EREH ROOLF ENO
TS A SI EREHT":GOTO 1010
2750 IF D4>0 THEN M$="!!EREH GUD YDAER
LA EV'I":GOTO 1010
2760 M$="!!ROOLF EHT NI ELOH A SI EREH
T":D4=1:I(16)=7:GOSUB 20:GOSUB 3510:GO
TO 2880
2770 IF C2$<>"ARB" AND C2$<>"YEK" THEN
 2780
2775 IF R=17 AND I(20)=0 THEN KH=2:E=3
4:I(20)=99:IN=IN-1:GOTO 2880
2780 M$="!!KROW T'NSEOD TI":GOTO 1010
2790 I(T1)=35:IF T1=4 THEN I(T1)=19
2800 IF T1=14 THEN I$(T1)="HCROT"
2810 T1=0:FOR T=1 TO 22:IF I(T)=0 THEN
 T1=T:T=22
2820 NEXT:IF T1>0 THEN GOSUB 3500:GOSU
B 40:GOTO 960
2830 R=47:GOTO 2880
2840 PRINT :PRINT "OKAY!":IF DK=0 THEN
 GOSUB 3200
2850 IF I(14)<>0 THEN DK=0
2860 IF I(18)=0 AND LEFT$(I$(18),5)="T
SEHC" AND R=1 THEN 3410
2870 L$="":V1$="":V2$="":IT$="":CO$=""
:N=0:S=0:E=0:W=0:U=0:D=0
2880 IF R<=21 THEN ON R GOTO 600,610,5
0,70,90,110,120,130,150,160,170,180,19
0,200,210,220,230,240,250,260,270
2890 IF R<=41 THEN ON R-21 GOTO 290,30
0,310,320,340,350,360,370,380,390,410,
420,430,440,460,470,480,490,500,510
```

```
2900 ON R-41 GOTO 520,530,540,550,560,
570,600,610,620,630
2910 GRAPHICS 0:POKE 752,1:POKE 710,0:
GOSUB 3450:PRINT AT(2,10);"YOU SEEM TO
 HAVE GOTTEN YOUR WARRIOR  KILLED."
2920 PRINT :PRINT "WOULD YOU LIKE SEND
 ANOTER ONE DOWN         TO TRY AGAIN  (
Y/N)?":PRINT
2930 A$=INKEY$:IF A$="Y" THEN RUN
2940 IF A$="N" THEN M$="!EMITEMOS NIAG
A YRT OT EKIL DLUOW I":GOSUB 20:END
2950 GOTO 2930
2960 GRAPHICS 2:POKE 710,0:PRINT #6,AT
(3,3);"halls of death":POKE 752,1:PRIN
T "          by Jack Hardy"
2970 RESTORE 3170:GOSUB 3120:GOSUB 345
0
2980 PRINT #6,AT(1,7);"WANT INSTRUCTIO
NS?        (Y/N)"
2990 A$=INKEY$:IF A$="Y" THEN 3020
3000 IF A$="N" THEN GRAPHICS 0:POKE 71
0,0:GOTO 2880
3010 GOTO 2990
3020 GRAPHICS 0:POKE 710,130:POKE 712,
130:POKE 752,1
3030 PRINT :PRINT " You have been assi
gned the duty of   giving instructions
 to a warrior.":PRINT
3040 PRINT " The warrior's task is to
explore the Halls of Death and if poss
ible to"
3050 PRINT "retrieve the GOLDEN TREASU
RE CHEST    which is invisible to most
 mortals."
3060 PRINT :PRINT " The warrior is in
two-way communica- tion and will repor
t everything"
3070 PRINT "he sees but will do nothin
g until     instructed by you."
3080 PRINT :PRINT "There are rumors of
 a Magic Ring       which will make the
 chest visible"
3090 PRINT "to its owner."
3100 PRINT:PRINT:PRINT "  WHEN READY T
O BEGIN TRANSMISSION              PRE
SS RETURN ":GET A$
3110 GOTO 2880
3120 DIM I$(23),I(23),DI$(6),M(3),VOC$
(31):D1=0:D2=0:D3=0:D4=0:D5=0:D6=0:D7=
0:D8=0:G1=0:KH=0:LD=0
```

```
3130 RANDOMIZE:FOR T=1 TO 3:M(T)=1:NEX
T:CN=0:R=1:TT=0:Q=0:IN=0
3140 FOR T=1 TO 22:READ I$(T):READ I(T
):NEXT T
3150 FOR T=1 TO 6:READ DI$(T):NEXT:EYE
$=" ●  ● "
3155 FOR T=1 TO 31:READ VOC$(T):NEXT
3160 CL$="↑
         ":CL1$="↑↑
                  ":RETURN
3170 DATA SEHCTAM,46,LEVOHS,34,LLORCS,
24,YEK NORI,1,DROWS,99,DICA,99,DLOG FO
 GAB,4,DICA,99
3180 DATA KOOB,30,SSALG,16,DNAW,99,LEN
NUF,3,GNIR CIGAM,99,HCROT,46,YEK DLOG,
2,YEK REVLIS,99
3190 DATA YRANOITCID,27,NOITOP,25,REDD
AL,99,YEK SSARB,33,DRAOB,20,HSIF,99,N,
S,E,W,U,D
3195 DATA OG,LNU,EPO,MUJ,ILC,VIG,ORD,R
HT,TEG,KAT,OOL,AXE,LEH,AER,IRD,VAW,LIK
,GIF,EPS,YAS,EEF,UOT,VNI,TXE,GIL
3197 DATA IRW,GIS,UOP,GID,SNI,ESU
3200 I$(14)="HCROT":IF R=1 OR R=46 OR
R=35 THEN RETURN
3210 POKE 764,255:POKE 712,0:POKE 710,
0:PRINT "↖          I CAN'T SEE - WHERE A
M I?       "
3220 PRINT AT(2,15);"              COM
MAND?               "
3230 PRINT AT(16,7);"╱‾╲ ╱‾╲"
3240 FOR EY=1 TO 2:FOR EYE=17 TO 15 ST
EP -1:PRINT AT(EYE,8);EYE$:FOR T=1 TO
150:NEXT T:NEXT EYE
3250 FOR EYE=15 TO 17:PRINT AT(EYE,8);
EYE$:FOR T=1 TO 150:NEXT T:NEXT EYE
3330 NEXT EY:IF PEEK(764)<>255 THEN 33
60
3340 PRINT AT(16,8);" —— ——"
3350 FOR T=1 TO 200:NEXT:GOTO 3240
3360 POKE 752,0:POKE 764,255:INPUT AT(
2,16) CO$:IF LEN(CO$)<3 THEN PRINT AT(
2,16);"        ":GOTO 3360
3370 POKE 752,1:IF LEFT$(CO$,3)="LIG"
AND I(1)=0 AND I(14)=0 THEN DK=1:GOSUB
 3500:I$(14)=I$(14)+" GNINRUB
":RETURN
3380 IF LEFT$(CO$,3)="LIG" AND I(1)<>0
 THEN M$="!!SEHCTAM YNA EVAH T'NOD I◘"
:GOSUB 20:GOSUB 3490:GOTO 3360
```

```
3390 IF LEFT$(CO$,3)="LIG" AND I(14)<>
0 THEN M$="!!HCROT A EVAH T'NOD I↖":GO
SUB 20:GOSUB 3490:GOTO 3360
3400 M$="!!KRAD    EHT NI SGNIHT OD OT
 SUOREGNAD S'TI":GOSUB 20:GOSUB 3510:G
OTO 2910
3410 GRAPHICS 18:PRINT #6,AT(3,4);"CON
GRATULATIONS":PRINT #6,AT(6,5);"you wo
n!"
3420 FOR L=1 TO 3:FOR T=1 TO 10:FOR S=
60 TO 80:SOUND 0,S,10,8:NEXT S:POKE 71
2,T*10:NEXT T:SOUND 0,0,0,0
3430 POKE 712,0:FOR T=1 TO 500:NEXT T:
NEXT L
3440 GRAPHICS 18:PRINT #6, AT(0,5);"TH
ANKS FOR THE GAME!":FOR T=1 TO 4000:NE
XT:END
3450 RESTORE 3480
3460 READ A,B:IF A>1 THEN SOUND 0,A,10
,8:FOR I=1 TO B:NEXT:SOUND 0,0,0,0:FOR
 I=1 TO 40:NEXT:GOTO 3460
3470 RETURN
3480 DATA 100,360,100,240,100,60,100,3
60,85,480,90,360,100,240,105,120,100,3
60,0,0
3490 FOR T=1 TO LEN(M$)*100:NEXT:PRINT
 CL$:PRINT CL1$:CO$="":RETURN
3500 PRINT "OKAY!":FOR T=1 TO 150:NEXT
:RETURN
3510 FOR T=1 TO LEN(M$)*100:NEXT:RETUR
N
3520 FOR L=1 TO 20:SOUND 0,20,INT(8*RN
D(0))+1,15:FOR T=1 TO 8:NEXT T:SOUND 0
,0,0,0
3530 FOR T=1 TO INT(80*RND(0)):NEXT T:
NEXT L:RETURN
```

MAD BOMBER

Objective: to find the bomb container, get it open and dismantle the bomb before it explodes. The bomb is inside a house and there is a different time limit for each level of play. The various objects necessary to dismantle the bomb are randomly placed in the six main rooms of the house. It is a unique game each time it is played since the objects must be used in a certain order.

The program requires Atari BASIC and 8K from cassette or 16K from disk.

```
5 GOSUB 1000
7 FOR G=1 TO 10:SOUND 0,80,2,4:FOR D=1
 TO 2:NEXT D:SOUND 0,0,0,0:FOR D=1 TO
55:NEXT D:NEXT G
8 POKE 710,ROOM*16+2:POKE 712,PEEK(710
):N=0:S=0:E=0:W=0
9 N=N(ROOM):S=S(ROOM):E=E(ROOM):W=W(RO
OM)
100 PRINT "↖"
110 PRINT "━━━━━━━━━━━━━━━━━━━━━━━━━━
━━━━━━━━━━━━━━"
━━━━━━━━━━━━━━
120 PRINT :PRINT "YOU'RE IN THE ";LOCA
TION$(ROOM*14-13,ROOM*14)
130 PRINT :PRINT "╲╲╲╲╲╲╲╲╲╲╲╲╲╲╲╲╲╲╲◥◢
╱╱╱╱╱╱╱╱╱╱╱╱╱╱╱╱╱╱╱╱╱╱╱╱╱╱"
140 PRINT "╱╱╱╱╱ THERE ARE DOORWAYS GO
ING: ╲╲╲╲╲╲"
150 PRINT :IF N THEN PRINT "NORTH, ";
160 IF S THEN PRINT "SOUTH, ";
170 IF E THEN PRINT "EAST, ";
180 IF W THEN PRINT "WEST, ";
190 PRINT "←← "
230 PRINT :PRINT "━━━━━━━━━━━━━━━━━━━━
▬▬▬▬▬▬▬▬▬▬▬▬▬━━━━━━━━━━━━━━━━━━━━━━━━━"
━━━━━━━━▬▬▬▬▬█████████▙▬▬▬▬▬
240 PRINT "━━━━━━━━━━━━━━ ITEMS YOU SEE ▟
▬▬▬▬▬▬▬▬▬▬▬▬"
━━━━━━━━━━━━━━
250 PRINT :FOR G=1 TO 10
260 IF ITEM(G)=ROOM THEN I$=ITEM$(G*11
-10,G*11):GOSUB 270:PRINT I$;", ";:I$=
""
265 NEXT G:GOTO 275
270 IF I$(LEN(I$),LEN(I$))=" " THEN I$
=I$(1,LEN(I$)-1):GOTO 270
272 RETURN
```

```
275 PRINT "←← "
280 PRINT :PRINT "▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄
▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄"
290 PRINT "             INSTRUCTIONS? 
           ":PRINT
294 TIME=PEEK(19)/14:IF TIME>=MAXTIME
THEN 950
296 IF TIME>=MAXTIME-1 THEN SOUND 1,9,
2,1
300 TRAP 3000:MESSAGE$="":SUBMESSAGE$=
""
310 INPUT MESSAGE$:IF LEN(MESSAGE$)>1
THEN 400
320 FOR G=1 TO 4:IF DIR$(G,G)=MESSAGE$
 THEN DIR=G:POP :GOTO 340
330 NEXT G:PRINT "I DIDN'T UNDERSTAND
THAT!":GOTO 3050
340 ON DIR GOTO 350,360,370,380
350 IF N THEN ROOM=N:GOTO 7
355 GOTO 390
360 IF S THEN ROOM=S:GOTO 7
365 GOTO 390
370 IF E THEN ROOM=E:GOTO 7
375 GOTO 390
380 IF W THEN ROOM=W:GOTO 7
390 PRINT "YOU CAN'T GO IN THAT DIRECT
ION!":GOTO 3050
400 FOR G=1 TO LEN(MESSAGE$):IF MESSAG
E$(G,G)=" " THEN SUBMESSAGE$=MESSAGE$(
G+1,LEN(MESSAGE$)):POP :GOTO 415
410 NEXT G
415 IF TIME>=MAXTIME THEN 950
420 FOR G=1 TO 18:IF MESSAGE$(1,3)=VOC
ABULARY$(G*3-2,G*3) THEN VOC=G:POP :GO
TO 440
430 NEXT G:PRINT "THAT IS NOT PART OF
MY VOCABULARY!":GOTO 3050
440 ON VOC GOTO 450,450,450,470,470,52
0,520,600,600,600,670,670,710,710,730,
730,770,830
450 FOR G=1 TO 4:IF SUBMESSAGE$(1,1)=D
IR$(G,G) THEN DIR=G:POP :GOTO 340
455 IF SUBMESSAGE$(1,3)="CHA" AND ITEM
(1)=ROOM THEN 710
460 NEXT G:PRINT "YOU CAN'T GO THERE!"
:GOTO 3050
470 FOR G=1 TO 10:IF SUBMESSAGE$(1,3)=
ITEM$(G*11-10,G*11-8) THEN POP :GOTO 4
90
480 NEXT G:PRINT "YOU CAN'T TAKE A ";S
```

```
UBMESSAGE$:GOTO 3050
490 IF ITEM(G)=10 THEN PRINT "YOU ALRE
ADY HAVE IT!":GOTO 3050
500 IF ITEM(G)<>ROOM THEN PRINT "THAT
ITEM IS NOT HERE!":GOTO 3050
510 IF SUBMESSAGE$="KEY" AND (CHAIR<>1
 OR ITEM(1)<>ROOM) THEN 518
515 ITEM(G)=10:GOTO 9
518 PRINT "IT'S TOO HIGH. YOU NEED TO
STAND ON   SOMETHING!↑":GOTO 3050
520 FOR G=1 TO 10:IF SUBMESSAGE$(1,3)=
ITEM$(G*11-10,G*11-8) THEN POP :GOTO 5
40
530 NEXT G:PRINT "IT LOOKS LIKE A ";SU
BMESSAGE$;"!!":GOTO 3050
540 IF ITEM(G)<>ROOM AND ITEM(G)<>10 T
HEN PRINT "IT'S NOT HERE TO LOOK AT!":
GOTO 3050
545 ON G GOTO 590,590,550,560,570,580,
590,590,585,590
550 PRINT "THERE IS A KEYHOLE!":GOTO 3
050
560 PRINT "IT WILL HAVE TO BE FORCED O
PEN!":GOTO 3050
570 PRINT "SCREWS HOLD THE LID CLOSED!
":GOTO 3050
580 PRINT "YOU FIND A PAIR OF SCISSORS
!":ITEM(10)=ROOM:FOR D=1 TO 200:NEXT D
:GOTO 9
585 PRINT "THE WIRES NEED TO BE CUT!":
GOTO 3050
590 PRINT "I SEE NOTHING SPECIAL!":GOT
O 3050
600 FOR G=3 TO 6:IF SUBMESSAGE$(1,3)=I
TEM$(G*11-10,G*11-8) THEN POP :GOTO 61
0
605 NEXT G:PRINT "YOU CAN'T OPEN THAT!
":GOTO 3050
610 IF ITEM(G)<>ROOM THEN PRINT "IT'S
NOT IN THIS ROOM!":GOTO 3050
612 IF ITEM(G+4) THEN PRINT "IT'S ALRE
ADY OPEN!":GOTO 3050
615 ON G-2 GOTO 620,630,640,650
620 IF ITEM(2)=10 THEN ITEM(7)=ROOM:GO
TO 9
625 GOTO 660
630 IF ITEM(7)=10 THEN ITEM(8)=ROOM:GO
TO 9
635 GOTO 660
640 IF ITEM(8)=10 THEN ITEM(9)=ROOM:GO
```

```
TO 9
645 GOTO 660
650 PRINT "OKAY!":GOTO 3050
660 PRINT "YOU NEED HELP TO OPEN THAT!
":GOTO 3050
670 FOR G=1 TO 10:IF SUBMESSAGE$(1,3)=
ITEM$(G*11-10,G*11-8) THEN POP :GOTO 6
90
680 NEXT G:PRINT "DROP WHAT!!!!":GOTO 
3050
690 IF ITEM(G)=10 THEN ITEM(G)=ROOM:GO
TO 9
700 PRINT "YOU'RE NOT CARRYING IT!!":G
OTO 3050
710 IF SUBMESSAGE$(1,3)="CHA" THEN 722

715 IF MESSAGE$(1,3)="STA" AND SUBMESS
AGE$(4,6)="CHA" THEN 722
720 PRINT "YOU CAN'T ";MESSAGE$(1,5);"
 ON THAT!!":GOTO 3050
722 IF ITEM(1)<>ROOM THEN PRINT "IT'S 
NOT HERE!":GOTO 3050
725 CHAIR=1:GOSUB 3060:GOTO 3050
730 FOR G=1 TO 10:IF SUBMESSAGE$(1,3)=
ITEM$(G*11-10,G*11-8) THEN POP :GOTO 7
50
740 NEXT G:PRINT "IT FEELS LIKE A ";SU
BMESSAGE$;" TO ME!":GOTO 3050
750 IF ITEM(G)<>ROOM THEN PRINT "IT'S 
NOT HERE TO TOUCH!":GOTO 3050
755 IF SUBMESSAGE$(1,3)="BOM" THEN PRI
NT "I FEEL SOME WIRES!":GOTO 3050
760 GOSUB 3060:GOTO 3050
770 PRINT "↰":PRINT "▌▌▌▌▌▌▌▌▌▌▌▌ ITEMS
YOU HAVE: ▌▌▌▌▌▌▌▌▌▌▌▌":PRINT :FOR G=1 T
O 10
780 IF ITEM(G)=10 THEN PRINT ITEM$(G*1
1-10,G*11)
790 NEXT G
800 POSITION 5,22:PRINT "TOUCH  SPACE 
BAR  TO CONTINUE"
810 OPEN #1,4,0,"K:":GET #1,K:IF K<>32
 THEN CLOSE #1:GOTO 800
820 CLOSE #1:GOTO 9
830 IF SUBMESSAGE$(1,3)<>"WIR" THEN PR
INT "I CAN'T CUT THAT!":GOTO 3050
840 IF ITEM(9)<>ROOM THEN PRINT "THE B
OMB IS NOT HERE!":GOTO 3050
845 IF ITEM(10)<>10 THEN PRINT "YOU DO
N'T HAVE ANY SCISSORS!":GOTO 3050
```

