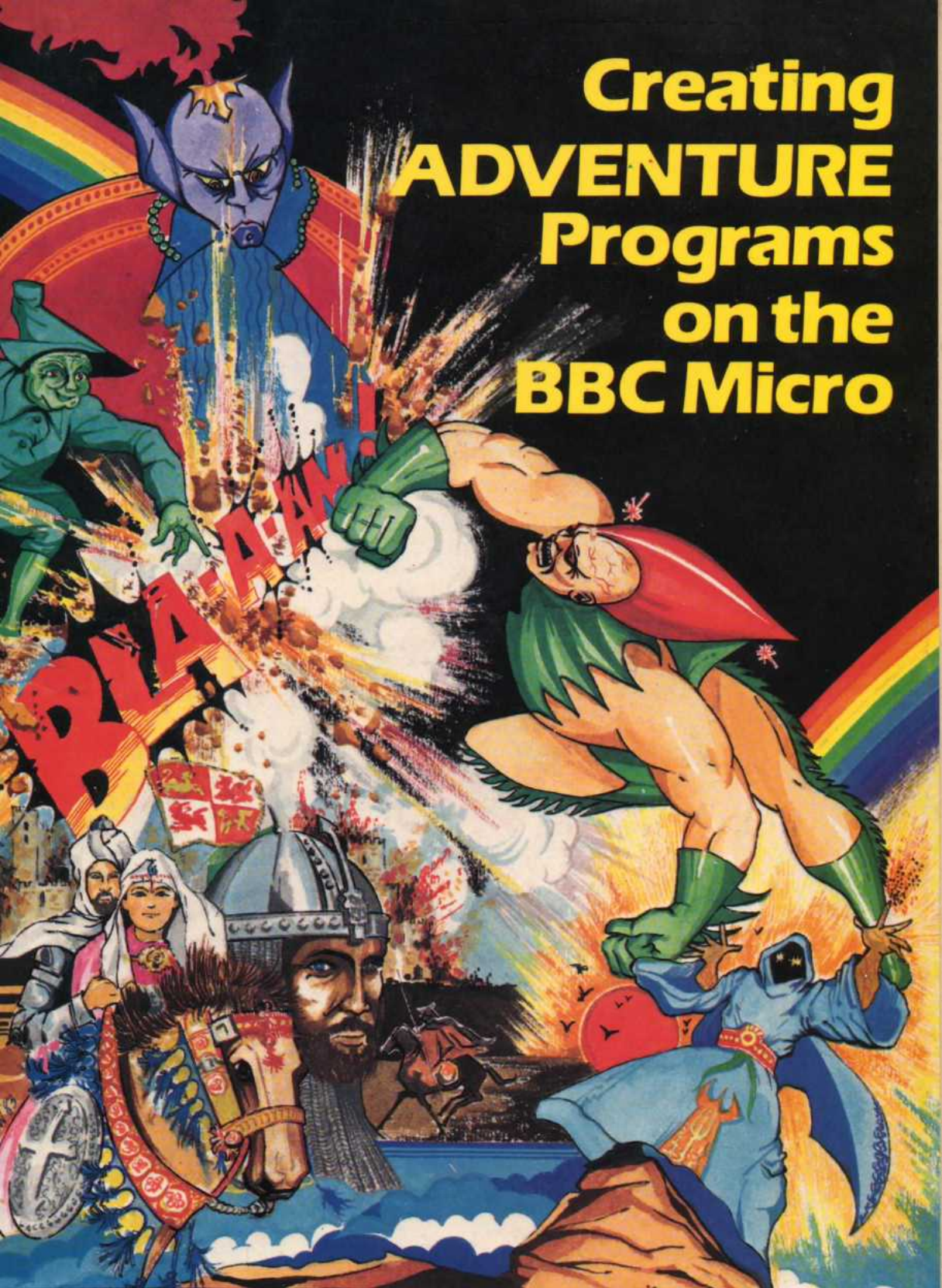


Creating **ADVENTURE** Programs on the **BBC Micro**



Ian Watt

**Creating ADVENTURE
programs on the
BBC MICRO
by
Ian Watt**

INTERFACE PUBLICATIONS
44-46 Earls Court Road
London W8 6EJ



ADDISON-WESLEY PUBLISHING COMPANY

London • Reading, Massachusetts • Menlo Park, California • Amsterdam
Don Mills, Ontario • Manila • Singapore • Sydney • Tokyo

© 1983 by Addison-Wesley Publishers Limited and
Interface publications

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior written permission of the publisher.