```
850 GRAPHICS 18:POSITION 4,1:PRINT #6;
"YOU DID IT!"
860 RESTORE 4050:FOR D=1 TO 7:READ SO,
DELAY
870 SOUND 1,SO,10,10:POKE 712,SO:FOR D
2=1 TO DELAY:NEXT D2:NEXT D:SOUND 1,0,
0,0
880 IF LEVEL=3 THEN POSITION 2,2:PRINT
 #6;"YOU SURVIVED THE":POSITION 3,3:PR
INT #6;"HIGHEST LEVEL":GOTO 965
890 LEVEL=LEVEL+1
900 POSITION 2,3:PRINT #6;"YOU HAVE BE
EN":POSITION 1,4:PRINT #6;"PROMOTED TO
 LEVEL ";LEVEL
910 FOR DELAY=1 TO 100:NEXT DELAY
920 GOSUB 1130:GOTO 7
950 SOUND 1,0,0,0:GOSUB 2000:FOR DELAY
=1 TO 20:NEXT DELAY
960 GRAPHICS 18:POSITION 1,3:PRINT #6;
"you were too late!"
965 POSITION 1,5:PRINT #6;"would you l
ike to":POSITION 2,6:PRINT #6;"try aga
in? (y/n)"
970 OPEN #1,4,0,"K:":GET #1,K:IF K<>78
 AND K<>89 THEN CLOSE #1:GOTO 970
980 CLOSE #1:IF K=89 THEN GOSUB 1130:G
OTO 7
990 POSITION 1,8:PRINT #6;"THANKS FOR 
PLAYING":FOR DELAY=1 TO 250:NEXT DELAY
:END
1000 FOR G=8 TO 2 STEP -1
1010 GRAPHICS G+16:REM GRAPHICS WITH N
O WINDOW
1020 POSITION G+3,4:? #6;"mad bomber"
1030 SOUND 0,RND(0)*255,10,10
1035 FOR DELAY=1 TO 10:NEXT DELAY
1040 NEXT G
1050 DIM MESSAGE$(30):MESSAGE$(1,13)="
BY JACK HARDY"
1070 FOR G=1 TO 13
1080 POKE 709,INT(16*RND(0))*16+8:POSI
TION G+3,10:? #6;MESSAGE$(G,G)
1090 SOUND 0,80,10,4
1095 FOR DELAY=1 TO 5:NEXT DELAY:SOUND
 0,0,0,0:FOR DELAY=1 TO 5:NEXT DELAY
1100 NEXT G
1110 FOR DELAY=1 TO 100:NEXT DELAY:GOS
UB 2000:LEVEL=1
1120 DIM ITEM(10),I$(14),ITEM$(110),DI
R$(4),VOCABULARY$(54),LOCATION$(112),S
```

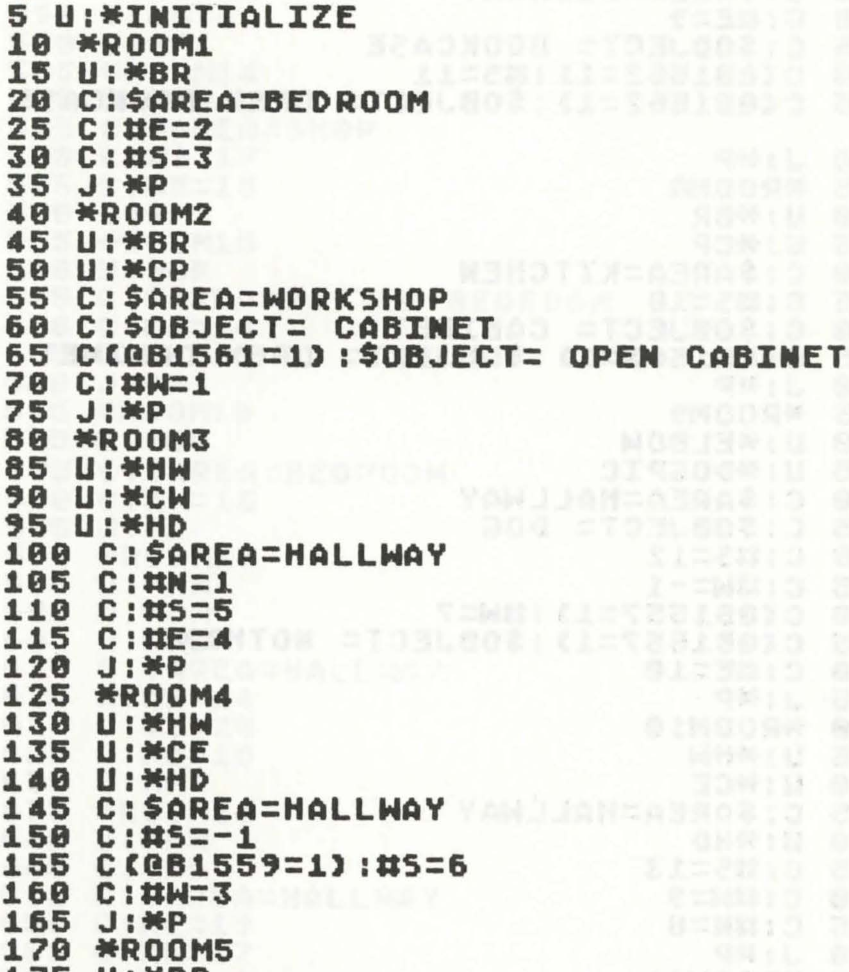
```
UBMESSAGE$(30)
1125 DIM N(8),S(8),E(8),W(8)
1130 ITEM$(1)=" ":ITEM$(110)=" ":ITEM$
(2)=ITEM$:I$=ITEM$(1,11):MESSAGE$=ITEM
$(1,30)
1140 LOCATION$(1)=" ":LOCATION$(112)="
 ":LOCATION$(2)=LOCATION$
1155 REM ** READ IN ITEMS - DIRECTION
S - COMMANDS **
1160 RESTORE :FOR G=1 TO 10:READ I$:IT
EM$(G*11-10,G*11)=I$:NEXT G
1170 DIR$="NSEW"
1180 FOR G=1 TO 18:READ I$:VOCABULARY$
(G*3-2,G*3)=I$(1,3):NEXT G
1185 FOR G=1 TO 8:READ I$,D1,D2,D3,D4:
LOCATION$(G*14-13,G*14)=I$:N(G)=D1:S(G
)=D2:E(G)=D3:W(G)=D4:NEXT G
1190 FOR G=1 TO 6:ITEM(G)=G:NEXT G:FOR
 G=7 TO 10:ITEM(G)=0:NEXT G
1200 FOR G=6 TO 2 STEP -1:J=INT(G*RND(
0))+1:I=ITEM(J):ITEM(J)=ITEM(G):ITEM(G
)=I:NEXT G
1210 I$="":CHAIR=0:ROOM=8
1220 POSITION 2,8:PRINT #6;"press"
1230 POSITION 2,9:PRINT #6;"START TO B
EGIN"
1235 POSITION 2,10:PRINT #6;"SELECT /
LEVEL ";LEVEL
1240 IF PEEK(53279)=6 THEN 1270
1250 IF PEEK(53279)=5 THEN LEVEL=LEVEL
+1:IF LEVEL>3 THEN LEVEL=1
1260 FOR DELAY=1 TO 10:NEXT DELAY:GOTO
 1235
1270 IF LEVEL=1 THEN MAXTIME=15
1280 IF LEVEL=2 THEN MAXTIME=5
1290 IF LEVEL=3 THEN MAXTIME=2
1300 GRAPHICS 0:POKE 710,0:POKE 752,1
1310 POSITION 11,11:PRINT "THE GAME IS
 AFOOT!":POKE 18,0:POKE 19,0:POKE 20,0
:RETURN
2000 DLIST=PEEK(560)+256*PEEK(561):G=P
EEK(712)
2010 FOR W=15 TO 0 STEP -0.25:SOUND 0,
50,0,W:CO=1-CO:POKE 712,CO*(4*16+6):PO
KE 710,PEEK(712):POKE DLIST,112*CO
2020 FOR DELAY=1 TO 5:NEXT DELAY:NEXT
W
2030 POKE DLIST,112:POKE 712,G:POKE 71
0,G:RETURN
3000 PRINT "PLEASE SAY THAT AGAIN!":TR
```

```
AP 40000
3050 FOR G=1 TO 200:NEXT G:PRINT "↑↑↑↓
↓↓↓↓↓":GOTO 300
3060 PRINT "OKAY!":RETURN
4000 DATA CHAIR,KEY,CHEST,CABINET,BOX,
BASKET,CROWBAR,SCREWDRIVER,BOMB,SCISSO
RS
4010 DATA WALK,RUN,GO ,GET,TAKE,LOOK,E
XAMINE,OPEN,UNSCREW,UNLOCK,DROP,PUT,ST
AND ON,CLIMB,TOUCH,FEEL,INVENTORY,CUT
4020 DATA KITCHEN,3,0,8,0,LIVING ROOM,
4,0,0,8,DINING ROOM,5,1,7,0,DEN,6,2,0,
7
4030 DATA MASTER BEDROOM,0,3,6,0,BEDRO
OM,0,4,0,5,HALL WAY,0,8,4,3,HALL WAY,7
,0,2,1
4050 DATA 96,15,72,15,57,15,48,30,57,1
5,48,60,0,0
```

ESCAPE

Objective: to escape from the top floor of a building. You awaken in a dark room. The last thing you remember is being hit on the head and thrown into a car. The only thing on your mind now is escape.

The program requires PILOT and 32K with cassette or 48K with disk.



```
5 U:*INITIALIZE
10 *ROOM1
15 U:*BR
20 C:$AREA=BEDROOM
25 C:#E=2
30 C:#S=3
35 J:*P
40 *ROOM2
45 U:*BR
50 U:*CP
55 C:$AREA=WORKSHOP
60 C:$OBJECT= CABINET
65 C(@B1561=1):$OBJECT= OPEN CABINET
70 C:#W=1
75 J:*P
80 *ROOM3
85 U:*HW
90 U:*CW
95 U:*HD
100 C:$AREA=HALLWAY
105 C:#N=1
110 C:#S=5
115 C:#E=4
120 J:*P
125 *ROOM4
130 U:*HW
135 U:*CE
140 U:*HD
145 C:$AREA=HALLWAY
150 C:#S=-1
155 C(@B1559=1):#S=6
160 C:#W=3
165 J:*P
170 *ROOM5
175 U:*BR
180 U:*PICPIC
185 C:$AREA=OFFICE
```

```
190 C:#N=3
195 J:*P
200 *ROOM6
205 U:*SR
210 C:$AREA=ELEVATOR
215 C:#N=4
220 C:#D=13
225 J:*P
230 *ROOM7
235 U:*BR
240 U:*BOOKCPIC
245 C:$AREA=LIBRARY
250 C:#E=9
255 C:$OBJECT= BOOKCASE
260 C(@B1562=1):#S=11
265 C(@B1562=1):$OBJECT= OPEN BOOKCASE

270 J:*P
275 *ROOM8
280 U:*BR
285 U:*CP
290 C:$AREA=KITCHEN
295 C:#S=10
300 C:$OBJECT= CABINET
305 C(@B1563=1):$OBJECT= OPEN CABINET
310 J:*P
315 *ROOM9
320 U:*ELBOW
325 U:*DOGPIC
330 C:$AREA=HALLWAY
335 C:$OBJECT= DOG
340 C:#S=12
345 C:#W=-1
350 C(@B1557=1):#W=7
355 C(@B1557=1):$OBJECT= NOTHING
360 C:#E=10
365 J:*P
370 *ROOM10
375 U:*HW
380 U:*CE
385 C:$AREA=HALLWAY
390 U:*HD
395 C:#S=13
400 C:#W=9
405 C:#N=8
410 J:*P
415 *ROOM11
420 U:*BR
425 C:$AREA=MAID'S ROOM
430 C:#N=7
```

```
435 C:#D=20
440 C:$OBJECT= LAUNDRY CHUTE
445 J:*P
450 *ROOM12
455 U:*BR
460 C:$AREA=BEDROOM
465 C:#N=9
470 J:*P
475 *ROOM13
480 U:*SR
485 C:$AREA=ELEVATOR
490 C:#N=10
495 C:#U=6
500 J:*P
505 *ROOM14
510 U:*BR
515 C:$AREA=SHOP
520 C:#S=17
525 C:#E=15
530 J:*P
535 *ROOM15
540 U:*BR
545 C:$AREA=MASTER BEDROOM
550 C:#W=14
555 C:#E=16
560 J:*P
565 *ROOM16
570 U:*BR
575 C:$AREA=BEDROOM
580 C:#W=15
585 J:*P
590 *ROOM17
595 U:*HW
600 U:*CW
605 U:*HD
610 C:$AREA=HALLWAY
615 C:#N=14
620 C:#S=20
625 C:#E=18
630 J:*P
635 *ROOM18
640 U:*HW
645 U:*HD
650 C:$AREA=HALLWAY
655 C:#E=19
660 C:#W=17
665 C:#S=-1
670 C(@B1566=1):#S=21
675 J:*P
680 *ROOM19
```

```
685 U:*HW
690 U:*HD
695 U:*CE
700 C:$AREA=HALLWAY
705 C:#W=18
710 C:#S=-1
715 C(@B1564=1):#S=22
720 J:*P
725 *ROOM20
730 U:*BR
735 C:$AREA=LAUNDRY ROOM
740 C:#N=-1
745 C:#U=11
750 C(@B1565=1):#N=17
755 C:$OBJECT= DIRTY CLOTHES
760 J:*P
765 *ROOM21
770 U:*BR
775 C:$AREA=BUTLER'S ROOM
780 C:#N=18
785 J:*P
790 *ROOM22
795 U:*SR
800 C:$AREA=ELEVATOR SHAFT
805 C:#D=27
810 C:#N=19
815 J:*P
820 *ROOM23
825 U:*BR
830 U:*SINKPIC
835 C:$AREA=JANITOR'S WORKROOM
840 C:$OBJECT= SINK
845 C:#E=24
850 C:#S=26
855 J:*P
860 *ROOM24
865 U:*BR
870 C:$AREA=STORAGE
875 C:#S=27
880 C:#W=23
885 J:*P
890 *ROOM25
895 U:*SR
900 U:*FIREPIC
905 C:$AREA=CHIMNEY
910 C:#U=31
915 C:#E=26
920 J:*P
925 *ROOM26
930 U:*BR
```

```
935 U:*FIREPIC
940 C:$AREA=JANITOR'S QUARTERS
945 C:$OBJECT= BURNING FIREPLACE
950 C:#N=23
955 C:#W=-1
960 C(@B1567=1):#W=25
965 C(@B1567=1):$OBJECT= FIREPLACE
970 J:*P
975 *ROOM27
980 U:*SR
985 C:$AREA=ELEVATOR SHAFT
990 C:#N=24
995 C:#U=22
1000 J:*P
1005 *ROOM28
1010 U:*BR
1015 U:*GORILLAPIC
1020 C:$AREA=ZOO
1025 C:#S=32
1030 C:$OBJECT= GORILLA
1035 C(@B1568=1):$OBJECT= SLEEPING GOR
ILLA
1040 C:#E=-1
1045 C(@B1568=1):#E=29
1050 J:*P
1055 *ROOM29
1060 U:*ELBOW
1065 C:$AREA=HALLWAY
1070 C:#W=28
1075 C:#E=34
1080 J:*P
1085 *ROOM30
1090 U:*BR
1095 C:$AREA=MAIN LOBBY
1100 C:#N=50
1105 C:#S=34
1110 J:*P
1115 *ROOM31
1120 U:*SR
1125 U:*FIREPIC
1130 C:$AREA=CHIMNEY
1135 C:#E=32
1140 C:#D=25
1145 J:*P
1150 *ROOM32
1155 U:*BR
1160 U:*FIREPIC
1165 C:$AREA=LOUNGE
1170 C:$OBJECT= FIREPLACE
1175 C:#N=28
```

```
1180 C:#E=33
1185 C:#W=31
1190 J:*P
1195 *ROOM33
1200 U:*BR
1205 C:$AREA=FOOD LOCKER
1210 C:#S=35
1215 C:#W=32
1220 C(@B1552<>0):@B1552=33
1225 J:*P
1230 *ROOM34
1235 U:*HW
1240 U:*CE
1245 C:$AREA=HALLWAY
1250 C:#W=29
1255 C:#N=-1
1260 C(@B1569=1):#N=30
1265 J:*P
1270 *ROOM35
1275 U:*SR
1280 U:*MCP
1285 C:$AREA=BATHROOM
1290 C:#N=33
1295 C:$OBJECT= MEDICINE CABINET
1300 C(@B1570=1):$OBJECT= OPEN MEDICIN
E CABINET
1305 J:*P
1310 *P
1315 U(@B1558=0):*DARK
1320 T:↖LOCATION: $AREA
1325 T(#F=4):▶▶▶▶↑←4TH FLOOR
1330 T(#F=3):▶▶▶▶↑←3RD FLOOR
1335 T(#F=2):▶▶▶▶↑←2ND FLOOR
1340 T(#F=1):▶▶▶▶↑←1ST FLOOR
1345 T(#F=0):▶▶▶▶↑BASEMENT
1350 U(#N<>0):*NORTHDOOR
1355 U(#S<>0):*SOUTHDOOR
1360 U(#E<>0):*EASTDOOR
1365 U(#W<>0):*WESTDOOR
1370 U(#D<>0):*DOWNARROW
1375 U(#U<>0):*UPARROW
1380 U:*PL
1385 C:#L=1
1390 C:#M=1540
1395 C:#I=0
1400 C:$ITEMS=$OBJECT
1405 A:=$ITEMS
1410 M: NOTHING
1415 CN:#I=#I+1
1420 U:*ITEMS
```

```
1425 T:VISIBLE ITEMS:\
1430 T:$ITEMS
1435 C:@B764=255
1440 T:COMMAND => \
1445 A:
1450 M: NOR, SOU, EAS, WES, UP , DOW,
1455 JM:*NORTH,*SOUTH,*EAST,*WEST,*UP,
*DOWN
1460 M:TAK,GET,DRO,GIV,UNL,OPE,LOO,EXA
,LIG,INS,USE,PUS,GO ,INV,THR,WEA
1465 JM:*TD,*TD,*TD,*TD,*OPEN,*OPEN,*L
OOK,*LOOK,*LIGHT,*USE,*USE,*PUSH,*GO,*
INVENTORY,*THROW,*WEAR
1470 T:↖⬉I DON'T UNDERSTAND THAT!!
1475 PA:90
1480 J:*P
1485 *ITEMS
1490 C:#M=#M+1
1495 U(#R=@B#M):*VISIBLE
1500 J(#M<1557):*ITEMS
1505 E:
1510 *INVENTORY
1515 C:$ITEMS= NOTHING
1520 C:#I=0
1525 C:#M=1541
1530 C:#X=-16
1535 C:#Y=8
1540 GR:CLEAR
1545 J:*INV2
1550 *INV2
1555 U(@B#M=0):*VISIBLE
1560 C:#M=#M+1
1565 J(#M<1557):*INV2
1570 T:↖$ITEMS
1575 T(@B1549=50):WEARING GLOVES
1580 J:*PRESSKEY
1585 *VISIBLE
1590 C(#I>0):$ITEMS=$ITEMS$COMMA
1595 C:#I=#I+1
1600 C:#V=#M-1540
1605 C(#V=1):$ITEM= FLASHLIGHT
1610 C(#V=2):$ITEM= FUSES
1615 C(#V=3):$ITEM= PAINTING
1620 C(#V=4):$ITEM= DOG BISCUITS
1625 C(#V=5):$ITEM= BONE
1630 C(#V=6):$ITEM= BROWN KEY
1635 C(#V=7):$ITEM= CROWBAR
1640 C(#V=8):$ITEM= BLUE KEY
1645 C(#V=9):$ITEM= GLOVES
1650 C(#V=10):$ITEM= WATER
```

```
1655 C(#V=11):$ITEM= BUCKET
1660 C(#V=12):$ITEM= BANANA
1665 C(#V=13):$ITEM= DRUGS
1670 C(#V=14):$ITEM= RED KEY
1675 C(#V=15):$ITEM= FUSEBOX
1680 C(#V=16):$ITEM= PASSAGEWAY
1685 C(#I=1):$ITEMS=$ITEM
1690 C(#I>1):$ITEMS=$ITEMS$ITEM
1695 U:*DRAWPICTURE
1700 E:
1705*DRAWPICTURE
1710 C:#V=#M-1540
1715 U(#V=1):*FLASHPIC
1720 U(#V=2):*FUSEPIC
1725 U(#V=3):*PICPIC
1730 U(#V=4):*BISPIC
1735 U(#V=5):*BONEPIC
1740 U(#V=6):*KEYPIC
1745 U(#V=7):*CROPIC
1750 U(#V=8):*KEYPIC
1755 U(#V=9):*GLOVEPIC
1760 U(#V=11):*BUCKETPIC
1765 U(#V=12):*BANANAPIC
1770 U(#V=13):*DRUGPIC
1775 U(#V=14):*KEYPIC
1780E:
1785 *NORTH
1790 J(#N=0):*CANTGO
1795 J(#N=-1):*CHECKDOOR
1800 C:#R=#N
1805 J:*CR
1810 *SOUTH
1815 J(#S=0):*CANTGO
1820 J(#S=-1):*CHECKDOOR
1825 C:#R=#S
1830 J:*CR
1835 *EAST
1840 J(#E=0):*CANTGO
1845 J(#E=-1):*CHECKDOOR
1850 C:#R=#E
1855 J:*CR
1860 *WEST
1865 J(#W=0):*CANTGO
1870 J(#W=-1):*CHECKDOOR
1875 C:#R=#W
1880 J:*CR
1885 *UP
1890 J(#U=0):*CANTGO
1895 C:#R=#U
1900 C:#F=#F+1
```

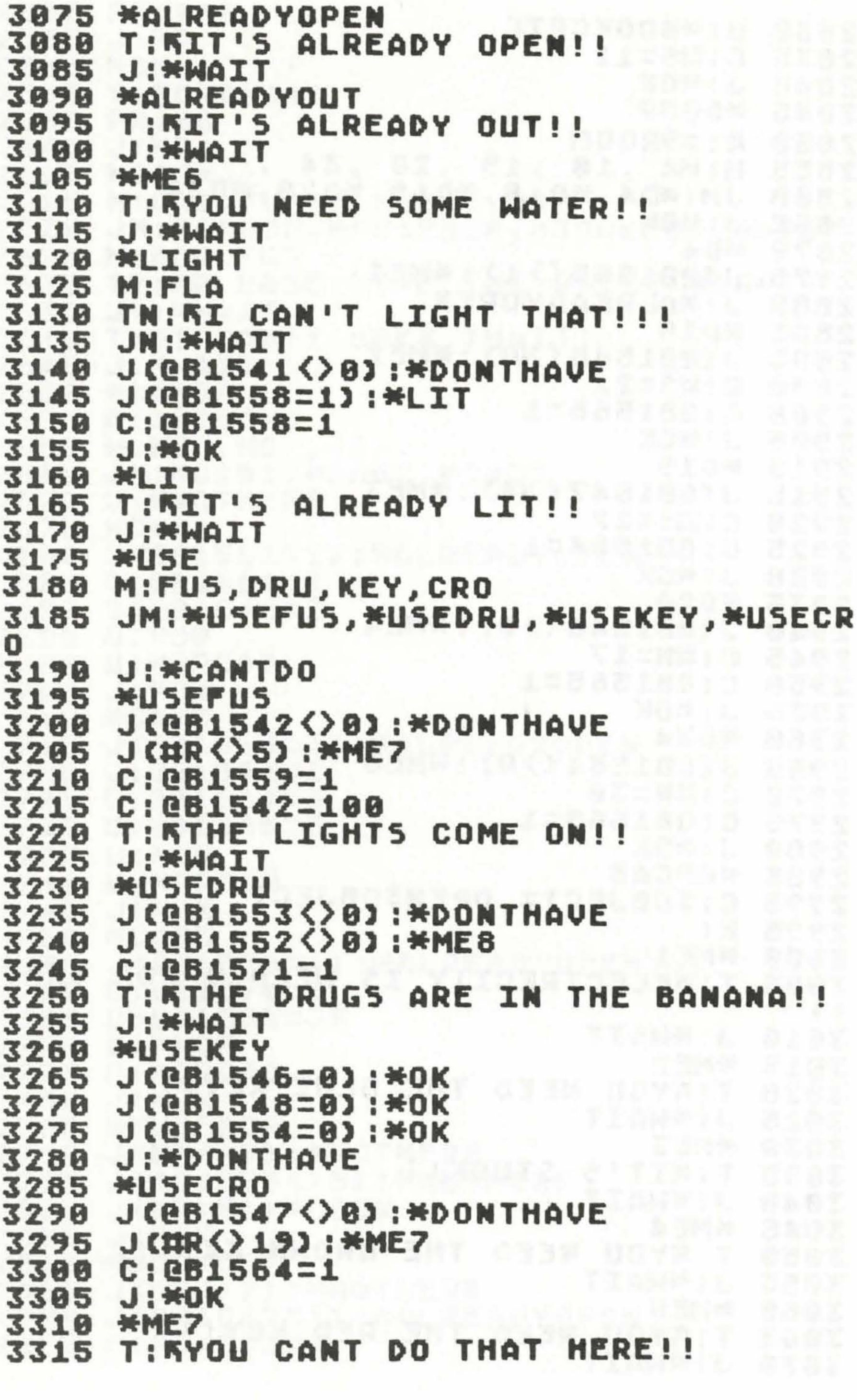
```
1905 C(#R=27):#F=#F+1
1910 J:*CR
1915 *DOWN
1920 J(#D=0):*CANTGO
1925 J(#R=11):*HOW
1930 J(#R=22):*CKG
1935 C:#R=#D
1940 C:#F=#F-1
1945 J:*CR
1950 *HOW
1955 T:↖HOW????
1960 J:*WAIT
1965 *CANTGO
1970 T:↖I CAN'T GO IN THAT DIRECTION!!

1975 J:*WAIT
1980 *CKG
1985 J(@B1549<>50):*DEAD1
1990 C:#R=#D
1995 C:#F=#F-2
2000 J:*CR
2005 *CHECKDOOR
2010 A:=$ROOM
2015 M:M4 ,M9 ,18 ,19 ,20 ,28 ,26 ,32
,34 ,
2020 JM:*STUCK,*BLOCKED,*LOCKED,*STUCK
,*LOCKED,*BLOCKED2,*HOW,*HOW,*LOCKED
2025 *STUCK
2030 T:↖THE DOOR WILL NOT OPEN!!
2035 J:*WAIT
2040 *LOCKED
2045 T:↖THE DOOR IS LOCKED!!
2050 J:*WAIT
2055 *BLOCKED
2060 T:↖THE DOG BLOCKS YOUR WAY!!!
2065 J:*WAIT
2070 *BLOCKED2
2075 T:↖THE GORILLA BLOCKS YOUR WAY!!
2080 J:*WAIT
2085 *CR
2090 C:$ROOMNUM=#R
2095 C:$OBJECT= NOTHING
2100 C:#N=0
2105 C:#S=0
2110 C:#E=0
2115 C:#W=0
2120 C:#U=0
2125 C:#D=0
2130 C:$ROOM=ROOM$ROOMNUM
2135 A:=$ROOM
```

```
2140 M:M1 ,M2 ,M3 ,M4 ,M5 ,M6 ,M7 ,M8
,M9 ,M10 ,11 ,12 ,13 ,14 ,
2145 JM:*ROOM1,*ROOM2,*ROOM3,*ROOM4,*R
OOM5,*ROOM6,*ROOM7,*ROOM8,*ROOM9,*ROOM
10,*ROOM11,*ROOM12,*ROOM13,*ROOM14
2150 M:15 ,16 ,17 ,18 ,19 ,20 ,21 ,22
,23 ,24 ,25 ,26 ,
2155 JM:*ROOM15,*ROOM16,*ROOM17,*ROOM1
8,*ROOM19,*ROOM20,*ROOM21,*ROOM22,*ROO
M23,*ROOM24,*ROOM25,*ROOM26
2160 M:27 ,28 ,29 ,30 ,31 ,32 ,33 ,34
,35 ,50 ,
2165 JM:*ROOM27,*ROOM28,*ROOM29,*ROOM3
0,*ROOM31,*ROOM32,*ROOM33,*ROOM34,*ROO
M35,*WIN
2170 E:
2175 *TD
2180 M:FLA,FUS,PAI,BIS,BON,KEY,CRO,GLO
,WAT,BUC,BAN,DRU,
2185 JM:*FLASHLIGHT,*FUSES,*PAINTING,*
BISCUITS,*BONE,*KEY,*CROWBAR,*GLOVES,*
WATER,*BUCKET,*BANANA,*DRUGS
2190 M:GET,TAK
2195 TY:↰I CAN'T TAKE THAT!!
2200 JN:*DONTHAVE
2205 J:*WAIT
2210 *FLASHLIGHT
2215 C:#M=1541
2220 J:*CM
2225 *FUSES
2230 C:#M=1542
2235 J:*CM
2240 *PAINTING
2245 C:@B1555=5
2250 C:#M=1543
2255 J:*CM
2260 *BISCUITS
2265 C:#M=1544
2270 J:*CM
2275 *BONE
2280 C:#M=1545
2285 J:*CM
2290 *CROWBAR
2295 C:#M=1547
2300 J:*CM
2305 *GLOVES
2310 C:#M=1549
2315 J:*CM
2320 *WATER
2325 J(@B1551<>0):*ME8
```

```
2330 C:#M=1550
2335 J:*CM
2340 *BUCKET
2345 C:#M=1551
2350 J:*CM
2355 *BANANA
2360 C:#M=1552
2365 J:*CM
2370 *DRUGS
2375 C:#M=1553
2380 J:*CM
2385 *KEY
2390 M:BROWN,BLUE,RED
2395 JM:*BROWN,*BLUE,*RED
2400 T:↖WHICH COLOR KEY???
2405 J:*WAIT
2410 *BROWN
2415 C:#M=1546
2420 J:*CM
2425 *BLUE
2430 C:#M=1548
2435 J:*CM
2440 *RED
2445 C:#M=1554
2450 J:*CM
2455 *CM
2460 M:GET,TAK,DRO,GIV
2465 JM:*GET,*GET,*DROP,*DROP
2470 *GET
2475 J(@B#M=0):*HAVE
2480 J(@B#M<>#R):*NOTHERE
2485 C:@B#M=0
2490 C:#L=0
2495 U:*DRAWPICTURE
2500 J:*OK
2505 *DROP
2510 J(@B#M<>0):*DONTHAVE
2515 C:@B#M=#R
2520 C:#L=1
2525 J:*OK
2530 *HAVE
2535 T:↖YOU ALREADY HAVE IT!!
2540 J:*WAIT
2545 *NOTHERE
2550 T:↖THAT ITEM IS NOT HERE!!
2555 J:*WAIT
2560 *DONTHAVE
2565 T:↖YOU AREN'T CARRYING THAT!!
2570 J:*WAIT
2575 *WAIT
```

```
2580 PA:240
2585 J:*P
2590 *OK
2595 T:↖OKAY!!
2600 PA:60
2605 J:*P
2610 *OPEN
2615 M:CAB,FUSEB,BOOKC,DOO
2620 JM:*CABOP,*FUSEBOP,*BOOKCOP,*DOOP
2625 M:BOO,FUS
2630 TY:↖PLEASE TYPE THE WHOLE WORD!!
2635 JY:*WAIT
2640 T:↖I CAN'T OPEN THAT!!
2645 J:*WAIT
2650 *CABOP
2655 A:=$ROOM
2660 M:M2 ,M8 ,35 ,
2665 JM:*CAB1,*CAB2,*CAB3
2670 J:*NOTHERE
2675 *CAB1
2680 J(@B1561=1):*ALREADYOPEN
2685 C:@B1561=1
2690 C:@B1542=2
2695 U:*CP
2700 U:*OPCAB
2705 J:*OK
2710 *CAB2
2715 J(@B1563=1):*ALREADYOPEN
2720 C:@B1563=1
2725 C:@B1544=8
2730 C:@B1545=8
2735 U:*CP
2740 U:*OPCAB
2745 J:*OK
2750 *CAB3
2755 J(@B1570=1):*ALREADYOPEN
2760 C:@B1570=1
2765 C:@B1553=35
2770 U:*MCP
2775 U:*OPCAB
2780 J:*OK
2785 *FUSEBOP
2790 J(#R<>5):*NOTHERE
2795 J(@B1555<>5):*NOTHERE
2800 J(#R=5):*LOOK
2805 J:*OK
2810 *BOOKCOP
2815 J(#R<>7):*NOTHERE
2820 J(@B1562=1):*ALREADYOPEN
2825 C:@B1562=1
```

```
2830 U:*BOOKCPIC
2835 C:#S=11
2840 J:*OK
2845 *DOOP
2850 A:=$ROOM
2855 M:M4 ,18 ,19 ,20 ,34 ,
2860 JM:*D4,*D18,*D19,*D20,*D34
2865 J:*OK
2870 *D4
2875 J(@B1559<>1):*ME1
2880 J:*ALREADYOPEN
2885 *D18
2890 J(@B1548<>0):*ME2
2895 C:#S=21
2900 C:@B1566=1
2905 J:*OK
2910 *D19
2915 J(@B1547<>0):*ME3
2920 C:#S=22
2925 C:@B1564=1
2930 J:*OK
2935 *D20
2940 J(@B1546<>0):*ME4
2945 C:#N=17
2950 C:@B1565=1
2955 J:*OK
2960 *D34
2965 J(@B1554<>0):*ME5
2970 C:#N=30
2975 C:@B1569=1
2980 J:*OK
2985 *OPCAB
2990 C:$OBJECT= OPEN$OBJECT
2995 E:
3000 *ME1
3005 T:↖ELECTRICITY IS NEEDED FOR THAT
!!
3010 J:*WAIT
3015 *ME2
3020 T:↖YOU NEED THE BLUE KEY!!
3025 J:*WAIT
3030 *ME3
3035 T:↖IT'S STUCK!!
3040 J:*WAIT
3045 *ME4
3050 T:↖YOU NEED THE BROWN KEY!!
3055 J:*WAIT
3060 *ME5
3065 T:↖YOU NEED THE RED KEY!!
3070 J:*WAIT
```



```
3075 *ALREADYOPEN
3080 T:↖IT'S ALREADY OPEN!!
3085 J:*WAIT
3090 *ALREADYOUT
3095 T:↖IT'S ALREADY OUT!!
3100 J:*WAIT
3105 *ME6
3110 T:↖YOU NEED SOME WATER!!
3115 J:*WAIT
3120 *LIGHT
3125 M:FLA
3130 TN:↖I CAN'T LIGHT THAT!!!
3135 JN:*WAIT
3140 J(@B1541<>0):*DONTHAVE
3145 J(@B1558=1):*LIT
3150 C:@B1558=1
3155 J:*OK
3160 *LIT
3165 T:↖IT'S ALREADY LIT!!
3170 J:*WAIT
3175 *USE
3180 M:FUS,DRU,KEY,CRO
3185 JM:*USEFUS,*USEDRU,*USEKEY,*USECR
O
3190 J:*CANTDO
3195 *USEFUS
3200 J(@B1542<>0):*DONTHAVE
3205 J(#R<>5):*ME7
3210 C:@B1559=1
3215 C:@B1542=100
3220 T:↖THE LIGHTS COME ON!!
3225 J:*WAIT
3230 *USEDRU
3235 J(@B1553<>0):*DONTHAVE
3240 J(@B1552<>0):*ME8
3245 C:@B1571=1
3250 T:↖THE DRUGS ARE IN THE BANANA!!
3255 J:*WAIT
3260 *USEKEY
3265 J(@B1546=0):*OK
3270 J(@B1548=0):*OK
3275 J(@B1554=0):*OK
3280 J:*DONTHAVE
3285 *USECRO
3290 J(@B1547<>0):*DONTHAVE
3295 J(#R<>19):*ME7
3300 C:@B1564=1
3305 J:*OK
3310 *ME7
3315 T:↖YOU CANT DO THAT HERE!!
```

```
3320 J:*WAIT
3325 *ME8
3330 T:↖IN WHAT!!
3335 J:*WAIT
3340 *CANTDO
3345 T:↖YOU CAN'T DO THAT!!
3350 J:*WAIT
3355 *PUSH
3360 M:BOOKC,GOR
3365 JM:*BOOKCOP,*PUSHGO
3370 T:↖THAT DOESN'T HELP!!
3375 J:*WAIT
3380 *PUSHGO
3385 J(#R<>28):*NOTHERE
3390 J:*DEAD2
3395 *THROW
3400 M:BAN,WAT,BIS.BON
3405 JM:*THRBAN,*THRWAT,*THRBIS,*THRBO
N
3410 T:↖WHY DO YOU WANT TO THROW THAT!
!
3415 J:*WAIT
3420 *THRBAN
3425 J(@B1552<>0):*DONTHAVE
3430 C:@B1552=#R
3435 J(#R=28):*BANGOR
3440 J:*OK
3445 *BANGOR
3450 T:↖THE GORILLA EATS THE BANANA!!
3455 C:@B1552=100
3460 J(@B1571=1):*SLEEP
3465 J:*WAIT
3470 *SLEEP
3475 C:@B1568=1
3480 U:*GORILLAPIC
3485 C:#E=29
3490 C:$OBJECT= SLEEPING GORILLA
3495 C:@B1552=100
3500 J:*P
3505 *LOOK
3510 C(#R=20):@B1549=#R
3515 J(#R=20):*OK
3520 T(#R=5):↖THERE ARE FUSES MISSING!
!
3525 J(#R=5):*WAIT
3530 T:↖I SEE NOTHING SPECIAL!!
3535 J:*WAIT
3540 *GO
3545 M:CHI,FIR,SHA,GOR,DOG,CHU
3550 JM:*GOFIR,*GOFIR,*GOSHA,*GOGOR,*G
```

```
ODOG,*GOCHU
3555 J:*CANTGO
3560 *GOFIR
3565 A:=$ROOM
3570 M:26 ,32 ,
3575 JN:*NOTHERE
3580 J(@B1567<>1):*DEAD5
3585 C(#R=26):#R=25
3590 C(#R=32):#R=31
3595 J:*CR
3600 *GOGOR
3605 J(#R<>28):*NOTHERE
3610 J(@B1568<>1):*DEAD2
3615 J:*OK
3620 *GODOG
3625 J(#R<>9):*NOTHERE
3630 J(@B1557=0):*DEAD3
3635 J:*NOTHERE
3640 *GOCHU
3645 J(#R<>11):*ME7
3650 C:#R=20
3655 C:#F=#F-1
3660 J:*CR
3665 *GOSHA
3670 J(#R<>22):*ME7
3675 C:#R=27
3680 J:*CR
3685 *PRESSKEY
3690 T:          PRESS SPACE BAR TO CONTIN
UE    \
3695 C:@B764=255
3700 *LOOP2
3705 J(@B764=33):*CR
3710 J:*LOOP2
3715 *THRWAT
3717 J(@B1567=1):*ALREADYOUT
3718 J(@B1550<>0):*ME6
3720 C:@B1550=23
3725 J(#R<>26):*OK
3730 C:@B1567=1
3735 C:$OBJECT= FIREPLACE
3740 U:*FIREPIC
3745 J:*OK
3750 *THRBIS
3755 J(@B1544<>0):*DONTHAVE
3760 C:@B1544=#R
3765 J(#R<>9):*OK
3770 T:↰THE DOG EATS THE BISCUIT!!
3775 C:@B1544=8
3780 J:*WAIT
```

```
3785 *THRBON
3790 J(@B1545<>0):*DONTHAVE
3795 C:@B1545=#R
3800 J(#R<>9):*OK
3805 C:@B1557=1
3810 C:#W=7
3815 C:@B1545=100
3820 T:↖THE DOG LEAVES TO BURY THE BON
E!!
3825 C:$OBJECT= NOTHING
3830 U:*DOGPIC
3835 J:*WAIT
3840 *WEAR
3845 M:GLO
3850 TN:↖I CAN'T WEAR THAT!!!
3855 JN:*WAIT
3860 J(@B1549<>0):*DONTHAVE
3865 C:@B1549=50
3870 J:*OK
3875 *BR
3880 GR:CLEAR;PEN YELLOW
3885 GR:GOTO74,44;TURNTO180;DRAW23;TUR
N45;DRAW71;TURN135;FILL23;TURN45;FILL7
0
3890 GR:GOTO-50,-6;DRAW37
3895 GR:GOTO-76,-30;TURNTO0;FILL23;TUR
N45;FILL72
3900 GR:PEN ERASE;GOTO50,21;DRAW40
3905 GR:GOTO-75,-6;TURN45;DRAW26
3910 GR:GOTO-23,45;TURN90;DRAW24
3915 GR:GOTO23,-6;DRAW24
3920 E:
3925 *SR
3930 GR:CLEAR;PEN YELLOW
3935 GR:GOTO56,44;TURNTO180;DRAW23;TUR
N45;DRAW41;TURN135;FILL23;TURN45;FILL4
0
3940 GR:GOTO-6,16;DRAW6
3945 GR:GOTO-32,-9;TURNTO0;FILL24;TURN
45;FILL41
3950 GR:PEN ERASE;GOTO32,21;DRAW40
3955 GR:GOTO-31,16;TURN45;DRAW26
3960 GR:GOTO26,16;TURN90;DRAW24
3965 GR:GOTO-1,45;DRAW24;GOTO-32,-9
3970 E:
3975 *HW
3980 GR:CLEAR;PEN YELLOW
3985 GR:GOTO74,44;TURNTO180;DRAW23;TUR
N90;DRAW121;TURN90;FILL23
3990 GR:GOTO46,16;TURN180;DRAW23;TURN9
```

```
0;DRAW121;TURN90;FILL23
3995 GR:TURN45;GOTO-50,17;DRAW4
4000 GR:GOTO46,-7;DRAW38
4005 E:
4010 *CE
4015 GR:PEN YELLOW;TURNTO0;GOTO46,-7;F
ILL24;TURN45;FILL4;PEN ERASE
4020 GR:TURN-45;GO1;TURN45;DRAW38
4025 GR:TURN-45;GOTO45,-7;DRAW24
4030 E:
4035 *CW
4040 GR:PEN YELLOW;TURNTO45;GOTO-75,17
;FILL38
4045 GR:PEN ERASE;GO3;TURN135;DRAW29;T
URN90;DRAW30
4050 E:
4055 *ELBOW
4060 GR:CLEAR;PEN YELLOW
4065 GR:TURNTO180;GOTO44,10;DRAW23;TUR
N45;DRAW23;TURN135;DRAW16;FILL7;TURN18
0;DRAW23;TURN90
4070 GR:DRAW104;TURN90;FILL23
4075 GR:PEN ERASE;TURN90;DRAW46;TURN-9
0;PEN YELLOW;FILL16
4080 GR:TURN90;GO23;TURN-45;GO1;DRAW22
;TURN-45;DRAW23;TURN-135;FILL53
4085 GR:GOTO-22,23;DRAW44;GOTO-22,48;F
ILL76
4090 GR:TURN-135;PEN ERASE;DRAW104;TUR
N-45;DRAW23
4095 GR:TURN-135;DRAW74;TURN135;DRAW52

4100 GR:TURN-45;GOTO-21,24;DRAW23;GOTO
28,-29;DRAW23
4105 E:
4110 *SOUTHDOOR
4115 A:=#R
4120 M:7
4125 JY:*SOUTHARROW
4130 GR:PEN ERASE;TURNTO0;GOTO-40,-30;
2(DRAW21;TURN90;DRAW15;TURN90);GOTO-28
,-20;4(DRAW1;TURN90)
4135 J:*SOUTHARROW
4140 *SOUTHARROW
4145 GR:PEN RED;TURNTO90;GOTO61,-30;DR
AW4;TURN-90;DRAW2;TURN-90;DRAW4;TURN90
;DRAW2;TURN90;DRAW4
4150 GR:TURN90;GOTO63,-20;DRAW4;TURN13
5;DRAW3;TURN90;GOTO63,-24;DRAW3
4155 E:
```

```
4160 *NORTHDOOR
4165 A:=$ROOM
4170 M:M6 ,13 ,22 ,27 ,
4175 CY:#X=10
4180 CY:#Y=20
4185 JY:*ELEDOOR
4190 GR:PEN ERASE;TURNTO0;GOTO8,20;2(D
RAW21;TURN90;DRAW15;TURN90);GOTO20,30;
4(DRAW1;TURN90)
4195 J:*NORTHARROW
4200 *NORTHARROW
4205 GR:PEN RED;GOTO63,-18;DRAW4;TURN1
35;DRAW3;TURN90;GOTO63,-14;DRAW3
4210 GR:TURNTO0;GOTO61,-12;DRAW4;TURN1
35;DRAW5;TURN-135;DRAW4
4215 E:
4220 *EASTDOOR
4225 A:=$ROOM
4230 M:M3 ,M9 ,17 ,18 ,29 ,
4235 JY:*EASTARROW
4240 GR:PEN ERASE
4245 GR:TURNTO0;GOTO48,-8;2(DRAW21;TUR
N45;DRAW13;TURN135);GOTO54,8;2(DRAW1;T
URN45;DRAW1;TURN135)
4250 J:*EASTARROW
4255 *EASTARROW
4260 GR:PEN RED;TURNTO90;GOTO64,-19;DR
AW6;TURN135;DRAW3;TURN90;GOTO70,-19;DR
AW3
4265 GR:TURNTO270;GOTO75,-21;DRAW3;TUR
N90;DRAW4;TURN90;DRAW3;GOTO73,-19;DRAW
1
4270 E:
4275 *WESTDOOR
4280 A:=$ROOM
4285 M:M4 ,10 ,18 ,19 ,34 ,
4290 JY:*WESTARROW
4295 M:26 ,32 ,
4300 EY:
4305 GR:PEN ERASE
4310 GR:TURNTO0;GOTO-48,-5;2(DRAW21;TU
RN45;DRAW13;TURN135);GOTO-42,10;(DRAW1
;TURN45;DRAW1;TURN135)
4315 J:*WESTARROW
4320 *WESTARROW
4325 GR:PEN RED;TURNTO270;GOTO62,-19;D
RAW6;TURN135;DRAW3;TURN90;GOTO56,-19;D
RAW3
4330 GR:TURNTO180;GOTO54,-17;DRAW4;TUR
N135;DRAW3;TURN-90;DRAW3;TURN135;DRAW4
```

```
4335 E:
4340 *HD
4345 A:=$ROOM
4350 M:ROOM4 ,ROOM10 ,ROOM19 ,
4355 CY:#X=-25
4360 CY:#Y=-8
4365 JY:*ELEDOOR
4370 GR:PEN ERASE;TURNTO0;GOTO-20,-8;2
(DRAW21;TURN90;DRAW15;TURN90);GOTO-8,2
;4(DRAW1;TURN90)
4375 J:*SOUTHARROW
4380 *ELEDOOR
4385 GR:PEN ERASE
4390 GR:TURNTO0;GOTO#X,#Y;3(DRAW20;TUR
N90);DRAW10;TURN90;DRAW20;GOTO#X+22,#Y
+10;GOTO#X+22,#Y+12
4395 J(#X=-25):*SOUTHARROW
4400 J:*NORTHARROW
4405 *DOWNARROW
4410 A:=#R
4415 M:11 ,
4420 EY:
4425 GR:PEN RED;TURNTO180;GOTO-72,34;D
RAW4;TURN135;DRAW3;TURN90;GOTO-72,30;D
RAW3
4430 GR:GOTO-74,28;TURN45;11(DRAW1;TUR
N18);TURNTO0;DRAW6;PEN ERASE
4435 E:
4440 *UPARROW
4445 GR:PEN RED;TURNTO0;GOTO-72,36;DRA
W4;TURN135;DRAW3;TURN90;GOTO-72,40;DRA
W3
4450 GR:TURNTO180;GOTO-70,46;3(DRAW4;T
URN90);PEN ERASE
4455 E:
4460 *PL
4465 C:#X=-24
4470 C:#Y=16
4475 A:=$ROOM
4480 M:M3 ,M4 ,M6 ,10 ,13 ,17 ,18 ,19
,22 ,25 ,27 ,31 ,34 ,35 ,
4485 CY:#X=-54
4490 CY:#Y=-12
4495 M:M9 ,29 ,
4500 CY:#X=10
4505 CY:#Y=24
4510 E:
4515 *FLASHPIC
4520 GR(#L=1):PEN BLUE
4525 GR(#L=0):PEN ERASE
```

```
4530 GR:TURNTO0;GOTO#X,#Y-1
4535 GR:2(DRAW1;TURN90;DRAW3;TURN90)
4540 GR(#L=1):PEN YELLOW
4545 GR(#L=0):PEN ERASE
4550 GR:GOTO#X+3,#Y;TURN45;DRAW3;TURN1
35;DRAW4;TURN135;DRAW3
4555 E:
4560 *FUSEPIC
4565 GR(#L=1):PEN RED
4570 GR(#L=0):PEN ERASE
4575 GR:TURNTO315;GOTO#X+9,#Y-1;3(DRAW
1;TURN135)
4580 E:
4585 *PICPIC
4590 GR:PEN RED;TURNTO 0
4595 GR(@B1555=#R):PEN BLUE
4600 GR:GOTO-48,-2;2(DRAW15;TURN45;DRA
W15;TURN135);PEN BLUE
4605 GR(@B1555=#R):PEN YELLOW
4610 GR:GOTO -47,4
4615 C:#A=0
4620 *SPIRAL
4625 GR:2(DRAW#A;TURN45;DRAW#A*2;TURN1
35)
4630 C:#A=#A+2
4635 J(#A<8):*SPIRAL
4640 E:
4645 *BISPIC
4650 GR(#L=1):PEN RED
4655 GR(#L=0):PEN ERASE
4660 GR:TURNTO45;GOTO#X+13,#Y-1;4(DRAW
1;TURN90)
4665 E:
4670 *KEYPIC
4675 A:=$ITEM
4680 M:BRO,RED,BLUE
4685 JM:*BROWNPIC,*REDPIC,*BLUEPIC
4690 *BROWNPIC
4695 GR:PEN YELLOW
4700 C:#K=#X+40
4705 J:*KEYPIC2
4710 *REDPIC
4715 GR:PEN RED
4720 C:#K=#X+24
4725 J:*KEYPIC2
4730 *BLUEPIC
4735 GR:PEN BLUE
4740 C:#K=#X+32
4745 J:*KEYPIC2
4750 *KEYPIC2
```