The programs presented in this book have been included for their instructional value. They have been tested with care but are not guaranteed for any particular purpose. While every care has been taken the publishers cannot be held responsible for running mistakes that may occur.

Typeset in England by Commercial Colour Press, London E7.
Printed in Finland by Werner Söderström Osakeyhtiö,
Member of Finnprint.

ISBN 0 201 14678 9

ABCDE 89876543

CONTENTS:

Foreword — Tim Hartnell	iv
Chapter One — The Adventurer	1
Chapter Two — A Model Adventure	4
Chapter Three — Captive — A complete adventure listing	35
Chapter Four — Conversion Notes	40
Chapter Five — Dracula — complete listing of another adventure	51
Chapter Six — Journey — complete listing of another adventure	67
Chapter Seven — Improving Adventures	80
Chapter Eight — Now write your own Adventures	83

APPENDIX:

One — The commands and their meanings	90
Two — The objects and their uses	99
Three — How to solve the adventures	111

FOREWORD

There's little chance that one day you'll get the opportunity in real life to explore labyrinthine cave systems, battle dragons or discover chests filled with gems and gold. But with your BBC Microcomputer, and this book by Ian Watt, you can at least experience all this, and more.

Since two programmers — Crowther and Woods — wrote the first adventure program on a mainframe computer at Stanford University over a decade ago, role-playing programs have proved firm favourites.

As you'll discover by reading and applying the ideas in this book, inventing adventure programs is an enthralling experience. You play the role of creator, devising and developing a complete mini-universe, complete with its own laws and logic. Once you've mastered the tricks of the adventure writing trade which are revealed in this book, you'll know how to invent environments which are self-consistent, mappable, and which can be discovered and explored by players.

Part of the immense interest which lies in adventure gaming comes from the sense of discovery players feel, as they work out how parts of the adventure 'universe' fit together; as clues are deciphered to solve the problems the program poses; until the game yields up its final secret.

It takes skill to create adventure programs which can provide lasting challenges to players, to write adventures which are challenging, satisfying and fun. In this book, Ian Watt shares the secrets of adventure-writing with you. And, as a bonus, there are a number of complete, ready-to-run adventure programs for you.

Down into the mist we go, stalking through the murky woods. Look out! What's that swooping down from the tree? And is that a diamond I see reflecting the starlight, at that bottom of that ravine?

The world of adventure awaits. Pluck up your courage now, and follow the guidance of Ian Watt into that wondrous world.

Tim Hartnell
April 1983

(Tim Hartnell is the author of a number of books for the BBC Micro, including 'The Book of Listings' (BBC Publications), 'Games BBC Computers Play' (Addison-Wesley /Interface) and 'Let Your BBC Micro Teach You to Program' (Interface Publications)).

CHAPTER 1

THE ADVENTURER

The Adventure is a type of computer game in which the player is placed in an imaginary environment and is required to direct the computer to manipulate this environment, often with the intention of escape, but sometimes with the object of collecting treasure. Normally, in Adventures, these directions are in the form of two words: the first word is a command, and the second is an object which the command may act upon. An example of this could be “GET REVOLVER”, where “GET” is the command, “REVOLVER” the object, and the intention is for the computer to pick up a revolver. The converse of picking objects up is dropping them down, the command being “DROP”. One Appendix of all the commands and their meanings, and another of all the objects used in the adventures in this book are given at the end.

Other types of directions are those involving movement, usually in the form of “N”, “S”, “E”, and “W” standing for north, south, east, and west, although “U” for up, and “D” for down are sometimes used. These movements are between locations or places, which are termed as “rooms” to anyone writing Adventures — this expression will be used throughout this book. This means that even though an area is in the open air, it is still called a room.

The relevant information is displayed on the computer screen. Firstly, the name of the room is printed out at the top of the screen — this may take the form of “Prison cell” or “Winding staircase”, both rooms being in the model adventure which will be examined in great detail in the next chapter. In the first adventures made on the large mainframe computers there were vivid descriptions for each room. However, this is not truly feasible on the microcomputer as there is not a great deal of memory available for such lengthy descriptions, and after the initial novelty of these wears off, the user gradually becomes uninterested in the technicalities as he or she only really requires the basic room names as markers for their whereabouts in the maze of rooms.

Secondly, the possible exits from the room in question are printed out. It may be expected that if one goes east and then back west then one will end up back in the same location. However, although this is true of most of the recent adventures, it was not true of many of the earlier adventures. This is a poor technique in programming as a player will be tempted to curse the programmer if he or she finds themselves in a completely different room from the one they would expect themselves to arrive in logically.

Next, the objects in the room are displayed followed by an inventory of what the player is carrying. Such a constant display of an inventory is yet to be seen in other adventures, although I have employed this feature from quite an early stage in the structures of my adventures. In other adventures it is necessary for one to type out 'INVENT' (or something similar) every time that one wishes to see what one is carrying. In my adventures, however, one can see all the time what one is carrying.

It is useful to have a limit to what one can drop in a room, or carry, at any one time, for the screen can become cluttered if a dozen or so objects have to be printed out in each category of 'Objects' and 'Inventory'. Many adventures have a limit in the inventory only, but not in the number of objects in each room. I define this limit as four for each category in adventures which have around twenty rooms; as the number of rooms approaches fifty or sixty, I raise this limit to six objects.

Below the printing of the inventory on the screen, an input is asked for. This is where the users may input their two-word commands, or their intended directions. In my adventures only the first three letters of each command and object need to be typed in, although the words may be entered in full if desired. This is quite a common feature as it reduces the amount of labour required in instructing the computer. For example, instead of typing out 'EXAMINE CEILING' in full, 'EXA CEI' may be entered. Another feature that I have included in the input procedure, is that if the space is accidentally omitted between the two words, then the computer will still accept the command provided that it is able to carry it out, although it will print out 'Learn to type'.

The final aspect of the display of the adventure is the printing out of the computer's reactions to the inputted two-word commands. If the computer replies in the affirmative to 'GET' or 'DROP', then it prints out 'O.K.'. On the other hand, however, it will print out 'I cannot do that' if it is unable to carry out your command. There are various similar reactions, depending on what situation arises.

Now that the syntax for the display of the adventure has been dealt with, the actual structure of the logic for it must be handled, for problems must be given to the player to solve — this is what makes the adventure interesting, and a great deal of pleasure can be found in solving a very tricky problem in an adventure. From the model adventure, for example, it is necessary for a magnifying glass to be carried, before writing, which is too small to read under normal circumstances, can be read.

The adventurer may gain information by reading any writing that may be found, or by examining objects. For example, in the model adventure, at

the “Observation point” , if one examines a window then one will gain the information that a space ship can be seen outside ready to take off.

A useful technique in adventure logic is to tempt the player into making actions which would not be too beneficial, although he or she may succeed in a problem in spite of yielding to this temptation. From the same adventure, in the “Bell tower” ,if one rings a bell, then the “dead” are woken, object to the noise, and kill you. This leads onto the topic of the death of a character in an adventure. This may occur in two forms, the first being the running out of time — the amount of time taken is the number of moves made. The use of such a time limit puts pressure on the player to complete the adventure in the most concise way. The other form of death arises when a particular wrong action is made, or the right one is not made.

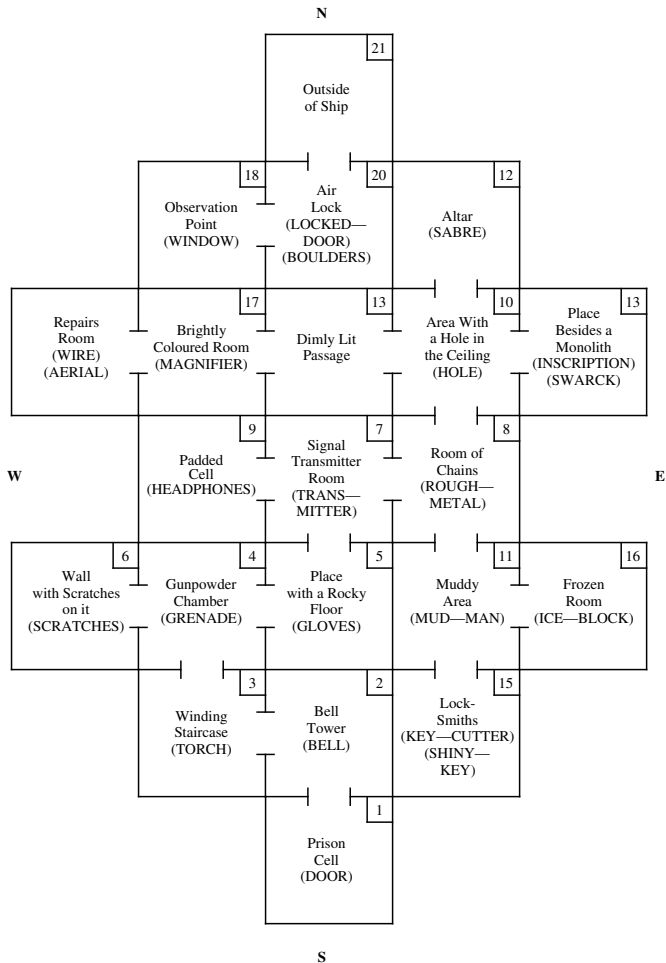
Now that the format of the adventure has been made clear, I would like to continue in the next chapter to show how these features are brought together in an adventure and also to show how such an adventure may be written along with the necessary routines which you will be able to apply to adventures which you may try to write yourself in the future.