```
4755 GR(#L=0):PEN ERASE
4760 GR:TURNTO0;GOTO#K,#Y-1;4(DRAW2;TU
RN90);TURNTO90;GOTO#K+3,#Y;DRAW3;TURN9
0;DRAW1;GOTO#K+4,#Y;DRAW1
4765 E:
4770 *BONEPIC
4775 GR:PEN BLUE
4780 GR(#L=0):PEN ERASE
4785 GR:TURNTO180;GOTO#X+17,#Y;2(DRAW2
;TURN90;DRAW1;TURN90);GO1;TURN-90;DRAW
4
4790 GR:2(TURN90;DRAW1);TURN90;DRAW2;2
(TURN90;DRAW1)
4795 E:
4800 *CROPIC
4805 GR:PEN YELLOW
4810 GR(#L=0):PEN ERASE
4815 GR:TURNTO325;GOTO#X,#Y-8;DRAW2;TU
RNTO90;DRAW5;TURN-135;DRAW2
4820 E:
4825 *GLOVEPIC
4830 GR:PEN RED
4835 GR(#L=0):PEN ERASE
4840 GR:TURNTO0;GOTO#X+10,#Y-8;2(DRAW2
;TURN90;DRAW4;TURN90);TURN-45;DRAW3
4845 GR:GOTO#X+10,#Y-6;TURNTO0;DRAW2;G
OTO#X+12,#Y-6;DRAW2;GOTO#X+14,#Y-6;DRA
W1
4850 E:
4855 *BUCKETPIC
4860 GR:PEN YELLOW;TURNTO 0
4865 GR(#L=0):PEN ERASE
4870 GR:GOTO#X+17,#Y-8;DRAW3;6(TURN30;
DRAW1);DRAW3;TURN90;DRAW4;PENBLUE
4875 GR(#L=0):PEN ERASE
4880 GR:GOTO#X+18,#Y-6;TURNTO90;DRAW2
4885 E:
4890 *DRUGPIC
4895 GR:PEN RED
4900 GR(#L=0):PEN ERASE
4905 GR:GOTO#X+24,#Y-6;GOTO#X+26,#Y-7;
GOTO#X+28,#Y-8
4910 GR:PEN BLUE
4915 GR(#L=0):PEN ERASE
4920 GR:GOTO#X+24,#Y-8;GOTO#X+30,#Y-6;
GOTO#X+27,#Y-7
4925 E:
4930 *BANANAPIC
4935 GR:PEN YELLOW
4940 GR(#L=0):PEN ERASE
```

```
4945 GR:GOTO#X+32,#Y-8;TURNTO90;DRAW3;
TURN-30;DRAW2
4950 E:
4955 *CP
4960 GR:PEN RED;TURNTO0;GOTO10,25
4965 GR:4(DRAW 15;TURN 90)
4970 U:*CHECKCABPIC
4975 GR(#O=1):PEN YELLOW
4980 GR:GOTO17,26;DRAW14;GOTO15,32;GOT
O19,32
4985 GR(#O=1):PEN RED;GOTO10,25;TURNTO
225;DRAW5;TURN135;DRAW15;TURN45;DRAW5
4990 GR(#O=1):TURN45;DRAW15;TURN45;DRA
W5;TURN45;DRAW15;TURN135;DRAW5
4995 E:
5000 *CHECKCABPIC
5005 A:=#R
5010 M:2 ,8 ,
5015 JM:*CAB2PIC,*CAB8PIC
5020 E:
5025 *CAB2PIC
5030 C:#O=0
5035 C(@B1561=1):#O=1
5040 E:
5045 *CAB8PIC
5050 C:#O=0
5055 C(@B1563=1):#O=1
5060 E:
5065 *BOOKCPIC
5070 GR:PEN BLUE;GOTO-30,-6;TURNTO0
5075 GR(@B1562=0):2(DRAW3;TURN90;DRAW3
2;TURN90);FILL3
5080 GR(@B1562=1):GOTO-27,-6;PEN RED;T
URN67;DRAW26;TURN-90;DRAW3;TURN-90;FIL
L32
5085 E:
5090 *MCP
5095 GR:PEN RED;GOTO-4,15;TURNTO45;2(D
RAW3;TURN45;DRAW15;TURN135);FILL3
5100 GR(@B1570=1):GOTO-2,18;DRAW9
5105 E:
5110 *GORILLAPIC
5115 GR:PEN YELLOW;TURNTO270
5120 GR(@B1568=1):PEN ERASE
5125 GR:GOTO32,4;DRAW1;TURN90;DRAW2;TU
RN-45;DRAW3;TURN90;DRAW1;TURN-45;DRAW3
;TURN90;DRAW2
5130 GR:TURN90;DRAW2;TURNTO90;DRAW1;TU
RN-90;DRAW3;TURN-90;DRAW4;2(TURN90;DRA
W1);TURN-90
```

```
5135 GR:DRAW2;TURN-90;DRAW4;2(TURN-90;
DRAW2);TURN90;DRAW1;TURN90;DRAW5;TURN-
90;DRAW3
5140 GR:TURN-90;DRAW1;TURN-90;DRAW2;TU
RN90;DRAW2;TURN90;DRAW4;GOTO27,8;TURNT
O225;DRAW3;TURN-45
5145 GR:DRAW2;TURN90;DRAW1;GOTO28,15;G
OTO30,8;GOTO25,15;GOTO31,15
5150 GR:GOTO29,12;DRAW2;GOTO29,11;DRAW
2;GOTO29,10;DRAW2;GOTO29,9;DRAW2
5155 GR(@B1568=1):PEN BLUE;TURNTO0;GOT
O18,4;DRAW1;TURN90;DRAW3;TURN-135;DRAW
3
5160 GR(@B1568=1):PEN YELLOW;TURN90;DR
AW1;TURN45;DRAW4;TURN45;DRAW3;TURNTO0;
4(DRAW3;TURN90)
5165 GR(@B1568=1):TURN-90;DRAW5;TURN45
;FILL3;GOTO27,8;TURN135;DRAW1;GOTO27,7
;DRAW1
5170 GR(@B1568=1):PEN BLUE;GOTO25,7;TU
RN45;DRAW3;TURN-45;DRAW3
5175 E:
5180 *DOGPIC
5185 GR:PEN YELLOW;TURNTO0
5190 GR(@B1557=1):PEN ERASE
5195 GR:GOTO-40,-4;DRAW1;2(DRAW2;TURN9
0;DRAW4;TURN90);GO2;TURN-45;DRAW 3
5200 GR:GOTO-33,-2;DRAW3;GOTO-33,-2;TU
RN-45;DRAW6;GOTO-35,-1;TURN-90;GOTO-36
,-3;DRAW1
5205 E:
5210 *FIREPIC
5215 C:#X=-48
5220 C:#Y=-5
5225 A:=$ROOM
5230 M:25 ,31 ,
5235 CY:#X=36
5240 CY:#Y=0
5245 GR:PEN RED;TURNTO0;GOTO#X,#Y;2(DR
AW21;TURN45;DRAW21;TURN135)
5250 GR:GOTO#X+4,#Y+4;2(DRAW18;TURN45;
DRAW14;TURN135);GOTO#X,#Y+23;TURN45;DR
AW20
5255 C:#P=#Y+3
5260 *BRICKS
5265 GR:GOTO#X+1,#P;DRAW3
5270 C:#P=#P+3
5275 J(#P<16):*BRICKS
5280 GR:PENBLUE;GOTO#X+6,#Y+8;TURNTO0;
DRAW1;TURN90;DRAW2;TURN-90;DRAW1
```

```
5285 GR:GOTO#X+10,#Y+12;DRAW1;TURN-90;
DRAW4;TURN180;DRAW7;TURN-90;DRAW1
5290 GR:PEN RED
5295 GR(@B1567=1):PEN YELLOW
5300 GR:GOTO#X+5,#Y+15;TURNTO45;DRAW3;
TURN90;DRAW3;TURNTO45;GO2;DRAW3
5305 E:
5310 *SINKPIC
5315 GR:PEN RED;TURNTO0;GOTO-48,5;2(DR
AW2;TURN90;DRAW6;TURN90);GO1;TURN90;DR
AW6;TURN-90;GO1;TURN-90
5320 GR:DRAW6;TURN90;GO1;PEN BLUE;2(TU
RN45;DRAW6);TURN135;DRAW6;2(TURN45;DRA
W5;TURN135;DRAW5)
5325 GR:2(TURN45;DRAW3;TURN135;DRAW3);
2(TURN45;DRAW2;TURN135;DRAW2);TURN45;D
RAW2
5330 GR:PEN RED;GOTO-41,8;TURNTO45;DRA
W6;GOTO-41,7;DRAW6
5335 GR:GOTO-41,6;DRAW5;GOTO-41,5;DRAW
5
5340 GR:GOTO-41,5;TURNTO180;DRAW6;GOTO
-37,8;DRAW6
5345 E:
5350 *INITIALIZE
5355 GR:QUIT
5360 C:@B764=255
5365 C:@B1373=0 [NO TEXT WINDOW
5370 C:@B1374=2 [GRAPHICS MODE 2
5375 WRITE:S,
5380 POS:7,4
5385 WRITE:S,eSCApE
5390 POS:3,8
5395 WRITE:S,by jack hardy
5400 CLOSE:S
5405 C:#L=1
5410 C:#R=1
5415 C:#F=4
5420 C:$ROOM=ROOM1
5425 C:@B1541=1
5430 C:@B1542=100
5435 C:@B1543=5
5440 C:@B1544=100
5445 C:@B1545=100
5450 C:@B1546=12
5455 C:@B1547=14
5460 C:@B1548=16
5465 C:@B1549=100
5470 C:@B1550=23
5475 C:@B1551=24
```

```
5480 C:@B1552=33
5485 C:@B1553=100
5490 C:@B1554=21
5495 C:@B1555=100
5500 C:@B1556=100
5505 C:#M=1557
5510 *LOOP1
5515 C:@B#M=0
5520 C:#M=#M+1
5525 J(#M<1572):*LOOP1
5530 C:$OBJECT= NOTHING
5535 C:$COMMA=,
5540 U:*DARK
5545 E:
5550 *DARK
5555 E(@B1559=1):
5560 GR:CLEAR
5565 C:@B710=0
5570 *COMMAND
5575 T:↖IT'S VERY DARK!!
5580 T:WHAT DO YOU WANT TO DO?
5585 A:
5590 M:FEE,LOO,GO ,MOV,LIG
5595 JM:*F,*L,*G,*G,*LI
5600 T:↖YOU CAN'T DO THAT NOW!!
5605 *WAIT2
5610 PA:140
5615 J:*COMMAND
5620 *F
5625 T(@B1541<>#R):↖YOU DON'T FEEL ANY
THING!!
5630 J(@B1541<>#R):*WAIT2
5635 T:↖YOU FIND A FLASHLIGHT!!
5640 C:@B1541=0
5645 J:*WAIT2
5650 *L
5655 T:↖IT IS TOO DARK TO SEE ANYTHING
!!
5660 J:*WAIT2
5665 *G
5670 T:↖IT'S DANGEROUS TO MOVE IN THE
DARK!!
5675 PA:240
5680 J:*DEAD4
5685 *LI
5690 T(@B1541<>0):↖YOU DON'T HAVE THE
FLASHLIGHT!!
5695 J(@B1541<>0):*WAIT2
5700 C:@B1558=1
5705 C:@B710=148
```

```
5710 C:@B709=42
5715 E:
5720 *DEAD1
5725 T:↖YOUR HANDS SLIP AND YOU FALL!!

5730 C:#P=31
5735 *REPEAT
5740 SO:#P
5745 C:#P=#P-1
5750 J(#P<>0):*REPEAT
5755 J:*DEATHMARCH
5760 *DEAD2
5765 T:↖THE GORILLA TEARS YOU LIMB FRO
M LIMB!!!
5770 J:*DEATHMARCH
5775 *DEAD3
5780 T:↖THE DOG ATTACKS!!!
5785 J:*DEATHMARCH
5790 *DEAD4
5795 T:↖YOU FALL AND BREAK YOUR NECK!!

5800 J:*DEATHMARCH
5805 *DEAD5
5810 T:↖YOU ARE COOKED ALIVE!!!
5815 J:*DEATHMARCH
5820 *DEATHMARCH
5825 C:#P=2
5830 SO:5
5835 PA:32
5840 U:*PAUSE
5845 SO:5
5850 PA:16
5855 U:*PAUSE
5860 SO:5
5865 PA:4
5870 U:*PAUSE
5875 SO:5
5880 PA:32
5885 U:*PAUSE
5890 SO:8
5895 PA:16
5900 U:*PAUSE
5905 SO:7
5910 PA:8
5915 U:*PAUSE
5920 SO:7
5925 PA:8
5930 U:*PAUSE
5935 SO:5
5940 PA:8
```

```
5945 U:*PAUSE
5950 SO:5
5955 PA:8
5960 U:*PAUSE
5965 SO:4
5970 PA:8
5975 U:*PAUSE
5980 SO:5
5985 PA:32
5990 SO:0
5995 *PLAYAGAIN
6000 T:↖     WANT TO TRY AGAIN? (Y/N)
6005 C:@B764=255
6010 *YNKEY
6015 J(@B764=43):*INITIALIZE
6020 J(@B764=35):*STOP
6025 J:*YNKEY
6030 *PAUSE
6035 SO:0
6040 PA:#P
6045 E:
6050 *STOP
6055 GR:QUIT
6060 C:@764=255
6065 T:↖THANKS FOR PLAYING!!!
6070 E:
6075 *WIN
6080 GR:QUIT
6085 C:@B1373=16 [TEXT WINDOW
6090 C:@B1374=2
6095 WRITE:5,
6100 POS:3,4
6105 WRITE:5,yOU EsCAPed!
6110 C:#P=0
6115 SO:20
6120 PA:2
6125 U:*PAUSE
6130 SO:22
6135 PA:2
6140 U:*PAUSE
6145 SO:24
6150 PA:2
6155 U:*PAUSE
6160 SO:26
6165 PA:16
6170 U:*PAUSE
6175 SO:22
6180 PA:2
6185 U:*PAUSE
6190 SO:26
```

```
6195 PA:32
6200 SO:0
6205 WRITE:S,
6210 WRITE:S, CONGRATULATIONS!!
6215 CLOSE:S
6220 J:*PLAYAGAIN
```

THE HUNTER

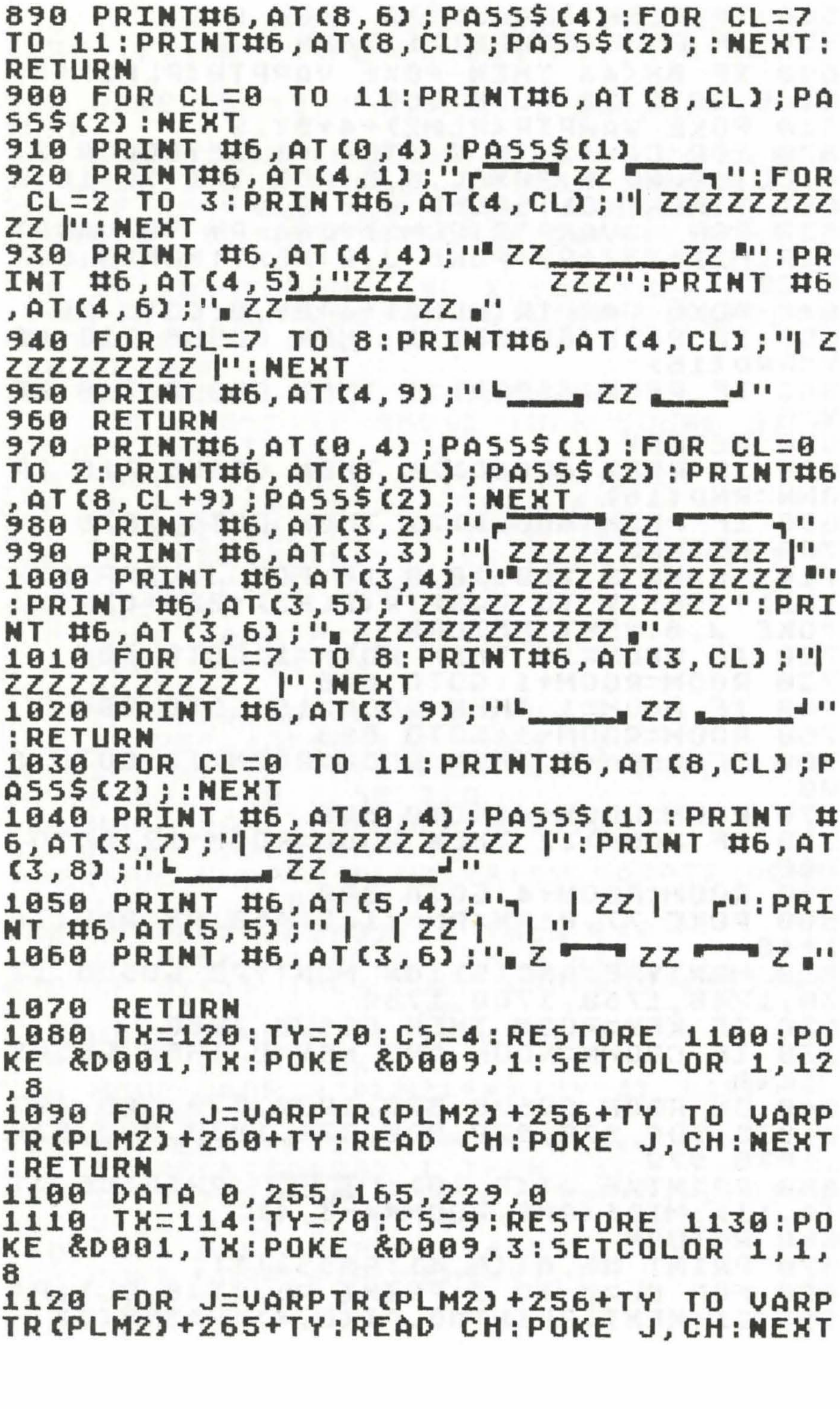
Objective: to locate the magic key which will reveal the hidden location of the crown. There are 16 rooms through which you must search to find the hidden crown. Only after the magic key is found will the crown be visible.

The program requires Atari Microsoft BASIC and 32K when loaded from disk or cassette.

```
10 GRAPHICS 2+16:GOSUB 1140:GOSUB 840
20 S=PEEK(&278):POKE &D01E,0:IF PEEK(&
D010)=0 THEN 290
30 IF KEY=ROOM THEN GOSUB 650
40 IF CROWN=ROOM AND KEY=0 THEN GOSUB
680
50 IF MON THEN GOSUB 200
60 IF PEEK(&D004)=3 THEN 1650
70 IF PEEK(&D00E)=1 THEN 1650
80 IF S=15 THEN 20
90 CX=(S=9 OR S=10 OR S=11)-(S=5 OR S=
6 OR S=7)
100 CY=(S=6 OR S=10 OR S=14)-(S=5 OR S
=9 OR S=13)
110 X=X+CX*3:POKE &D000,X
120 IF CY=+1 THEN MOVE VARPTR(PLM2)+12
8+(Y-2),VARPTR(PLM2)+128+Y,12:Y=Y+2
130 IF CY=-1 THEN MOVE VARPTR(PLM2)+12
8+Y,VARPTR(PLM2)+128+(Y-2),12:Y=Y-2
140 IF X>200 THEN X=44:GOSUB 250:PRINT
#6,AT(0,0);"↖":POKE &D000,X:GOSUB 720
150 IF X<44 THEN X=200:GOSUB 250:PPINT
#6,AT(0,0);"↖":POKE &D000,X:GOSUB 740
160 IF Y>106 THEN GOSUB 190:GOSUB 250:
PRINT #6,AT(0,0);"↖":Y=12:GOSUB 780:GO
SUB 1610
170 IF Y<12 THEN GOSUB 190:GOSUB 250:P
RINT #6,AT(0,0);"↖":Y=106:GOSUB 760:GO
SUB 1610
180 GOTO 20
190 FOR J=VARPTR(PLM2)+128+Y TO VARPTR
(PLM2)+137+Y:POKE J,0:NEXT:POKE &DP1D,
0:RETURN
200 IF X>RX THEN RX=RX+1
210 IF X<RX THEN RX=RX-1
220 IF Y<RY THEN MOVE VARPTR(PLM2)+384
+RY,VARPTR(PLM2)+384+(RY-1),10:RY=RY-1
```

```
230 IF Y>RY THEN MOVE VARPTR(PLM2)+384
+(RY-1),VARPTR(PLM2)+384+RY,10:RY=RY+1

240 POKE &D002,RX:SOUND 1,200,2,4,1:RE
TURN
250 FOR J=VARPTR(PLM2)+384+RY TO VARPT
R(PLM2)+392+RY:POKE J,0:NEXT:MON=0
260 IF ROOM=KEY THEN FOR J=VARPTR(PLM2
)+256+TY TO VARPTR(PLM2)+260+TY:POKE J
,0:NEXT
270 IF ROOM=CROWN AND KEY=0 THEN FOR J
=VARPTR(PLM2)+256+TY TO VARPTR(PLM2)+2
65+TY:POKE J,0:NEXT
280 RETURN
290 IF PEEK(&278)=15 THEN 50
300 AMO=AMO-1:IF AMO<1 THEN SOUND 0,10
,10,10,2:GOTO 50
310 SOUND 0,100,6,10,5:BX=X:BY=Y
320 IF PEEK(&278)=14 THEN 420
330 IF PEEK(&278)=11 THEN 540
340 IF PEEK(&278)=13 OR PEEK(&278)=9 O
R PEEK(&278)=5 THEN 480
350 BX=BX+6
360 POKE VARPTR(PLM2)+4+BY,3
370 POKE &D004,BX
380 BX=BX+4
390 IF PEEK(&D000)<>0 THEN 640
400 IF PEEK(&D008)>1 THEN 610
410 IF BX>240 THEN POKE VARPTR(PLM2)+4
+BY,0:GOTO 20 ELSE 370
420 POKE VARPTR(PLM2)+4+BY,1
430 POKE &D004,BX+4
440 MOVE VARPTR(PLM2)+4+BY,VARPTR(PLM2
)+4+(BY-4),8:BY=BY-4
450 IF PEEK(&D000)<>0 THEN 640
460 IF PEEK(&D008)>1 THEN 610
470 IF BY<15 THEN POKE VARPTR(PLM2)+4+
BY,0:GOTO 20 ELSE 440
480 POKE VARPTR(PLM2)+4+BY,1
490 POKE &D004,BX+4
500 MOVE VARPTR(PLM2)+4+(BY-4),VARPTR(
PLM2)+4+BY,8:BY=BY+4
510 IF PEEK(&D000)<>0 THEN 640
520 IF PEEK(&D008)>1 THEN 610
530 IF BY>124 THEN POKE VARPTR(PLM2)+4
+BY,0:GOTO 20 ELSE 500
540 POKE VARPTR(PLM2)+4+BY,3
550 POKE VARPTR(PLM2)+4+Y,3
560 POKE &D004,BX
570 BX=BX-4
```