CHAPTER 2

A MODEL ADVENTURER

The Plan

The first stage in writing an adventure is the making of a plan with the layout and labelling of the rooms with a system of numerical values for the objects in each room along with the room number and name:



As can be seen from the diagram, the names of the rooms are given along with the room number for each room, and the objects associated with the individual rooms are placed in brackets inside the room square.

Although the method for numbering rooms may at first appear not to be logical, the rooms are in fact in a logical order. Room “1” is where the player starts from, and room “2” is the first room that can be entered from this room. Rooms “5” and “6” are to the right and left of room “4” — I give preference in numbering to north then south then east and then west. I build up the numbering system by looking for the room with the smallest numerical value which has an exit, or exits, to an unnumbered room, or rooms (numbering is done in the above priority). The above method works because rooms beside each other have the lowest possible difference between their numerical values — this is a key feature, especially in larger adventures, in the movement between rooms, which is dealt with later in this chapter.

If this seems a little confusing, then the step by step process for deciding the numerical values for the rooms in the given example is as follows: the only exit from room “1” is room “2”, and the only unnumbered room from room “2” is room “3”. From room “3”, the only yet unnumbered room, is room “4”. However, from room “4” there are two unnumbered rooms, the eastern being numbered first as room “5”, and the western second as room “6”. At this point, the lowest numbered room is room “5”, and the only room off this room without a number is labelled room “7”. Room “6” is a dead end, so no further rooms may be numbered from this room. Room “7” is another room which has two unnumbered rooms off it, the eastern being labelled room “8”, and the western, which is a dead end, is labelled room “9”. From room eight, there are two rooms yet to be numbered — the northern is designated room “16”, and the southern, room “11”. The room with the lowest numerical value here, with other rooms off it wanting numbers, is room “1”, and rooms “12”, “13”, and “14” are to the north, east, and west of it — rooms “12” and “13” are both dead ends. After dealing with room “16”, the next room to be dealt with is room “11”: the rooms south and east of this room are two more dead ends, and are numbered with “15” and “16” respectively. After this fairly complex branching, the only room left that is not a dead end, with a yet unnumbered room off it, is room “14”, and the room off this, is given the next number, which is “17”. Two rooms requiring numbers are north and west of room “17”, and are labelled with “18” and “19”, the latter being a dead end. Off room “18” is room “2fl” to the east, and off room “2fj” to the north is the final room, room “21”.

WORKING OUT OF THE DISPLAY

1) Display of the Room Name — Type in the following lines into your computer (if you own a computer other than the BBC Micro, then see Chapter Four for conversion notes):

```

30      A=1
150     CLS
160     RESTORE
1310    DATA Prison cell, Bell tower, Winding staircase, Gunpowder
        chamber, Place with a rocky floor, Wall with scratches on it,
        Signal transmitter room, Room of chains, Padded cell, Area
        with a hole in the ceiling, Muddy area, Altar
1320    DATA Place beside a monolith, Dimly lit passage, Locksmiths,
        Frozen room, Brightly coloured room, Observation point,
        Repairs room, Air lock, Outside of ship

```

Variable “A” contains the room number for the adventure (i.e. the room in which the character is situated), and is defined as being equal to “1” in line 3a.

In line 150 the screen is cleared.