```
580 IF PEEK(&D000)<>0 THEN 640
590 IF PEEK(&D008)>1 THEN 610
600 IF BX<44 THEN POKE VARPTR(PLM2)+4+
BY,0:GOTO 20 ELSE 560
610 POKE VARPTR(PLM2)+4+BY,0
620 FOR CL=12 TO 0 STEP -1:SETCOLOR 2,
4,CL:SOUND 0,20*CL,6,8:FOR T=1 TO 10:N
EXT T:NEXT CL:SOUND 0,0,0,0
630 FOR J=VARPTR(PLM2)+384+RY TO VARPT
R(PLM2)+392+RY:POKE J,0:NEXT:MON=0:GOT
O 20
640 POKE VARPTR(PLM2)+4+BY,0:GOTO 20
650 IF PEEK(&D00D)>1 THEN GOSUB 710:KE
Y=RND(16)
660 IF PEEK(&D00D)=1 THEN GOSUB 710:KE
Y=0
670 RETURN
680 IF PEEK(&D00D)>1 THEN GOSUB 710:CR
OWN=RND(16)
690 IF PEEK(&D00D)=1 THEN GOTO 1830
700 RETURN
710 SOUND 0,200,10,8,10:FOR J=VARPTR(P
LM2)+256+TY TO VARPTR(PLM2)+256+CS+TY:
POKE J,0:NEXT:RETURN
720 IF ROOM=16 THEN ROOM=1:GOTO 800
730 ROOM=ROOM+1:GOTO 800
740 IF ROOM=1 THEN ROOM=16:GOTO 800
750 ROOM=ROOM-1:GOTO 800
760 IF ROOM<5 THEN ROOM=ROOM+12:GOTO 8
00
770 ROOM=ROOM-4:GOTO 800
780 IF ROOM>12 THEN ROOM=ROOM-12:GOTO
800
790 ROOM=ROOM+4:GOTO 800
800 POKE 77,0:RX=RND(140)+50:RY=RND(70
)+40
810 MONTYPE=RND(5):ON MONTYPE GOSUB 17
30,1740,1750,1760,1780
820 IF KEY=ROOM THEN GOSUB 1080
830 IF CROWN=ROOM AND KEY=0 THEN GOSUB
 1110
840 ON ROOM GOSUB 870,1030,970,900,103
0,970,900,870,970,900,870,1030,900,870
,1030,970
850 PRINT#6,AT(0,10);"ROOM":PRINT#6,AT
(0,11);MID$(RM$,ROOM*4-3,4);
860 RETURN
870 PRINT #6,AT(0,4);PASS$(1);
880 FOR CL=0 TO 3:PRINT #6,AT(8,CL);PA
SS$(2):NEXT:PRINT#6,AT(8,4);PASS$(3)
```



```
890 PRINT#6,AT(8,6);PASS$(4):FOR CL=7
TO 11:PRINT#6,AT(8,CL);PASS$(2);:NEXT:
RETURN
900 FOR CL=0 TO 11:PRINT#6,AT(8,CL);PA
SS$(2):NEXT
910 PRINT #6,AT(0,4);PASS$(1)
920 PRINT#6,AT(4,1);"┌▔▔▔▔▀ ZZ ▀▔▔▔▔┐":FOR
 CL=2 TO 3:PRINT#6,AT(4,CL);"| ZZZZZZZZ
ZZ |":NEXT
930 PRINT #6,AT(4,4);"▀ ZZ▁▁▁▁▁▁ZZ ▀":PR
INT #6,AT(4,5);"ZZZ      ZZZ":PRINT #6
,AT(4,6);"▄ ZZ▔▔▔▔▔▔ZZ ▄"
940 FOR CL=7 TO 8:PRINT#6,AT(4,CL);"| Z
ZZZZZZZZZ |":NEXT
950 PRINT #6,AT(4,9);"└▁▁▁▁▄ ZZ ▄▁▁▁▁┘"
960 RETURN
970 PRINT#6,AT(0,4);PASS$(1):FOR CL=0
TO 2:PRINT#6,AT(8,CL);PASS$(2):PRINT#6
,AT(8,CL+9);PASS$(2);:NEXT
980 PRINT #6,AT(3,2);"┌▔▔▔▔▔▀ ZZ ▀▔▔▔▔▔┐"
990 PRINT #6,AT(3,3);"| ZZZZZZZZZZZZ |"
1000 PRINT #6,AT(3,4);"▀ ZZZZZZZZZZZZ ▀"
:PRINT #6,AT(3,5);"ZZZZZZZZZZZZZZ":PRI
NT #6,AT(3,6);"▄ ZZZZZZZZZZZZ ▄"
1010 FOR CL=7 TO 8:PRINT#6,AT(3,CL);"|
ZZZZZZZZZZZZ |":NEXT
1020 PRINT #6,AT(3,9);"└▁▁▁▁▁▄ ZZ ▄▁▁▁▁▁┘"
:RETURN
1030 FOR CL=0 TO 11:PRINT#6,AT(8,CL);P
ASS$(2);:NEXT
1040 PRINT #6,AT(0,4);PASS$(1):PRINT #
6,AT(3,7);"| ZZZZZZZZZZZZ |":PRINT #6,AT
(3,8);"└▁▁▁▁▁▄ ZZ ▄▁▁▁▁▁┘"
1050 PRINT #6,AT(5,4);"┐  | ZZ |  ┌":PRI
NT #6,AT(5,5);" |  | ZZ |  | "
1060 PRINT #6,AT(3,6);"▄ Z ▀▔▔▀ ZZ ▀▔▔▀ Z ▄"

1070 RETURN
1080 TX=120:TY=70:CS=4:RESTORE 1100:PO
KE &D001,TX:POKE &D009,1:SETCOLOR 1,12
,8
1090 FOR J=VARPTR(PLM2)+256+TY TO VARP
TR(PLM2)+260+TY:READ CH:POKE J,CH:NEXT
:RETURN
1100 DATA 0,255,165,229,0
1110 TX=114:TY=70:CS=9:RESTORE 1130:PO
KE &D001,TX:POKE &D009,3:SETCOLOR 1,1,
8
1120 FOR J=VARPTR(PLM2)+256+TY TO VARP
TR(PLM2)+265+TY:READ CH:POKE J,CH:NEXT
```

```
:RETURN
1130 DATA 0,16,56,16,254,170;214,170,2
54,0
1140 PRINT #6,AT(4,3);"the hunter":PRI
NT #6,AT(3,5);"BY JACK HARDY"
1150 LEL=1:LEVEL$=" NOVICE     PILOT
SERGEANT CAPTAIN COMMANDER"
1160 DIM PASS$(4)
1170 PASS$(1)="‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾ZZZ
ZZZZZZZZZZZZZZZZZ____________________"

1180 PASS$(2)="| ZZ |":PASS$(3)="▀ ZZ ▀":P
ASS$(4)="▄ ZZ ▄"
1190 RM$="iZZZiiZZiiiZivZZvZZZviZZviiZ
viiiixZZxZZZxiZZxiiZxiiixivZxvZZxviZ"
1200 ROOM=1
1210 RANDOMIZE:CROWN=RND(16):KEY=RND(1
6):IF CROWN=KEY THEN 1210
1220 FOR J=10 TO 0 STEP -1:FOR J1=20 T
O 0 STEP -2:SOUND 0,J1,10,J:NEXT J1
1230 FOR J1=120 TO 90 STEP -2:SOUND 1,
J,10,J:NEXT J1:NEXT J
1240 CHBAS=756:OPTION CHR2:ADDR%=VARPT
R(CHR2):PAGE%=(ADDR%/256)AND 255
1250 MOVE 57856,ADDR%,512:RESTORE 1390

1260 OF=0*8:FOR I=0 TO 7:READ CH:POKE
ADDR%+OF+I,CH:NEXT
1270 OF=3*8:FOR I=0 TO 7:READ CH:POKE
ADDR%+OF+I,CH:NEXT
1280 OF=5*8:FOR I=0 TO 7:READ CH:POKE
ADDR%+OF+I,CH:NEXT
1290 OF=9*8:FOR I=0 TO 7:READ CH:POKE
ADDR%+OF+I,CH:NEXT
1300 OF=11*8:FOR I=0 TO 15:READ CH:POK
E ADDR%+OF+I,CH:NEXT
1310 OF=15*8:FOR I=0 TO 7:READ CH:POKE
 ADDR%+OF+I,CH:NEXT
1320 OF=17*8:FOR I=0 TO 7:READ CH:POKE
 ADDR%+OF+I,CH:NEXT
1330 OF=26*8:FOR I=0 TO 7:READ CH:POKE
 ADDR%+OF+I,CH:NEXT
1340 OF=41*8:FOR I=0 TO 7:READ CH:POKE
 ADDR%+OF+I,CH:NEXT
1350 OF=50*8:FOR I=0 TO 7:READ CH:POKE
 ADDR%+OF+I,CH:NEXT
1360 OF=54*8:FOR I=0 TO 7:READ CH:POKE
 ADDR%+OF+I,CH:NEXT
1370 OF=56*8:FOR I=0 TO 7:READ CH:POKE
 ADDR%+OF+I,CH:NEXT
```

```
1380 OF=58*8:FOR I=0 TO 7:READ CH:POKE
 ADDR%+OF+I,CH:NEXT
1390 DATA 251,251,251,0,223,223,223,0
1400 DATA 3,3,3,3,3,3,255,255
1410 DATA 255,255,3,3,3,3,3,3
1420 DATA 0,0,0,0,0,0,3,3
1430 DATA 3,3,0,0,0,0,0,0,192,192,0,0,
0,0,0,0
1440 DATA 0,0,0,0,0,0,192,192
1450 DATA 255,255,192,192,192,192,192,
192
1460 DATA 192,192,192,192,192,192,255,
255
1470 DATA 0,126,24,24,24,24,126,0
1480 DATA 0,124,102,102,124,108,102,0
1490 DATA 0,102,102,102,102,60,24,0
1500 DATA 0,102,102,60,60,102,102,0
1510 DATA 0,0,0,0,0,0,0,0
1520 PRINT #6,AT(4,8);"select LEVEL":P
RINT #6,AT(4,11);"start TO PLAY";
1530 PRINT#6,AT(5,9);MID$(LEVEL$,LEL*9
-8,9):AMO=36-(LEL*4)
1540 IF PEEK(53279)=6 THEN 1570
1550 IF PEEK(53279)=5 THEN LEL=LEL+1:S
OUND 0,150,10,10,10:IF LEL>5 THEN LEL=
1
1560 FOR J=1 TO 100:NEXT:GOTO 1530
1570 X=120:Y=82
1580 OPTION PLM2:POKE 559,46:POKE &D00
C,1:POKE &D01D,3:POKE &D000,X:POKE &D0
08,1:POKE &26F,1
1590 SETCOLOR 0,9,8
1600 GOSUB 1610:POKE CHBAS,PAGE%:PRINT
#6,"↖":RETURN
1610 RESTORE 1630:FOR J=VARPTR(PLM2)+1
28+Y TO VARPTR(PLM2)+137+Y:READ CH:POK
E J,CH
1620 NEXT J
1630 DATA 0,16,16,16,56,40,124,254,56,
0
1640 POKE &D01D,3:RETURN
1650 RESTORE 1720:SETCOLOR 0,4,12:FOR
J=VARPTR(PLM2)+128+Y TO VARPTR(PLM2)+1
37+Y:READ CH:POKE J,CH:NEXT
1660 FOR CL=12 TO 0 STEP -1:SETCOLOR 0
,4,CL:SOUND 0,20*CL,6,8:FOR T=1 TO 10:
NEXT T:NEXT CL:SOUND 0,0,0,0
1670 FOR J=VARPTR(PLM2)+128+Y TO VARPT
R(PLM2)+137+Y:POKE J,0:NEXT:FOR TI=1 T
O 200:NEXT
```

```
1680 FOR J=VARPTR(PLM2)+384+RY TO VARP
TR(PLM2)+392+RY:POKE J,0:NEXT:POKE &D0
1D,0
1690 GRAPHICS 2+16:PRINT #6,AT(2,4);"Y
OU GOT ZAPPED!!"
1700 FOR J=15 TO 0 STEP -0.5:SOUND 0,6
0,12,J:SOUND 1,120,12,J:SOUND 2,60,12,
J:SOUND 3,120,12,J:NEXT
1710 GOTO 1870
1720 DATA 0,16,0,16,0,40,124,0,56,0
1730 SETCOLOR 2,6,8:RESTORE 1790:POKE
&D00A,1:GOTO 1770
1740 SETCOLOR 2,4,8:RESTORE 1800:POKE
&D00A,1:GOTO 1770
1750 SETCOLOR 2,3,8:RESTORE 1810:POKE
&D00A,0:GOTO 1770
1760 SETCOLOR 2,5,8:RESTORE 1820:POKE
&D00A,3
1770 POKE &D002,RX:FOR J=VARPTR(PLM2)+
384+RY TO VARPTR(PLM2)+392+RY:READ CH:
POKE J,CH:NEXT:MON=1
1780 RETURN
1790 DATA 0,66,66,90,255,219,255,231,0
1800 DATA 0,60,126,90,255,255,66,231,0

1810 DATA 0,60,126,90,255,255,195,0,0
1820 DATA 0,60,36,60,24,255,60,102,0
1830 PRINT #6,"↖":POKE CHBAS,224:PRINT
 #6,AT(2,3);"congratulations":PRINT #6
,AT(9,4);"a"
1840 PRINT #6,AT(1,5);"SUCCESSFUL MISS
ION"
1850 FOR J=15 TO 0 STEP -0.5:FOR J1=10
 TO 19:SOUND 0,J1,2,J:SOUND 1,J1*5,10,
J:NEXT J1:NEXT J
1860 FOR J=1 TO 500:NEXT:GOSUB 710
1870 PRINT #6,AT(0,10);"PRESS start  T
O play":IF PEEK(53279)=6 THEN GOSUB 19
0:RUN
1880 FOR J=1 TO 250:NEXT:PRINT#6,AT(6,
10);"option TO quit":IF PEEK(53279)=3
THEN END
1890 FOR J=1 TO 250:NEXT:GOTO 1870
```

TIME CRIME

Objective: to travel to other time zones, find and collect as much treasure as possible while avoiding being eaten by one of the many creatures present. There are helpful features added to the game. You can press the fire button and be drawn one step closer to the treasure, but the price you pay is $50 for each step. Also, you can see if there is a creature (normally invisible) in your location by pressing the joystick handle and the fire button at the same time. The price for this is $200. Of course, since you need money to use either of these features the first treasure will have to be found solely by searching for it.

The program requires Atari BASIC and 16K when used with cassette or 24K with disk.

```
10 GOSUB 3000
20 ? #6;"↰":POSITION 1,0:? #6;"$";LOOT

30 ON LOC GOSUB L(1),L(2),L(3),L(4),L(
5),L(6),L(7),L(8),L(9)
40 POSITION MX,MY:PUT #6,1
50 IF LOC=SHIP THEN POSITION SX,SY:PUT
 #6,251
60 TM=PEEK(19)/14:POKE 77,0
70 POSITION 14,0:? #6;MAXMIN-INT(TM);"
 MIN."
80 IF MAXMIN=TM THEN GOSUB 1000:GOTO 2
0
90 IF TM=(MAXMIN-0.5) THEN SOUND 1,0,4
,1
100 IF STRIG(0)=0 THEN GOSUB 210
110 S=STICK(0):NX=(S=5 OR S=6 OR S=7)-
(S=9 OR S=10 OR S=11)
120 NY=(S=5 OR S=9 OR S=13)-(S=6 OR S=
10 OR S=14):IF  NOT (NX OR NY) THEN 60

125 OX=MX:OY=MY:MX=MX+NX:MY=MY+NY
130 IF MX>19 THEN MX=0:GOTO 300
140 IF MX<0 THEN MX=19:GOTO 320
150 IF MY>11 THEN MY=1:GOTO 350
160 IF MY<1 THEN MY=11:GOTO 370
170 LOCATE MX,MY,Z:IF Z=251 THEN POSIT
ION OX,OY:? #6;" ";:GOSUB 1000:GOTO 60
175 IF Z<>32 THEN MX=OX:MY=OY:GOTO 60
```

```
180 IF MX=TX(LOC) AND MY=TY(LOC) THEN
GOSUB 250
185 POSITION OX,OY:? #6;" ";:POSITION
MX,MY:PUT #6,1:SOUND 0,150,6,4
190 SOUND 0,0,0,0:IF CX(LOC)=MX AND CY
(LOC)=MY THEN 460
200 GOTO 40
210 IF STICK(0)<>15 THEN 600
215 IF LOOT<50 THEN RETURN
217 LOOT=LOOT-50:POSITION 1,0:? #6;"$"
;LOOT;" "
218 IF TY(LOC)=25 THEN SOUND 3,10,2,8:
FOR T=1 TO 20:NEXT T:SOUND 3,0,0,0:GOT
O 245
220 OX=MX:OY=MY:OC=PEEK(709):POKE 709,
INT(RND(0)*16)*16+12:SOUND 3,200,10,8
222 IF MX>TX(LOC) THEN MX=MX-1
225 IF MY>TY(LOC) THEN MY=MY-1
230 IF MX<TX(LOC) THEN MX=MX+1
235 IF MY<TY(LOC) THEN MY=MY+1
240 POKE 709,OC:SOUND 3,0,0,0:POP :GOT
O 130
245 IF STRIG(0)=0 THEN 245
247 RETURN
250 POSITION TX(LOC),TY(LOC):PUT #6,13
7:FOR TME=1 TO 5:SOUND 0,10,10,6:FOR T
IME=1 TO 5:NEXT TIME
255 SOUND 0,0,0,0:FOR TIME=1 TO 20:NEX
T TIME:NEXT TME
260 TY(LOC)=25:LOOT=LOOT+INT(RND(0)*10
00*ZONE)+100:POSITION 1,0:? #6;"$";LOO
T:RETURN
300 LOC=LOC+1:IF LOC>9 THEN LOC=1
310 GOTO 20
320 LOC=LOC-1:IF LOC<1 THEN LOC=9
330 GOTO 20
350 LOC=LOC+3:IF LOC>9 THEN LOC=LOC-9
360 GOTO 20
370 LOC=LOC-3:IF LOC<1 THEN LOC=LOC+9
380 GOTO 20
400 FOR TME=1 TO 5:POSITION CX(LOC),CY
(LOC):? #6;"@";:SOUND 2,100,8,4
410 FOR TIME=1 TO 9:NEXT TIME:POSITION
 CX(LOC),CY(LOC):? #6;"'";
420 SOUND 2,0,0,0:FOR TIME=1 TO 9:NEXT
 TIME:NEXT TME:RETURN
460 GOSUB 400:POSITION 1,4:? #6;"A CRE
ATURE GOBBLED        YOU UP":GOTO 520
500 POSITION 2,4:? #6;"YOUR TIME MACHI
NE":? #6;"  LEFT WITHOUT YOU!"
```

```
510 POSITION 4,6:? #6;"LOOT $";LOOT
520 IF LOOT>HI THEN HI=LOOT
530 POSITION 2,7:? #6;"MOST LOOT $";HI

550 POSITION 3,9:? #6;"PLAY AGAIN? Y/N
█"
560 OPEN #1,4,0,"K:":GET #1,K:CLOSE #1

570 IF K<>78 AND K<>89 THEN 560
580 IF K=78 THEN POSITION 1,10:? #6;"T
HANKS FOR PLAYING":FOR TM=1 TO 500:NEX
T TM:END
590 GOSUB 3080:GOTO 20
600 IF LOOT<200 THEN RETURN
610 SOUND 0,200,10,8:IF CX(LOC) THEN G
OSUB 400
630 LOOT=LOOT-200:POSITION 1,0:? #6;"$
";LOOT;"  ":SOUND 0,0,0,0
640 IF STICK(0)<>15 THEN 640
650 RETURN
800 POSITION 2,2:? #6;"▥▥▥ ▥▥▥ ▥▥▥ ▥▥▥
":FOR BY=3 TO 5:POSITION 2,BY:? #6;"▥
    ▥ ▥     ▥":NEXT BY
810 POSITION 2,6:? #6;"▥▥▥▥▥▥▥ ▥▥▥▥▥▥▥
":FOR BY=7 TO 9:POSITION 2,BY:? #6;"▥
    ▥ ▥     ▥":NEXT BY
820 POSITION 2,10:? #6;"▥▥▥ ▥▥▥ ▥▥▥ ▥▥
▥":RETURN
840 POSITION 2,2:? #6;"▥▥▥▥▥▥▥   ▥▥▥▥▥
▥":GOSUB 970:GOSUB 980:GOSUB 990:RETUR
N
850 FOR BY=3 TO 4:POSITION 2,BY:? #6;"
▥":NEXT BY:FOR BY=8 TO 9:POSITION 2,BY
:? #6;"▥":NEXT BY
860 GOSUB 960:GOSUB 970:GOSUB 980:RETU
RN
870 FOR BY=3 TO 4:POSITION 17,BY:? #6;
"▥":NEXT BY:FOR BY=8 TO 9:POSITION 17,
BY:? #6;"▥":NEXT BY
880 GOSUB 960:GOSUB 980:GOSUB 990:RETU
RN
890 POSITION 2,10:? #6;"▥▥▥▥▥▥   ▥▥▥▥▥
▥▥":GOSUB 960:GOSUB 970:GOSUB 990:RETU
RN
900 POSITION 2,2:? #6;"▥▥▥▥▥▥   ▥▥▥▥▥▥
▥":FOR BY=3 TO 5:POSITION 7,BY:? #6;"▥
   ▥":NEXT BY
910 FOR BY=7 TO 9:POSITION 7,BY:? #6;"
▥   ▥":NEXT BY:POSITION 2,10:? #6;"▥▥▥
▥▥▥   ▥▥▥▥▥▥▥"
```

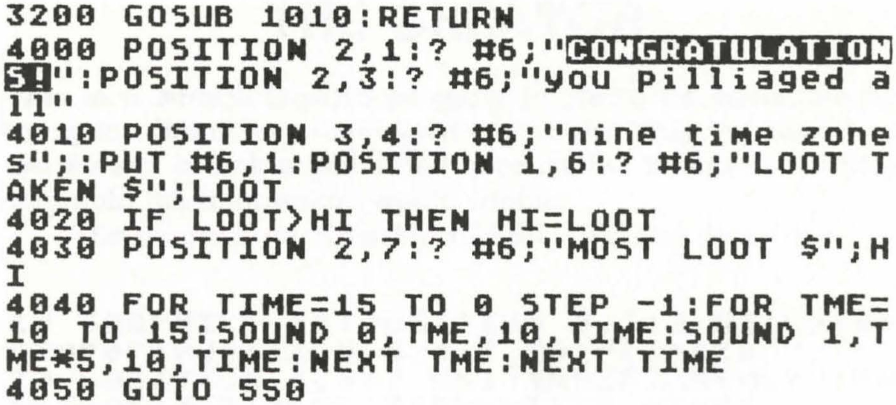
```
920 GOSUB 970:GOSUB 990:RETURN
930 FOR TREE=1 TO 10:TX=INT(RND(0)*18)
+1:TY=INT((RND(0)*4)+2)*2:POSITION TX,
TY:PUT #6,134
940 POSITION TX,TY-1:PUT #6,37:NEXT TR
EE:RETURN
950 FOR SWAMP=1 TO 10:TX=INT(RND(0)*17
)+1:TY=INT(RND(0)*9)+2:POSITION TX,TY:
? #6;"((":NEXT SWAMP:RETURN
955 GOSUB 950:GOSUB 930:RETURN
960 POSITION 2,2:? #6;"▦▦▦▦▦▦▦▦▦▦▦▦▦▦▦
▦":RETURN
970 FOR BY=3 TO 9:POSITION 17,BY:? #6;
"▦":NEXT BY:RETURN
980 POSITION 2,10:? #6;"▦▦▦▦▦▦▦▦▦▦▦▦▦▦
▦▦":RETURN
990 FOR BY=3 TO 9:POSITION 2,BY:? #6;"
▦":NEXT BY:RETURN
1000 IF MAXMIN<>TM THEN RETURN
1010 GRAPHICS 21:C=INT(RND(0)*16):SETC
OLOR 0,C,8:SETCOLOR 1,C,8:SETCOLOR 2,C
,4:COL=1
1020 FOR I=0 TO 20:COLOR COL
1030 X=I*2:Y=I:SOUND 1,I*5,10,8
1040 PLOT X,Y
1050 DRAWTO 79-X,Y:DRAWTO 79-X,47-Y:PL
OT 79-X-1,Y
1060 DRAWTO 79-X-1,47-Y:DRAWTO X,47-Y
1070 DRAWTO X,Y:PLOT X+1,47-Y:DRAWTO X
+1,Y
1080 COL=COL+0.5:IF COL+0.5>=4 THEN CO
L=COL-3
1090 NEXT I:IF (MX<>SX OR MY<>SY) THEN
 2000
1100 COLOR 0:PLOT 44,21:DRAWTO 36,21:P
LOT 39,19:DRAWTO 39,17:DRAWTO 41,17:DR
AWTO 41,19:PLOT 40,18:DRAWTO 40,23
1110 PLOT 39,22:DRAWTO 39,23:DRAWTO 37
,25:PLOT 41,22:DRAWTO 41,23:DRAWTO 43,
25
2000 FOR TME=1 TO 50:TEMP=PEEK(708):PO
KE 708,PEEK(709):POKE 709,PEEK(710):PO
KE 710,TEMP
2010 SOUND 0,TEMP*15,10,4:SOUND 1,PEEK
(708),10,4:FOR TIME=1 TO 10:NEXT TIME:
NEXT TME
2020 SOUND 0,0,0,0:SOUND 1,0,0,0:GRAPH
ICS 18:IF (MX<>SX OR MY<>SY) THEN 500
2025 ZONE=ZONE+1:IF ZONE=10 THEN 4000
2027 POSITION 4,4:? #6;"now entering":
```

```
POSITION 4,5:? #6;"TIME ZONE #";ZONE
2030 FOR SHU=9 TO 2 STEP -1:NL=INT(RND
(0)*SHU)+1:T=L(NL):L(NL)=L(SHU):L(SHU)
=T:NEXT SHU
2032 FOR CRE=1 TO 9:CY(CRE)=0:CX(CRE)=
0:NEXT CRE
2035 FOR TRE=1 TO 9:TX(TRE)=INT(RND(0)
*13)+3:TY(TRE)=INT(RND(0)*6)+4:NEXT TR
E
2036 FOR CRE=1 TO ZONE:CX(CRE)=INT(RND
(0)*13)+3:CY(CRE)=INT(RND(0)*6)+4:NEXT
 CRE
2040 ON ZONE GOSUB Z(1),Z(2),Z(3),Z(4)
,Z(5),Z(6),Z(7),Z(8),Z(9)
2045 POKE 708,INT(RND(0)*16)*16+8:POKE
 709,INT(RND(0)*16)*16+8:POKE 710,INT(
RND(0)*16)*16+8
2046 POKE 711,INT(RND(0)*16)*16+8
2050 LOC=INT(RND(0)*9)+1:POKE 712,INT(
RND(0)*16)*16+12:POKE 20,0:POKE 19,0:S
HIP=LOC:POKE 756,START/256:RETURN
2060 RESTORE 2081:FOR TIME=1 TO 7:READ
 X
2070 FOR Y=0 TO 7:READ Z:POKE X+Y+STAR
T,Z:NEXT Y:NEXT TIME:RETURN
2081 DATA 24,239,239,239,0,254,254,254
,0
2082 DATA 40,60,126,255,126,60,24,24,2
4
2083 DATA 48,24,24,24,24,24,24,24,60
2084 DATA 56,255,219,129,129,129,129,1
65,255
2085 DATA 64,16,40,68,130,8,20,34,65
2086 DATA 72,16,8,4,2,1,255,255,255
2087 DATA 256,0,0,255,255,255,255,0,0
2090 RESTORE 2111:FOR TIME=1 TO 7:READ
 X
2100 FOR Y=0 TO 7:READ Z:POKE X+Y+STAR
T,Z:NEXT Y:NEXT TIME:RETURN
2111 DATA 24,224,238,238,238,14,238,23
8,224
2112 DATA 40,189,90,189,24,24,24,24,24

2113 DATA 48,24,24,24,24,24,60,66,129
2114 DATA 56,170,255,213,129,129,171,2
55,85
2115 DATA 64,170,68,170,0,170,68,170,0

2116 DATA 72,0,0,60,126,126,126,60,0
2117 DATA 256,0,0,170,255,255,85,0,0
```

```
2130 RESTORE 2151:FOR TIME=1 TO 7:READ
 X
2140 FOR Y=0 TO 7:READ Z:POKE X+Y+STAR
T,Z:NEXT Y:NEXT TIME:RETURN
2151 DATA 24,255,255,255,255,255,255,2
55,255
2152 DATA 40,24,60,126,255,255,126,60,
24
2153 DATA 48,24,24,24,24,24,255,60,255

2154 DATA 56,24,36,66,129,129,66,36,24

2155 DATA 64,130,68,40,16,65,34,20,8
2156 DATA 72,0,32,112,36,14,68,224,64
2157 DATA 256,0,0,0,255,255,0,0,0
2160 RESTORE 2181:FOR TIME=1 TO 7:READ
 X
2170 FOR Y=0 TO 7:READ Z:POKE X+Y+STAR
T,Z:NEXT Y:NEXT TIME:RETURN
2181 DATA 24,102,255,255,126,126,255,2
55,102
2182 DATA 40,60,255,126,24,126,255,60,
24
2183 DATA 48,60,255,126,24,126,255,60,
24
2184 DATA 56,255,255,129,129,129,129,2
55,255
2185 DATA 64,56,238,0,28,119,0,56,238
2186 DATA 72,112,56,28,0,28,56,112,224

2187 DATA 256,0,0,0,255,255,0,0,0
2200 RESTORE 2221:FOR TIME=1 TO 7:READ
 X
2210 FOR Y=0 TO 7:READ Z:POKE X+Y+STAR
T,Z:NEXT Y:NEXT TIME:RETURN
2221 DATA 24,170,85,170,85,170,85,170,
85
2222 DATA 40,126,255,90,102,90,102,255
,24
2223 DATA 48,24,90,189,90,24,24,24,24
2224 DATA 56,60,102,195,129,129,195,10
2,60
2225 DATA 64,137,157,137,195,195,145,1
85,145
2226 DATA 72,0,120,120,0,60,60,0,0
2227 DATA 256,0,0,0,255,255,0,0,0
2240 RESTORE 2261:FOR TIME=1 TO 7:READ
 X
2250 FOR Y=0 TO 7:READ Z:POKE X+Y+STAR
T,Z:NEXT Y:NEXT TIME:RETURN
```

```
2261 DATA 24,170,170,170,170,170,170,1
70,170
2262 DATA 40,129,66,60,153,90,60,24,24

2263 DATA 48,24,60,90,153,24,60,66,129

2264 DATA 56,255,129,231,129,231,129,1
29,255
2265 DATA 64,73,0,0,146,0,0,73,0
2266 DATA 72,153,62,96,60,6,124,24,129

2267 DATA 256,255,129,129,231,129,231,
129,255
2280 RESTORE 2301:FOR TIME=1 TO 7:READ
 X
2290 FOR Y=0 TO 7:READ Z:POKE X+Y+STAR
T,Z:NEXT Y:NEXT TIME:RETURN
2301 DATA 24,255,0,255,0,255,0,255,0
2302 DATA 40,126,255,255,255,255,255,1
26,60
2303 DATA 48,60,60,60,60,60,60,255,255

2304 DATA 56,255,129,135,225,135,225,1
29,255
2305 DATA 64,198,198,0,24,24,0,198,198

2306 DATA 72,0,16,56,124,56,16,0,0
2307 DATA 256,255,129,225,135,225,135,
129,255
2320 RESTORE 2331:FOR TIME=1 TO 7:READ
 X
2330 FOR Y=0 TO 7:READ Z:POKE X+Y+STAR
T,Z:NEXT Y:NEXT TIME:RETURN
2331 DATA 24,85,85,85,85,85,85,85,85
2332 DATA 40,192,224,240,120,28,14,7,7
2333 DATA 48,7,7,7,7,15,30,62,60
2334 DATA 56,102,60,60,255,60,100,4,6
2335 DATA 64,68,146,41,68,0,36,82,137
2336 DATA 72,6,6,0,48,48,0,192,192
2337 DATA 256,102,60,60,255,60,38,32,9
6
2380 RESTORE 2391:FOR TIME=1 TO 7:READ
 X
2390 FOR Y=0 TO 7:READ Z:POKE X+Y+STAR
T,Z:NEXT Y:NEXT TIME:RETURN
2391 DATA 24,219,219,0,219,219,0,219,2
19
2392 DATA 40,63,126,252,248,240,224,96
,96
2393 DATA 48,96,96,96,96,96,96,96,96
```

```
2394 DATA 56,171,171,255,255,255,255,2
13,213
2395 DATA 64,16,40,2,37,80,4,74,160
2396 DATA 72,0,24,36,66,129,255,255,25
5
2397 DATA 256,213,213,255,255,255,255,
171,171
3000 POKE 106,PEEK(106)-5:GRAPHICS 18:
START=(PEEK(106)+1)*256
3010 POSITION 5,3:? #6;"TIME CRIME":PO
SITION 9,5:? #6;"by":POSITION 5,6:? #6
;"JACK HARDY"
3020 DIM L(9),Z(9),TX(9),TY(9),CX(9),C
Y(9):SET=0:LEVEL=1:HI=0
3030 FOR TME=9 TO 0 STEP -1:FOR TIME=1
0 TO 40:SOUND 0,TIME,10,TME:NEXT TIME
3040 FOR TIME=60 TO 90:SOUND 1,TIME,10
,TME:NEXT TIME:NEXT TME
3060 L(1)=800:L(2)=840:L(3)=850:L(4)=8
70:L(5)=890:L(6)=900:L(7)=930:L(8)=950
:L(9)=955
3070 Z(1)=2060:Z(2)=2090:Z(3)=2130:Z(4
)=2160:Z(5)=2200:Z(6)=2240:Z(7)=2280:Z
(8)=2320:Z(9)=2380
3080 FOR SHU=9 TO 2 STEP -1:NW=INT(RND
(0)*SHU)+1:T=Z(NW):Z(NW)=Z(SHU):Z(SHU)
=T:NEXT SHU
3090 ZONE=0:MX=9:MY=6:SX=MX:SY=MY:LOOT
=0:IF SET=1 THEN 3180
3110 DIM X$(38):X$="h)♥┐K┐M)♦┐N%j┴i H┐L
█♥1M┐KHPyfL fN%NIdPM♦"
3120 Z=USR(ADR(X$)):RESTORE 3150
3130 FOR TIME=1 TO 2:READ X
3140 FOR Y=0 TO 7:READ Z:POKE X+Y+STAR
T,Z:NEXT Y:NEXT TIME:SET=1
3150 DATA 8,56,56,16,124,124,56,40,40
3160 DATA 472,56,56,16,56,108,254,186,
130
3180 POSITION 3,9:? #6;"SELECT LEVEL -
█";LEVEL:POSITION 4,10:? #6;"start TO
PLAY"
3185 IF PEEK(53279)=6 THEN 3200
3190 IF PEEK(53279)=5 THEN LEVEL=LEVEL
+1:SOUND 0,30,10,4:IF LEVEL>3 THEN LEV
EL=1
3192 MAXMIN=4-LEVEL:POKE 710,INT(RND(0
)*16)*16+8:POKE 711,INT(RND(0)*16)*16+
8
3195 FOR TM=1 TO 30:NEXT TM:SOUND 0,0,
0,0:GOTO 3180
```



```
3200 GOSUB 1010:RETURN
4000 POSITION 2,1:? #6;"CONGRATULATION
S!":POSITION 2,3:? #6;"you pilliaged a
ll"
4010 POSITION 3,4:? #6;"nine time zone
s";:PUT #6,1:POSITION 1,6:? #6;"LOOT T
AKEN $";LOOT
4020 IF LOOT>HI THEN HI=LOOT
4030 POSITION 2,7:? #6;"MOST LOOT $";H
I
4040 FOR TIME=15 TO 0 STEP -1:FOR TME=
10 TO 15:SOUND 0,TME,10,TIME:SOUND 1,T
ME*5,10,TIME:NEXT TME:NEXT TIME
4050 GOTO 550
```

THE CREATOR

This is a straightforward program to create an adventure by using the disk drive to store adventure data. Only one adventure per side of a disk is permitted because the names of the files that hold the information would conflict.

The program requires Atari BASIC, 16K and disk drive.

```
10 GRAPHICS 18:POSITION 5,3:? #6;"adve
nture":POSITION 6,4:? #6;"creator"
20 POSITION 4,6:? #6;"INSERT NEWLY":PO
SITION 3,7:? #6;"FORMATTED DISK"
30 POSITION 4,9:? #6;"PRESS RETURN":OP
EN #2,4,0,"K:"
40 GET #2,K:IF K<>155 THEN 40
50 GRAPHICS 0:OPEN #3,12,0,"E:":POKE 7
10,12:POKE 712,12:POKE 709,2:DIM NA$(2
0),DIS$(14),RD$(120),DIR(6),ITEM(6)
55 RD$(1)=" ":RD$(120)=" ":RD$(2)=RD$:
NA$="":DIS$=""
60 POSITION 2,4:PRINT "DO YOU NEED INS
TRUCTIONS? (Y/N)"
70 GET #2,K:IF K<>78 AND K<>89 THEN 70

80 IF K=89 THEN GOSUB 2000
100 DIS$="D:ADVENAME":CLOSE #1
110 ? "↖":POSITION 2,4:? "WHAT IS THE
NAME OF THIS ADVENTURE   PLEASE LIMIT
 TO 20 CHARACTERS.":?
115 ? "FOR SPECIAL COLORS USE INVERSE
█ AND lower case LETTERS!!":?
120 TRAP 120:INPUT #3,NA$
130 TRAP 1100:OPEN #1,8,0,DIS$:PRINT #
1;NA$
140 NA$="":? "INPUT YOUR NAME - LIMIT
OF 20 LETTERS":?
150 TRAP 150:INPUT #3,NA$
160 PRINT #1;NA$:CLOSE #1
180 ? "↖":POSITION 2,4:? "PHASE ONE CO
MPLETED!":GOSUB 1000:DIS$="D:ADVEN.DAT
"
190 PRINT "↖":POSITION 2,4:PRINT "INPU
T NUMBER OF ROOMS ":? :TRAP 190:INPUT
#3,NR
200 IF NR<5 THEN ? "A ";NR;"-ROOM DUNG
```

```
EON IS VERY SMALL.":? "MAKE A BIGGER O
NE!":GOSUB 1000:GOTO 190
210 IF NR>50 THEN ? "YOU CAN ONLY CREA
TE A 50-ROOM DUNGEON ON ONE SIDE OF TH
E DISK.":GOSUB 1000:GOTO 190
215 OPEN #1,8,0,DIS$:PRINT #1;NR
220 ? :? "HOW MANY MOVABLE OBJECTS ARE
 IN YOUR  ADVENTURE":TRAP 220:INPUT #3
,OBS:IF OBS<=0 THEN 220
225 DIM IRN(OBS),VI(OBS),ITEM$(OBS*20)
:ITEM$(1)=" ":ITEM$(OBS*20)=" ":ITEM$(
2)=ITEM$
230 ? "↖":POSITION 2,4:? "OBJECT NAMES
 MUST BE MORE THAN 3 AND  LESS THAN 20
 LETTERS."
232 ? :? "YOU MUST ALSO REMEMBER TO MA
KE THE     FIRST FOUR LETTERS OF EACH W
ORD        DISTINCTLY DIFFERENT."
234 ? :? " EXAMPLE:  BOOKCASE and BOOK
END WOULD BY CONSIDERED THE SAME BY TH
E PROGRAM."
236 ? "YOU COULD SAY 'DUSTY BOOKCASE'
or      'HEAVY BOOKEND' THEN THE INTERP
RETER  WOULD UNDERSTAND."
237 ? :? "PRESS  SPACE BAR  TO CONTINU
E"
238 GET #2,K:IF K<>32 THEN 238
239 ? "↖":POSITION 2,4:? "◘USE UPPER C
ASE LETTERS FOR REST OF     PROGRAM.":G
OSUB 1000
240 FOR LP=1 TO OBS:? "↖":POSITION 2,4
250 TRAP 250:? "INPUT NAME OF OBJECT #
";LP:? :INPUT #3,NA$:IF LEN(NA$)<4 THE
N 250
255 ITEM$(LP*20-19,LP*20)=NA$
260 TRAP 260:? :? "INPUT ROOM NUMBER O
F ";NA$:? :INPUT #3,RM:IF RM<1 OR RM>N
R THEN 260
262 IRN(LP)=RM
265 TRAP 265:? :? "INPUT VALUE OF ";NA
$:? :INPUT #3,V
270 IF V>200000 THEN ? "SORRY, MAXIMUM
 VALUE CAN ONLY BE         $200000.":GOTO
 265
275 VI(LP)=V:NA$="":NEXT LP
280 PRINT #1;OBS:PRINT #1;ITEM$:FOR LP
=1 TO OBS:PRINT #1;IRN(LP):PRINT #1;VI
(LP):NEXT LP:CLOSE #1
290 ? "↖":POSITION 2,4:? "PHASE TWO CO
MPLETED!!":GOSUB 1000:DIS$="D:ROOM"
```

```
300 FOR LP=1 TO NR:? "↖":POSITION 2,4
310 IF LP<10 THEN DIS$(7,7)=STR$(LP):G
OTO 330
320 DIS$(7,8)=STR$(LP)
330 ? "INPUT DESCRIPTION OF ROOM #";LP
:? "(MAXIMUM OF THREE LINES)"
332 ? "FORMAT SCREEN SO WORDS ARE NOT
SPLIT  BETWEEN LINES.":?
335 TRAP 335:INPUT #3,RD$
340 ? :? "INPUT ROOM NUMBER FOR EACH D
IRECTION  NORTH,SOUTH,EAST,WEST,UP,DOW
N":?
350 ? "IF NO ROOM IN THAT DIRECTION, E
NTER 0":?
360 TRAP 1200:INPUT #3,N,S,E,W,U,D
370 OPEN #1,8,0,DIS$:PRINT #1;RD$:PRIN
T #1,N:PRINT #1,S:PRINT #1,E:PRINT #1,
W:PRINT #1,U:PRINT #1,D:CLOSE #1
380 NEXT LP
390 ? "↖":POSITION 2,4:? "PHASE THREE
COMPLETED!":GOSUB 1000
400 ? "↖":POSITION 2,4:? "ARE THERE AN
Y ROOMS THAT NEED A        SPECIAL ITEM
 TO GAIN ADMITTANCE.       (Y/N)"
410 GET #2,K:IF K<>89 AND K<>78 THEN 4
10
420 IF K=78 THEN 800
430 ? :? "PLEASE LIMIT ROOMS TO A TOTA
L OF 24.   (MAXIMUM OF 4 IN EACH DIRECT
ION)."
435 ? :? "HOW MANY ROOMS ARE NORTH?":G
OSUB 1250
450 IF K=0 THEN 500
460 OPEN #1,8,0,"D:SPNORTH.DAT":PRINT
#1;K
470 GOSUB 1300:GOSUB 1400
500 ? :? "HOW MANY ROOMS ARE SOUTH?":G
OSUB 1250
520 IF K=0 THEN 550
525 OPEN #1,8,0,"D:SPSOUTH.DAT":PRINT
#1;K
530 GOSUB 1300:GOSUB 1400
550 ? :? "HOW MANY ROOMS ARE EAST?":GO
SUB 1250
560 IF K=0 THEN 600
570 OPEN #1,8,0,"D:SPEAST.DAT":PRINT #
1;K
575 GOSUB 1300:GOSUB 1400
600 ? :? "HOW MANY ROOMS ARE WEST?":GO
SUB 1250
```

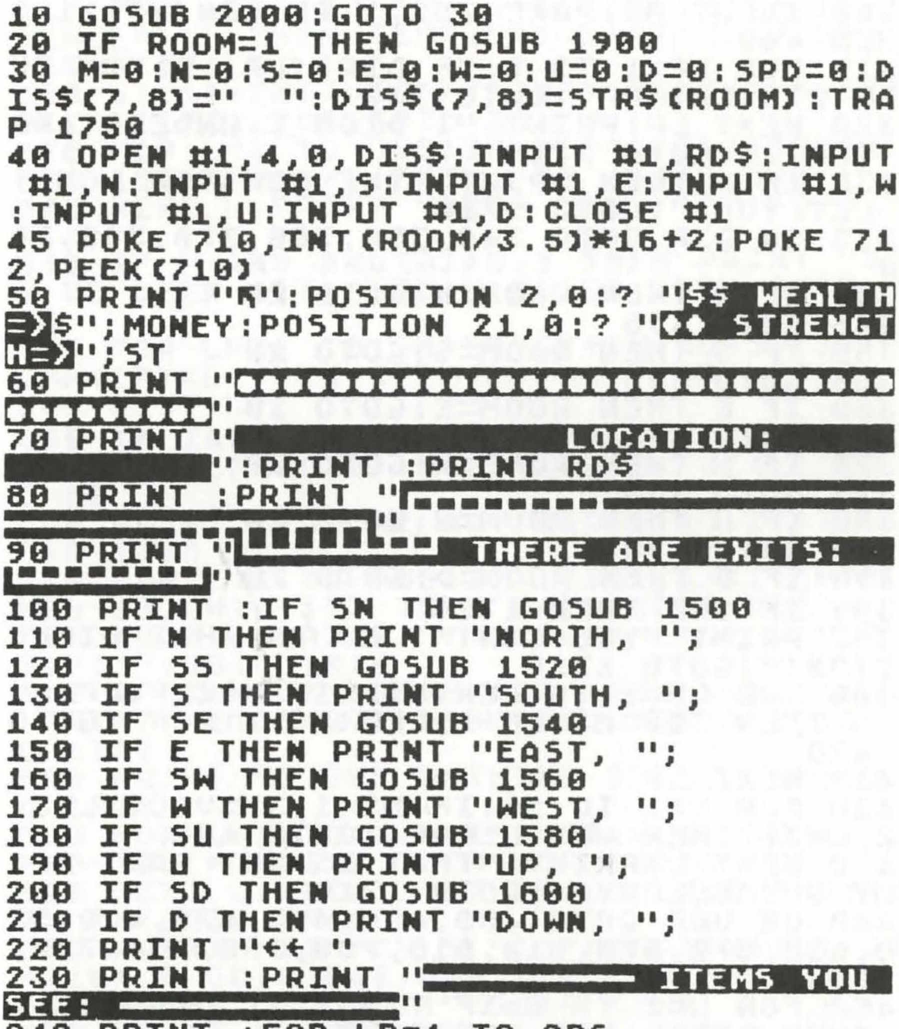
```
610 IF K=0 THEN 650
615 OPEN #1,8,0,"D:SPWEST.DAT":PRINT #
1;K
620 GOSUB 1300:GOSUB 1400
650 ? :?"HOW MANY ROOMS ARE UP?":GOSU
B 1250
660 IF K=0 THEN 700
665 OPEN #1,8,0,"D:SPUP.DAT":PRINT #1;
K
670 GOSUB 1300:GOSUB 1400
700 ?:?"HOW MANY ROOMS ARE DOWN?":GO
SUB 1250
710 IF K=0 THEN 800
720 OPEN #1,8,0,"D:SPDOWN.DAT":PRINT #
1;K
730 GOSUB 1300:GOSUB 1400
800 ?"↖":POSITION 2,4:? "PHASE FOUR C
OMPLETED!":GOSUB 1000
810 ? "↖":POSITION 2,4:? "ARE THERE AN
Y CREATURES TO FIGHT?  (Y/N)":?
820 GET #2,K:IF K<>78 AND K<>89 THEN 8
20
830 IF K=78 THEN 900
840 ? :?"HOW MANY CREATURES":TRAP 840
:INPUT #3,NC
850 IF NC>NR THEN ? "THERE ARE MORE MO
NSTERS THAN ROOMS IN YOUR DUNGEON":GOS
UB 1000:GOTO 840
860 OPEN #1,8,0,"D:CREATURE.DAT":PRINT
 #1,NC
870 DIM MON(NC):FOR LP=1 TO NC:? "↖":P
OSITION 2,4
880 ? "INPUT ROOM OF CREATURE #";LP:TR
AP 880:INPUT #3,RM
885 MON(LP)=RM:NEXT LP
890 FOR LP=1 TO NC:PRINT #1;MON(LP):NE
XT LP:CLOSE #1
895 ? "↖":POSITION 2,4:? "PHASE FIVE C
OMPLETED!":CLOSE #3:GOSUB 1000
900 ? "ARE YOU READY TO TRY YOUR DUNGE
ON?   (Y/N)"
910 GET #2,K:IF K<>89 AND K<>78 THEN 4
10
920 IF K=78 THEN END
930 TRAP 970:? "INSERT DISK WITH INTER
PRETER            PRESS  RETURN."
940 GET #2,K:IF K<>155 THEN 940
950 CLOSE #2:RUN "D:INTERPRE.TER"
960 END
970 TRAP 40000:OPEN #2,4,0,"K:":GOTO 9
```

```
30
1000 FOR WAIT=1 TO 400:NEXT WAIT:RETUR
N
1100 TRAP 40000:CLOSE #1:? :? "◘ UNLOC
K DISK AND  PRESS RETURN .
1110 GET #2,K:IF K<>155 THEN 1110
1120 GOTO 130
1200 TRAP 40000:PRINT "◘YOU MUST ENTER
 A NUMBER FOR EACH      DIRECTION.":GO
SUB 1000:GOTO 340
1250 ? "ENTER  0  IF NONE!"
1260 GET #2,K:K=K-48:IF K>4 THEN ? "PL
EASE LIMIT TO 4 OR LESS":GOTO 1260
1270 RETURN
1300 ? "↖":POSITION 2,4:? "REMEMBER, Y
OU MUST ENTER THE NUMBER OFTHE ROOM WH
ICH LETS YOU GAIN ACCESS"
1310 ? "TO THE SECRET ROOM.":GOSUB 100
0:GOSUB 1000:RETURN
1400 FOR LP=1 TO K:? "↖":POSITION 2,4
1410 ? "INPUT #";LP:? :? " - FROM ROOM
 NUMBER ":TRAP 1410:? :INPUT #3,RM:DIR
(LP)=RM
1420 ? :? "INPUT ITEM (NUMBER ONLY) NE
EDED TO    GAIN ACCESS FROM ROOM #";RM
:TRAP 1420:? :INPUT #3,IT:ITEM(LP)=IT
1430 NEXT LP:FOR LP=1 TO K:PRINT #1;DI
R(LP):PRINT #1;ITEM(LP):NEXT LP:CLOSE
#1:RETURN
2000 ? "↖":? "BEFORE YOU BEGIN, YOU MU
ST DESIGN    YOUR DUNGEON ON PAPER."
2005 ? :? "FIRST, LAY OUT YOUR ADVENTU
RE MAP      AND NUMBER YOUR ROOMS."
2010 ? :? "SECOND, LIST, NUMBER, AND V
ALUE THE    CONTENTS OF EACH ROOM."
2020 ? :? "THIRD, CONSTRUCT A LIST OF
ITEMS       BY NUMBER NEEDED TO ACCESS
SPECIAL     ROOMS."
2025 ? :? "FOURTH, PREPARE A LIST OF A
LL ROOMS    WHICH CONTAIN CREATURES."
2030 ? :? "FINALLY, ENTER ALL THE DATA
."
2040 ? :? "THEN, ANY TIME YOU WANT TO
PLAY YOUR   ADVENTURE, LOAD IN INTERPRE
TER         AND RUN."
2050 POKE 752,1:POSITION 4,23:? "PRESS
  SPACE BAR  TO CONTINUE";
2060 GET #2,K:IF K<>32 THEN 2060
2070 POKE 752,0:RETURN
```

INTERPRETER

This program accompanies the preceding one and is used to play the adventure which was created with that program.

The program requires Atari Basic, 16K and disk drive.



```
10 GOSUB 2000:GOTO 30
20 IF ROOM=1 THEN GOSUB 1900
30 M=0:N=0:S=0:E=0:W=0:U=0:D=0:SPD=0:D
IS$(7,8)="  ":DIS$(7,8)=STR$(ROOM):TRA
P 1750
40 OPEN #1,4,0,DIS$:INPUT #1,RD$:INPUT
 #1,N:INPUT #1,S:INPUT #1,E:INPUT #1,W
:INPUT #1,U:INPUT #1,D:CLOSE #1
45 POKE 710,INT(ROOM/3.5)*16+2:POKE 71
2,PEEK(710)
50 PRINT "↖":POSITION 2,0:? "$$ WEALTH
=>$";MONEY:POSITION 21,0:? "◆◆ STRENGT
H=>";ST
60 PRINT "○○○○○○○○○○○○○○○○○○○○○○○○○○
○○○○○○○○○○"
70 PRINT "████████████ LOCATION:
██████████":PRINT :PRINT RD$
80 PRINT :PRINT "┌▬▬▬▬▬▬▬▬▬▬━━━━━━━━━━━
━━━━━━━━━━━━━━━━━━━━━━━━"
━━━━━━━━━━▬▬┐██████└▬▬▬ THERE ARE EXITS:
90 PRINT "└▬▬▬▬▬▬▬▬▬┘"
100 PRINT :IF SN THEN GOSUB 1500
110 IF N THEN PRINT "NORTH, ";
120 IF SS THEN GOSUB 1520
130 IF S THEN PRINT "SOUTH, ";
140 IF SE THEN GOSUB 1540
150 IF E THEN PRINT "EAST, ";
160 IF SW THEN GOSUB 1560
170 IF W THEN PRINT "WEST, ";
180 IF SU THEN GOSUB 1580
190 IF U THEN PRINT "UP, ";
200 IF SD THEN GOSUB 1600
210 IF D THEN PRINT "DOWN, ";
220 PRINT "←←"
230 PRINT :PRINT "━━━━━━━━━ ITEMS YOU
SEE: ━━━━━━━━━━━"
240 PRINT :FOR LP=1 TO OBS
250 IF IRN(LP)=ROOM THEN I$=ITEM$(LP*2
0-19,LP*20):GOSUB 1650:PRINT I$;", ";:
I$=""
```

```
260 NEXT LP:PRINT "←← "
270 FOR L=1 TO CRE:IF MON(L)=ROOM THEN
 PRINT "THERE IS A MONSTER HERE!":M=L
275 NEXT L:PRINT
280 PRINT "       INSTRUCTIONS:
        ":PRINT
290 POKE 764,255:POKE 752,0:TRAP 1700:
M$="":TI$=""
300 INPUT M$:POKE 752,1:IF LEN(M$)>1 T
HEN 400
310 FOR LP=1 TO 6:IF DIR$(LP,LP)=M$ TH
EN DIR=LP:POP :GOTO 330
320 NEXT LP:PRINT "I DIDN'T UNDERSTAND
 THAT!":GOTO 1710
330 IF M THEN PRINT "THE MONSTER WON'T
 LET YOU!":GOTO 1710
335 ON DIR GOTO 340,350,360,370,380,39
0
340 IF N THEN ROOM=N:GOTO 20
345 GOTO 395
350 IF S THEN ROOM=S:GOTO 20
355 GOTO 395
360 IF E THEN ROOM=E:GOTO 20
365 GOTO 395
370 IF W THEN ROOM=W:GOTO 20
375 GOTO 395
380 IF U THEN ROOM=U:GOTO 20
385 GOTO 395
390 IF D THEN ROOM=D;GOTO 20
395 IF SPD THEN 1720
397 PRINT "YOU CANT'T GO IN THAT DIREC
TION!":GOTO 1710
400 FOR LP=1 TO LEN(M$):IF M$(LP,LP)="
" THEN TI$=M$(LP+1,LEN(M$)):POP :GOTO
 420
410 NEXT LP
420 FOR L=1 TO 16:IF M$(1,3)=VOC$(L*3-
2,L*3) THEN VOC=L:POP :GOTO 440
430 NEXT L:PRINT "THAT IS NOT PART OF
MY VOCABULARY!":GOTO 1710
440 ON VOC GOTO 450,450,450,470,470,53
0,530,570,570,610,610,700,700,750,750,
1000
450 FOR L=1 TO 6:IF TI$(1,1)=DIR$(L,L)
 THEN DIR=L:POP :GOTO 330
460 NEXT L:PRINT "YOU CAN'T GO THERE!"
:GOTO 1710
470 FOR L=1 TO OBS:IF TI$(1,4)=ITEM$(L
*20-19,L*20-16) THEN POP :GOTO 490
480 NEXT L:PRINT "YOU CAN'T TAKE A ";T
```

```
I$:GOTO 1710
490 IF IRN(L)=0 THEN PRINT "YOU ALREAD
Y HAVE IT!":GOTO 1710
500 IF IRN(L)<>ROOM THEN PRINT "THAT I
TEM IS NOT HERE!":GOTO 1710
510 IF ST-INT(VI(L)/1000)<0 THEN PRINT
 "YOU ARE TOO WEAK TO CARRY THAT ITEM!
":GOTO 1710
512 FOR LP=1 TO SN+SS+SE+SW+SU+SD:IF I
TEM(LP)=L THEN TEMP(LP)=L:ITEM(LP)=0
515 NEXT LP
520 ST=ST-INT(VI(L)/1000):MONEY=MONEY+
VI(L):IRN(L)=0:GOTO 30
530 FOR L=1 TO OBS:IF TI$(1,3)=ITEM$(L
*20-19,L*20-17) THEN POP :GOTO 545
540 NEXT L:PRINT "DROP WHAT!!!":GOTO 1
710
545 IF M AND RND(0)>0.5 THEN PRINT "YO
U'VE BEEN ATTACKED!":GOSUB 2500:GOTO 7
70
547 FOR LP=1 TO SN+SS+SE+SW+SU+SD:IF T
EMP(LP)=L THEN ITEM(LP)=L:TEMP(LP)=0
548 NEXT LP
550 IF IRN(L)=0 THEN IRN(L)=ROOM:ST=ST
+INT(VI(L)/1000):MONEY=MONEY-VI(L):GOT
O 50
560 PRINT "YOU'RE NOT CARRYING IT!!":G
OTO 1710
570 FOR L=1 TO OBS:IF TI$(1,3)=ITEM$(L
*20-19,L*20-17) THEN POP :GOTO 590
580 NEXT L:PRINT "IT LOOKS LIKE A ";TI
$;"!!":GOTO 1710
590 IF IRN(L)<>ROOM AND IRN(L)<>0 THEN
 PRINT "IT'S NOT HERE TO LOOK AT!":GOT
O 1710
600 PRINT "I SEE NOTHING SPECIAL!":GOT
O 1710
610 FOR L=1 TO OBS:IF TI$(1,3)=ITEM$(L
*20-19,L*20-17) THEN POP :GOTO 630
620 NEXT L:PRINT "IT FEELS LIKE A ";TI
$;"!":GOTO 1710
630 PRINT "IT FEELS ";:F=INT(RND(0)*6)
+1:ON F GOTO 640,650,660,670,680,690
640 PRINT "ROUGH!!":GOTO 1710
650 PRINT "SMOOTH!!":GOTO 1710
660 PRINT "WET!!":GOTO 1710
670 PRINT "DRY!!":GOTO 1710
680 PRINT "COLD!":GOTO 1710
690 PRINT "WARM!":GOTO 1710
700 PRINT "IF YOU HAVE THE CORRECT OBJ
```

```
ECT,       THE ITEM OPENS AUTOMATICALL
Y!↑":GOTO 1710
750 IF M THEN 770
760 PRINT "THERE IS NOTHING HERE TO FI
GHT!":GOTO 1710
770 PRINT "↰":POSITION 2,4:PRINT "THE
MONSTER IS A ";:POKE 710,66:POKE 712,6
6:POKE 752,1:F=INT(RND(0)*4)+1
775 ON F GOTO 780,790,800,810
780 PRINT "HOBGOBLIN!!":MST=INT(RND(0)
*80)+5:GOTO 850
790 PRINT "DRAGON!!":MST=INT(RND(0)*10
0)+30:GOTO 850
800 PRINT "ZOMBIE!!":MST=INT(RND(0)*40
)+10:GOTO 850
810 PRINT "GIANT SPIDER!":MST=INT(RND(
0)*80)+20:GOTO 850
850 POSITION 1,8:PRINT "MONSTER STRENG
TH ";MST;"  ":POSITION 1,10:PRINT "YOU
R STRENGTH ";ST;"  ";:POKE 764,255
860 POSITION 12,19:PRINT " PUSH SPACE
BAR ":ML=INT(RND(0)*10):POSITION 5,14:
PRINT "MONSTER LOSES ";ML
870 YL=INT(RND(0)*6):POSITION 24,14:PR
INT "YOU LOSE ";YL
880 IF PEEK(764)<>255 THEN 900
890 POSITION 12,19:PRINT " PUSH SPACE
BAR ":FOR T=1 TO 10:NEXT T:GOTO 860
900 COL=PEEK(710):POKE 710,0:POKE 712,
0:MST=MST-ML:ST=ST-YL:SOUND 0,100,2,10
:FOR T=1 TO 50:NEXT T:SOUND 0,0,0,0
910 IF MST<=0 THEN PRINT "   YOU KILL
ED THE CREATURE!!";:MON(M)=0:M=0:GOSUB
 2500:POKE 752,0:GOTO 45
920 IF ST<=0 THEN 1800
930 POKE 710,COL:POKE 712,COL:GOTO 850

1000 PRINT "↰":PRINT "  YOU ARE CARRYIN
G: - - VALUED AT:       "
1010 PRINT :FOR L=1 TO OBS:IF IRN(L)=0
 THEN PRINT ITEM$(L*20-19,L*20);:PRINT
 "▶    $";VI(L)
1020 NEXT L:POSITION 5,22:PRINT "TOUCH
  SPACE BAR  TO CONTINUE":OPEN #2,4,0,
"K:"
1030 GET #2,K:IF K<>32 THEN 1030
1040 CLOSE #2:GOTO 50
1500 FOR LP=1 TO SN:IF SR(LP)=ROOM AND
 ITEM(LP) THEN N=0:PRINT " NORTH , ";:
SPD=1
```

```
1510 NEXT LP:RETURN
1520 FOR LP=SN+1 TO SN+SS:IF SR(LP)=RO
OM AND ITEM(LP) THEN S=0:PRINT "  SOUTH
 , ";:SPD=1
1530 NEXT LP:RETURN
1540 FOR LP=SN+SS+1 TO SN+SS+SE:IF SR(
LP)=ROOM AND ITEM(LP) THEN E=0:PRINT "
 EAST , ";:SPD=1
1550 NEXT LP:RETURN
1560 FOR LP=SN+SS+SE+1 TO SN+SS+SE+SW:
IF SR(LP)=ROOM AND ITEM(LP) THEN W=0:P
RINT " WEST , ";:SPD=1
1570 NEXT LP:RETURN
1580 FOR LP=SN+SS+SE+SW+1 TO SN+SS+SE+
SW+SU:IF SR(LP)=ROOM AND ITEM(LP) THEN
 U=0:PRINT " UP , ";:SPD=1
1590 NEXT LP:RETURN
1600 FOR LP=SN+SS+SE+SW+SU+1 TO SN+SS+
SE+SW+SU+SD:IF SR(LP)=ROOM AND ITEM(LP
) THEN D=0:PRINT " DOWN , ";:SPD=1
1610 NEXT LP:RETURN
1650 IF I$(LEN(I$),LEN(I$))=" " THEN I
$=I$(1,LEN(I$)-1):GOTO 1650
1660 RETURN
1700 PRINT "↖PLEASE SAY THAT AGAIN!":T
RAP 40000
1710 FOR LP=1 TO 250:NEXT LP:PRINT "↑↑
↑↓↓↓↓↓↓↓↓↓":GOTO 290
1720 PRINT "↖ EXITS LISTED IN INVERSE
REQUIRE          SPECIAL OBJECTS TO USE!!
↑":GOTO 1710
1750 ? "↖":POSITION 2,4:? "↖CHECK DISK
 DIRECTORY FOR ROOM #";ROOM:STOP
1800 GRAPHICS 18:POSITION 5,1:PRINT #6
;"the monster":POSITION 5,2:PRINT #6;"
killed you"
1810 LN=INT((20-LEN(NA$))/2):POSITION
LN,4:PRINT #6;NA$:GOSUB 2400
1815 POSITION 1,6:PRINT #6;"TREASURES
$";MONEY
1820 POSITION 2,8:PRINT #6;"TRY AGAIN?
 (Y/N)":OPEN #2,4,0,"K:"
1830 GET #2,K:IF K<>78 AND K<>89 THEN
1830
1840 CLOSE #2:IF K=89 THEN RUN
1850 POSITION 1,9:PRINT #6;"THANKS FOR
 PLAYING!":FOR L=1 TO 200:NEXT L:END
1900 GRAPHICS 18:POSITION 1,1:PRINT #6
;"YOU ARE IN ROOM #1"
1910 POSITION 1,3:PRINT #6;"YOU HAVE $
```

```
";MONEY:POSITION 4,4:PRINT #6;"IN TREA
SURE"
1920 LN=((20-LEN(NA$))/2):POSITION LN,
6:PRINT #6;NA$:GOSUB 2500
1930 POSITION 5,8:PRINT #6;"do you wan
t":POSITION 1,9:PRINT #6;"to continue
(Y/N)":OPEN #2,4,0,"K:"
1940 GET #2,K:IF K<>78 AND K<>89 THEN
1940
1950 CLOSE #2:IF K=78 THEN POSITION 0,
7:PRINT #6;"LIVE LONG & PROSPER!":FOR
T=1 TO 250:NEXT T:END
1960 GRAPHICS 0:POKE 710,0:RETURN
2000 GRAPHICS 18:POSITION 7,3:? #6;"in
sert":POSITION 6,4:? #6;"ADVENTURE":PO
SITION 8,5:? #6;"DISK"
2010 POSITION 5,8:? #6;"PRESS RETURN":
OPEN #2,4,0,"K:"
2020 GET #2,K:IF K<>155 THEN 2020
2030 CLOSE #2:DIM TI$(20),NA$(20),DIS$
(14),DIR$(6),V$(3),VOC$(48),M$(30),I$(
20),RD$(120),SR(24),ITEM(24),TEMP(24)
2035 RD$(1)=" ":RD$(120)=" ":RD$(2)=RD
$:VOC$=RD$(1,48)
2040 NA$=RD$(1,20):TI$=NA$:DIS$="D:ROO
M":SN=0:SS=0:SE=0:SW=0:SU=0:SD=0:ROOM=
1:MONEY=0:ST=0
2045 FOR L=1 TO 24:SR(L)=0:ITEM(L)=0:T
EMP(L)=0:NEXT L
2050 TRAP 2700:OPEN #1,4,0,"D:ADVENAME
":INPUT #1,TI$:INPUT #1,NA$:CLOSE #1:L
N=INT((20-LEN(TI$))/2)
2060 TRAP 40000:? #6;"↰":POSITION LN,4
:? #6;TI$:LN=INT((20-LEN(NA$))/2)
2065 POSITION 9,7:PRINT #6;"BY":POSITI
ON LN,8:PRINT #6;NA$
2070 FOR V=0 TO 15 STEP 0.5:SOUND 0,60
,12,V:SOUND 1,120,12,V:SOUND 2,60,12,V
:SOUND 3,120,12,V:NEXT V
2080 FOR V=15 TO 0 STEP -0.5:SOUND 0,6
0,12,V:SOUND 1,120,12,V:SOUND 2,60,12,
V:SOUND 3,120,12,V:NEXT V
2090 OPEN #1,4,0,"D:ADVEN.DAT":INPUT #
1,NR:INPUT #1,OBS:DIM IRN(OBS),VI(OBS)
,ITEM$(20*OBS)
2100 INPUT #1,ITEM$:FOR LP=1 TO OBS:IN
PUT #1,RM:INPUT #1,VA:IRN(LP)=RM:VI(LP
)=VA:NEXT LP:CLOSE #1
2110 TRAP 2130:OPEN #1,4,0,"D:SPNORTH.
DAT":INPUT #1,SN
```

```
2120 FOR LP=1 TO SN:INPUT #1,RM:INPUT
#1,I:SR(LP)=RM:ITEM(LP)=I:NEXT LP
2130 CLOSE #1:TRAP 40000:TRAP 2150:OPE
N #1,4,0,"D:SPSOUTH.DAT":INPUT #1,SS
2140 FOR LP=SN+1 TO SN+SS:INPUT #1,RM:
INPUT #1,I:SR(LP)=RM:ITEM(LP)=I:NEXT L
P
2150 CLOSE #1:TRAP 40000:TRAP 2170:OPE
N #1,4,0,"D:SPEAST.DAT":INPUT #1,SE
2160 FOR LP=SN+SS+1 TO SN+SS+SE:INPUT
#1,RM:INPUT #1,I:SR(LP)=RM:ITEM(LP)=I:
NEXT LP
2170 CLOSE #1:TRAP 40000:TRAP 2190:OPE
N #1,4,0,"D:SPWEST.DAT":INPUT #1,SW
2180 FOR LP=SN+SS+SE+1 TO SN+SS+SE+SW:
INPUT #1,RM:INPUT #1,I:SR(LP)=RM:ITEM(
LP)=I:NEXT LP
2190 CLOSE #1:TRAP 40000:TRAP 2210:OPE
N #1,4,0,"D:SPUP.DAT":INPUT #1,SU
2200 FOR LP=SN+SS+SE+SW+1 TO SN+SS+SE+
SW+SU:INPUT #1,RM:INPUT #1,I:SR(LP)=RM
:ITEM(LP)=I:NEXT LP
2210 CLOSE #1:TRAP 40000:TRAP 2230:OPE
N #1,4,0,"D:SPDOWN.DAT":INPUT #1,SD
2220 FOR LP=SN+SS+SE+SW+SU+1 TO SN+SS+
SE+SW+SU+SD:INPUT #1,RM:INPUT #1,I:SR(
LP)=RM:ITEM(LP)=I:NEXT LP
2230 CLOSE #1:TRAP 40000:TRAP 2250:OPE
N #1,4,0,"D:CREATURE.DAT":INPUT #1,CRE

2240 DIM MON(CRE):FOR LP=1 TO CRE:INPU
T #1,RM:MON(LP)=RM:NEXT LP
2250 CLOSE #1:TRAP 40000
2260 FOR LP=1 TO 16:READ V$:VOC$(LP*3-
2,LP*3)=V$:NEXT LP
2270 FOR LP=1 TO 6:READ V$:DIR$(LP,LP)
=V$:NEXT LP
2280 FOR LP=1 TO OBS:ST=ST+INT(VI(LP)/
1000):NEXT LP
2290 GRAPHICS 0:POKE 710,0:POSITION 14
,8:? "    WELCOME!!     ":? "ENTER YOUR N
AME INTO THE BOOK OF THE  DEAD."
2300 TRAP 2300:INPUT NA$
2310 POKE 752,1:PRINT :PRINT "MAY THE
BLESSINGS OF THE GODS BE      WITH YOU
 ";NA$;"!"
2320 GOSUB 2400:RETURN
2400 RESTORE 3100:FOR L=1 TO 9:READ SO
,DELAY:SOUND 0,SO,10,8:FOR T=1 TO DELA
Y:NEXT T:SOUND 0,0,0,0
```

```
2410 FOR T=1 TO 8:NEXT T:NEXT L:RETURN

2500 RESTORE 3200:FOR L=1 TO 6:READ SO
,DELAY:SOUND 0,SO,10,10:FOR T=1 TO DEL
AY:NEXT T:NEXT L:SOUND 0,0,0,0:RETURN

2700 ? "↖":POKE 710,66:POKE 712,66:FOR
 T=1 TO 30:NEXT T:RUN
3000 DATA WAL,RUN,GO ,GET,TAK,DRO,GIV,
LOO,EXA,TOU,FEE,OPE,UNL,FIG,KIL,INV,N,
S,E,W,U,D
3100 DATA 100,60,100,40,100,10,100,60,
85,80,90,60,100,40,105,20,100,60
3200 DATA 96,10,72,10,57,10,48,20,57,1
0,48,40
```

APPENDIX B

Useful Information

The following pages contain additional information and instructions which you may find helpful in developing your own computer adventure. Included are:

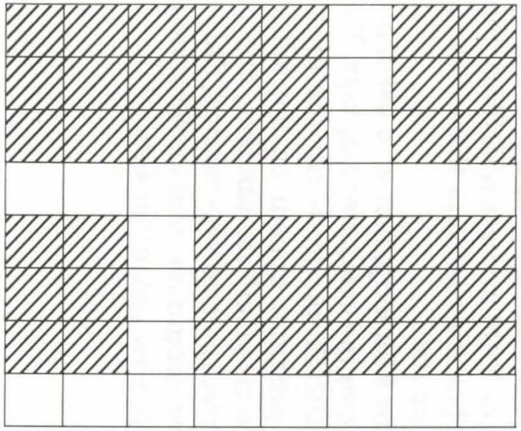
Character Set Manipulation
Functions of PEEK and POKE
Atari Hue Numbers and Colors
Keyboard and Internal Codes
Character Set Values

CHARACTER SET MANIPULATION

In the Operating System, ROM, of the Atari computer are stored several character sets—uppercase alphabet, lowercase alphabet, numbers, special punctuation characters, and a special graphics character set. There are instances when it becomes necessary to be able to define one's own character set. Processing words in foreign languages, making maps or playing fields for games, and character-set animation are some of the more frequent uses for this capability which is readily supported by Atari Computers and its associated languages, Atari BASIC and Atari Microsoft BASIC. The OS data base contains a pointer (CHBAS) at memory location 2F4 hex (756 decimal) which points to the character set to be used. Usually it points to the OS ROM standard character set. However, in BASIC, one can POKE one's own character set into a free area of RAM and reset CHBAS, the OS pointer, to point to the new character set and these new characters will be used by the computer.

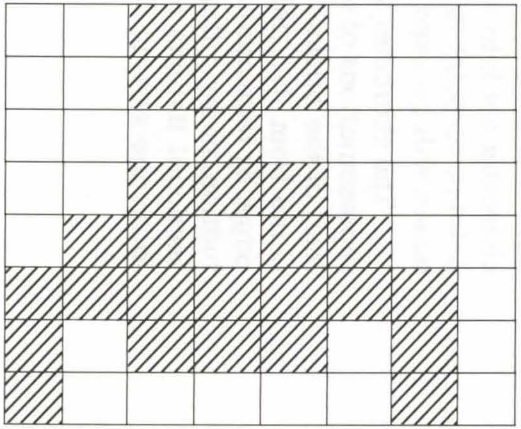
The character set is an 8 by 8 bit map; that is, a set of binary representations of each character the computer displays. The bit map of the bricks from the HUNTER program and the time machine from the TIME CRIME program appear on the following page. By redesigning the bit maps of letters, numbers, and punctuation, many different characters sets can be created. Study the sectional listings of the above programs to discover how each language sets up and displays new character sets.

BRICKS



BINARY								DECIMAL
1	1	1	1	1	0	1	1	251
1	1	1	1	1	0	1	1	251
1	1	1	1	1	0	1	1	251
0	0	0	0	0	0	0	0	0
1	1	0	1	1	1	1	1	223
1	1	0	1	1	1	1	1	223
1	1	0	1	1	1	1	1	223
0	0	0	0	0	0	0	0	0

TIME MACHINE



0	0	1	1	1	0	0	0	56
0	0	1	1	1	0	0	0	56
0	0	0	1	0	0	0	0	16
0	0	1	1	1	0	0	0	56
0	1	1	0	1	1	0	0	108
1	1	1	1	1	1	1	0	255
1	0	1	1	1	0	1	0	186
1	0	0	0	0	0	1	0	130

FUNCTIONS OF PEEK AND POKE

Every memory location in your computer, whether it is RAM or ROM, has its own address. The first address is 0 and the last is 65535. At the moment the computer is turned on the operating system, or OS, takes over and stores values in many memory locations. These are called the default values. It turns the color of the outer screen black (location 712) and the inner screen blue (location 710). It sets the left screen margin (location 82) to two, and so on.

To be able to find what number the computer has stored in a particular location it is necessary to ask the computer. This can be done by PEEKing a location in memory. The format could be:

```
PRINT PEEK(710)
```

which will return the value stored in the color register for the inner screen, or it could be:

```
A=PEEK(82)
```

which will now store the value of the particular location into the variable A. Location 82 contains the value of the left margin.

For PILOT the format would be:

```
T:@B710 and C:#A=@B82
```

The number stored in one memory location cannot exceed 255; therefore, the computer stores the larger number in two consecutive memory locations. The first location is called the low-order address and the second is called the high-order address. In BASIC, multiplying the number stored in the second location by 256 and adding the contents of the first address calculates the number stored. In PILOT, use the pointer variable @ without the B (Byte) to request two consecutive memory locations (Word).

Example: Locations 741 and 742 contain the number that corresponds to the top of your available memory or MEMTOP. This is always more than 255 and can be found by using PRINT PEEK(741) + PEEK(742)*256 in BASIC, or T:@741 in PILOT. (*Note:* This will not work in PILOT if your system has more than 32K, because PILOT will only return integers between −32768 and +32767.)

To alter what is stored in memory locations we use the POKE statement (or the C: statement in PILOT). The format would be POKE 710,0, which would turn your inner screen black. C:@B710 = 0 would accomplish the same thing in PILOT.

Some rules:

1. Do not POKE numbers into unknown locations because the wrong value could cause your system to lock up or crash. Nothing you POKE in can really hurt the computer, for recovery is achieved by turning the computer off and then on again. (Of course, any program in memory is lost.)
2. Always use the decimal number when POKing a value into a memory location.

USEFUL LOCATIONS TO PEEK AND POKE

DISPLAY LIST

512,513 Display List Interrupt Vector—VDSLST. These locations store the address of the instructions that will be executed in the event of a display list interrupt.

559 Direct Memory Access (DMA) Control Register—SDMCTL. This is used to turn off the screen display chip.

560,561 Display List Address—SDLST. These hold the address of the start of the display list.

DISPLAY SCREEN

77 Attract Mode—ATRACT. Less than 128 is the normal value stored here. More than 128 rotates the colors to protect the T.V. screen. This happens automatically after 9 minutes with no key being pressed. POKE this location to 0 when using games requiring joysticks or paddles to maintain color values.

82 Left Margin of Screen in mode 0—LMARGN. Default value = 2.

83 Right Margin of Screen in mode 0—RMARGN. Default value = 39.

85,86 Current Cursor Column Position—COLCRS.

87 Contains Current Display Mode—DINDEX.

90 Previous Graphics Cursor Row—OLDROW.

91,92 Previous Graphics Cursor Column—OLDCOL.

93 Cursor Character Save/Restore—OLDCHR. Character under graphics window cursor. Used to restore hidden character when the cursor moves.

94,95 Cursor Memory Address—OLDADR.

96 Ending Graphics Cursor Row—NEWROW. Point to which DRAWTO will go.

97,98 Ending Graphics Cursor Column—NEWCOL. Point to which DRAWTO will go.

201 Display Screen Tab Width—PTABW. Default = 10.

656 Text Cursor Row Position—TXTROW.

657,658 Text Cursor Column Position—TXTCOL.

660,661 Split-Screen Memory Address—TXTMSC. Contains address of upper left corner of text window.

675,689 Display Screen Tab Stop Map—TABMAP.

The following locations (708 through 712) apply to text colors when using Graphics Modes 1 or 2.

708 Color of Capital Letters—COLOR0.

709 Color of Lowercase Letters—COLOR1.

710 Color of Inverse Capital Letters (Text Screen in Mode 0 and Text Window in Split-Screen Modes 1 or 2)—COLOR2.

711 Color of Inverse Lowercase Letters—COLOR3.

712 Color of Background (Border-Mode 0)—COLOR4.

752 Cursor Inhibit—CRSINH. Default = 0. To turn off, POKE a 1 into this location.

755 Character and Cursor Mode Register—CHACT. POKE a 4, 5, 6 or 7 here and characters will appear upside down.

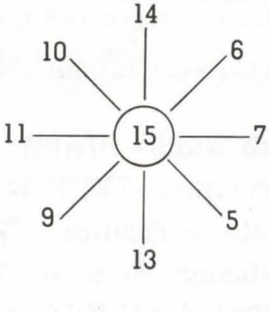
756 Character Base Register—CHBAS. This variable determines which character set will be used in screen modes 1 and 2. Default = 224. POKE with 226 for lower-case letters and graphics character set.

763 Last ATASCII Character Read or Written or Value of the Graphics Point—ATACHR.

GAME CONTROL PORTS

624-631 Paddle Value—PADDL0-PADDL7. Returns value of Paddle 0 through 7 (if plugged in). Values from 0-228.

632-635 Joystick Value—STICK0-STICK3. Returns value of Stick 0 through 3.



Joystick Values

636-643 Paddle Trigger Value—PTRIG0-PTRIG7. Returns value of paddle triggers 0 through 7. 0 = Pressed, 1 = Not Pressed.

644-647 Joystick Trigger Value—STRIG0-STRIG3. Returns value of joystick trigger 0 through 3. 0 = Pressed, 1 = Not Pressed.

Note: 632-635 also used to read lightpen switch when plugged into ports 1 through 4.

KEYBOARD

16 Pokey Interrupt Vector—POKMSK. Used along with location 53774 to disable the break key.

Example: Use the following as a subroutine to disable the break key (must be disabled after each graphics statement):

17 Break Key Flag—BRKKEY. 0 = Break key pressed.

702 Shift/Control Lock Flag—SHFLOK. Used to control input, etc. POKE a 64 into this location and keyboard will type only capital letters—0 = lowercase letters, 128 = control characters, 255 = no letters accepted.

764 Keyboard Character—CH. This location retains the value of the most recently pressed key or the value of 255, which means no key pressed.

53279 Console Switches—CONSOL. This location can be used to interpret which of the three console keys have been pressed.

0 = Option, Select, and Start
1 = Option and Select
2 = Option and Start
3 = Option
4 = Select and Start
5 = Select
6 = Start
7 = None

MEMORY CONFIGURATION

14,15 Display Screen Lower Limit—APPMHI. These locations contain the highest location available for program lines, data, and variables.

88,89 Screen Memory Address—SAVMSC. These locations contain the lowest address of screen memory. The value at the address is displayed at the upper left corner of the screen.

106 Top of RAM address (Most Significant Byte)—RAMTOP. If you multiply the contents of this memory location by 256, you will find how much RAM your system has.

741,742 Free Memory High Address—MEMTOP.

743,744 Free Memory Low Address—MEMLO.

PLAYER-MISSILE GRAPHICS

623 Priority Register—GPRIOR. Used to select which objects on the screen will appear to be in front of the others.

1 = All players have priority over all playfields

2 = Players 0 and 1 have priority over all playfields and all playfields have priority over players 2 and 3

4 = All playfields have priority over all players

8 = Playfields 0 and 1 have priority over all players and all players have priority over playfields 2 and 3

704 Color of Player and Missile 0—COLPM0.

705 Color of Player and Missile 1—COLPM1.

706 Color of Player and Missile 2—COLPM2.

707 Color of Player and Missile 3—COLPM3.

When using registers below you POKE in a number to set Horizontal Positions and PEEK the location to see if there has been a collision.

53248 Player 0 Horizontal Position Register—HPOSP0 (Poke).
Collision of Missile 0 to Playfield—M0PF (Peek).

53249 Player 1 Horizontal Position Register—HPOSP1 (Poke).
Collision of Missile 1 to Playfield—M1PF (Peek).

53250 Player 2 Horizontal Position Register—HPOSP2 (Poke).
Collision of Missile 2 to Playfield—M2PF (Peek).

53251 Player 3 Horizontal Position Register—HPOSP3 (Poke).
Collision of Missile 3 to Playfield—M3PF (Peek).

53252 Missile 0 Horizontal Position Register—HPOSM0 (Poke).
Collision of Player 0 to Playfield—P0PF (Peek).

53253 Missile 1 Horizontal Position Register—HPOSM1 (Poke).
Collision of Player 1 to Playfield—P1PF (Peek).

53254 Missile 2 Horizontal Position Register—HPOSM2 (Poke).
Collision of Player 2 to Playfield—P2PF (Peek).

53255 Missile 3 Horizontal Position Register—HPOSM3 (Poke).
Collision of Player 3 to Playfield—P3PF (Peek).

When the Width Registers below are POKED with a 0 or a 2, the player or missile is displayed at normal width, a 1 displays twice normal width and a 3 displays quadruple width.

53256 Player 0 Width Register—SIZEP0 (Poke).
Collision of Missile 0 to Player—M0PL (Peek).

53257 Player 1 Width Register—SIZEP1 (Poke).
Collision of Missile 1 to Player—M1PL (Peek).

53258 Player 2 Width Register—SIZEP2 (Poke).
Collision of Missile 2 to Player—M2PL (Peek).

53259 Player 3 Width Register—SIZEP3 (Poke).
Collision of Missile 3 to Player—M3PL (Peek).

53260 Missile Width Register—SIZEM (Poke).
This register controls the width of all 4 missiles.

Note: You can make missiles of different sizes by POKing with numbers from 0 to 255. *Example:* 0,1,3 control Missile 0; 0,4,12 control Missile 1; 0,16,48 control Missile 2; 0,64,192 control Missile 3. To have Missile 0 quadruple, Missile 1 double, Missile 2 normal, and Missile 3 quadruple width you would POKE 53260,3+4+0+192 (or 199).

Collision of Player 0 to Player—P0PL (Peek).

53261 Collision of Player 1 to Player—P1PL (Peek).

53262 Collision of Player 2 to Player—P2PL (Peek).

53263 Collision of Player 3 to Player—P3PL (Peek).

52277 Graphics Control Register—GRACTL.
This location works along with location 559 to control DMA for player-missile graphics. A value of 2 enables player DMA only, a value of 1 enables missile DMA only, and a value of 3 enables both.

54279,54280 Player-Missile Base Register—PMBASE.
These locations contain the starting address of the player-missile definition tables.

ATARI HUE NUMBERS AND COLORS

COLOR	HUE	POKE VALUE
GRAY	0	0
LIGHT ORANGE	1	16
ORANGE	2	32
RED-ORANGE	3	48
PINK	4	64
VIOLET	5	80
VIOLET-BLUE	6	96
BLUE	7	112
MED. BLUE	8	128
LIGHT BLUE	9	144
TURQUOISE	10	160
BLUE-GREEN	11	176
GREEN	12	192
YELLOW-GREEN	13	208
ORANGE-GREEN	14	224
LIGHT ORANGE	15	240

In Atari BASIC, the SETCOLOR statement is used to control the color of its 5 (numbered 0-4) color registers. In Atari Microsoft BASIC the SETCOLOR statement can control 9 (numbered 0-8) color registers (registers 0-3 are colors of player-missiles; registers 4-7 are playfield colors; 8 is always the background register). The format is SETCOLOR register, hue, luminance.

To use the POKE values enter the number and add luminance (0-14). By adding the value of 0-14 to any of the above POKE values, the luminance will be increased.

KEYBOARD INTERNAL CODES—PEEK (764)

KEY	NORMAL	SHIFT	CONTROL	SHIFT/CONTROL
A	63	127	191	255
B	21	85	149	
C	18	82	146	
D	58	122	186	250
E	42	106	170	234
F	56	120	184	248
G	61	125	189	253
H	57	121	185	249
I	13	77	141	205
J	1	65	129	
K	5	69	133	
L	0	64	128	
M	37	101	165	229
N	35	99	163	227
O	8	72	136	200
P	10	74	138	202
Q	47	111	175	239
R	40	104	168	232
S	62	126	190	254
T	45	109	173	237
U	11	75	139	203
V	16	80	144	
W	46	110	174	238
X	22	86	150	
Y	43	107	171	235
Z	23	87	151	
0	50	114	178	242
1	31	95		223
2	30	94	158	222
3	26	90	154	218
4	24	88	152	216
5	29	93	157	221

KEY	NORMAL	SHIFT	CONTROL	SHIFT/CONTROL
6	27	91	155	219
7	51	115	179	243
8	53	117	181	245
9	48	112	176	240
<	54	118	182	246
>	55	119	183	247
–	14	78	142	206
=	15	79	143	207
;	2	66	130	
+	6	70	134	
*	7	71	135	
,	32	96	160	224
.	34	98	162	226
/	38	102	166	230
ATARI	39	103	167	231
ESC	28	92	156	220
BACKS	52	116	180	244
TAB	44	108	172	236
RETURN	12	76	140	204
LOWER	60	124	188	252
SPACE	33	97	161	225

CHARACTER SET VALUES

GRAPHICS 1/2 POKE 756,224	COLOR REG.0	COLOR REG.1	COLOR REG.2	COLOR REG.3	GRAPHICS 1/2 POKE 756,226
SPACE	32	0	160	128	♥
!	33	1	161	129	├
"	34	2	162	130	▕
#	35	3	163	131	┘
$	36	4	164	132	┤
%	37	5	165	133	┐
&	38	6	166	134	╱
'	39	7	167	135	╲
(	40	8	168	136	◢
)	41	9	169	137	▗
*	42	10	170	138	◣
+	43	11	171	139	▝
,	44	12	172	140	▘
–	45	13	173	141	▔
.	46	14	174	142	▁
/	47	15	175	143	▖

GRAPHICS 1/2 POKE 756,224	COLOR REG.0	COLOR REG.1	COLOR REG.2	COLOR REG.3	GRAPHICS 1/2 POKE 756,226
0	48	16	176	144	♣
1	49	17	177	145	┌
2	50	18	178	146	─
3	51	19	179	147	┼
4	52	20	180	148	●
5	53	21	181	149	▄
6	54	22	182	150	│
7	55	23	183	151	┬
8	56	24	184	152	┴
9	57	25	185	153	▌
:	58	26	186	154	└
;	59	27	187	155	␛
<	60	28	188	156	↑
=	61	29	189	157	↓
>	62	30	190	158	←
?	63	31	191	159	→
@	64	96	192	224	♦
A	65	97	193	225	a
B	66	98	194	226	b

GRAPHICS 1/2 POKE 756,224	COLOR REG.0	COLOR REG.1	COLOR REG.2	COLOR REG.3	GRAPHICS 1/2 POKE 756,226
C	67	99	195	227	c
D	68	100	196	228	d
E	69	101	197	229	e
F	70	102	198	230	f
G	71	103	199	231	g
H	72	104	200	232	h
I	73	105	201	233	i
J	74	106	202	234	j
K	75	107	203	235	k
L	76	108	204	236	l
M	77	109	205	237	m
N	78	110	206	238	n
O	79	111	207	239	o
P	80	112	208	240	p
Q	81	113	209	241	q
R	82	114	210	242	r
S	83	115	211	243	s
T	84	116	212	244	t
U	85	117	213	245	u

GRAPHICS 1/2 POKE 756,224	COLOR REG.0	COLOR REG.1	COLOR REG.2	COLOR REG.3	GRAPHICS 1/2 POKE 756,226
V	86	118	214	246	v
W	87	119	215	247	w
X	88	120	216	248	x
Y	89	121	217	249	y
Z	90	122	218	250	z
[	91	123	219	251	♠
\	92	124	220	252	|
]	93	125	221	253	↖
∧	94	126	222	254	◀
—	95	127	223	255	▶

Adventures with the ATARI®

Jack B. Hardy

Adventures with the ATARI® gets you started designing and writing adventure games of your own. There are six actual programs you can study, key in, and play — three interactive adventures and three graphical adventures. The games are written in three different programming languages —ATARI® PILOT™, ATARI® Microsoft BASIC™, ATARI® BASIC™ — to show you the flexibility and capabilities of each language that best fits your needs. The last part of the book is packed with tips and techniques, including programs, to guide you in creating your own computer adventures. You'll find out how to use all of the ATARI's capabilities to write games that are fun to play and provide countless hours of high adventure for everyone!

A Reston Computer Group Book
RESTON PUBLISHING COMPANY, INC.
A Prentice-Hall Company
Reston, Virginia

0-8359-0172-6