Line 160 ‘restores’ the data pointer to the beginning of the first item of data in the program, so that the next item of data read will be that after the first time that “DATA” appears in the program.

Line 300 is a FOR/NEXT loop which counts along the first items of data until the number in “A” is reached; when this number is reached, it returns with the name of the room corresponding to it, and this name is then stored in the variable “A\$”.

Line 310 has the job of printing out “A\$”, the name of the room. “VDU 31,0,3” moves the cursor three lines down from the top of the screen, and to the first character in the line — this is necessary on the BBC Micro, since the first line or two is sometimes off the television screen. “CHR\$130” is a control code which tells the computer to print in green alphanumerics — the semi-colon allows “A\$” to be printed directly after the control code.

Lines 1310 and 1320 are the first two lines of data in the adventure listing, and they contain the twenty-one room names, each of which may have a corresponding value in “A”.

If you are wondering why the program line numbers above are in irregular jumps, this is because there are lines omitted between the given lines which have functions other than the display of the room name. When the program at this early stage is run, the name of the first room (since “A” equals one) is printed at the top of the screen.

When you are asked to type in more lines of the program, then do not delete any existing lines of program, for the new lines will add to those

already there, and you will gradually see the program grow in both size and complexity. The complete listing of this program is given in Chapter Three for those people who may wish to type in the program all at the one time. If you do not come under this category, then it will allow you to make a reasonably quick check to see if you have missed out any of the line numbers in the sections, or even complete section which you will have to type in.

2) *A Display of the Exits from the Rooms* — Type in the following program lines:

```
320  PRINT'CHR$131"Exits:- ";RESTORE 1330:FOR C=1 TO
      A:READ D:NEXTC:IF D<>0 PRINT":North:";
330  RESTORE 1340:FOR C=1 TO A:READ D:NEXTC:IF D<>0
      PRINT":South:";
340  RESTORE 1350:FOR C=1 TO A:READ D:NEXTC:IF D<>0
      PRINT":East:";
350  RESTORE 1360:FOR C=1 TO A:READ D:NEXTC:IF D<>0
      PRINT":West:";
1330 DATA 1,0,1,0,2,0,0,2,0,2,-3,0,0,0,-4,0,1,0,0,1,0
1340 DATA 0,-1,0,-1,0,0,-2,3,0,-2,4,-2,0,0,0,0,0,-1,0,0,-1
1350 DATA 0,0,-1,1,0,-2,1,0,-2,3,5,0,0,-4,0,0,-3,2,-2,0,0
1360 DATA 0,1,0,2,-1,0,2,-1,0,4,0,0,-3,3,0,-5,2,0,0,-2,0
```

The above routine is the first of several fairly complex routines which I use when writing an adventure.

Line 320 prints out ‘Exits:-’ in yellow alphanumeric. The apostrophe after ‘PRINT’ tells the computer to go onto the next line down on the screen. The semi-colon after the final inverted commas allows the possible exits to be printed directly after this on the same line. The data pointer is then restored to line 1336, which contains the data for any movement north. The FOR/NEXT loop counts along the line of data until it reaches ‘A’, the room number, and stores the corresponding number, from the list of data, in the variable ‘D’. If there is an exit north, then ‘D’ will not equal zero, and so ‘:North:’ will be printed out — the semi-colon after this is used so that if there are any other exits from the room, then they will be printed on the same line.

Lines 330 340, and 350 have a similar function as line 320 — they do not, however, print out ‘Exits:-’ again — and instead of dealing with the direction north, the instead deal with the directions, south, east, and west.

Hence line 1340 contains the data for movement south, line 1350 holds the data for movement east, and line 1360 has the data for the direction west.

You may be wondering why the data statements hold various numbers other than zero, some of them being negative. This is so that the computer will know which room to go to, depending on whether it has been told to go north, south, east, or west from a particular room. However, at this stage of the program it is only necessary to know that if the value corresponding to a movement in one direction from a certain room is zero, then there is no exit from the room in that particular direction.

3) *A Display of the Objects in the Rooms* — Type in the following lines of program:

```
360 PRINT ' ' CHR$132"Objects:- ";
370 H=0:RESTORE 1370
380 FOR G=1 TO 22:READ C$:IF E(G)<>A OR H=4 NEXT G
    ELSE PRINT":";C$;";":H=H+1:IF H<>2 NEXTG ELSE
    PRINT" CHR$132" ";:NEXTG
1370 DATA GRENADE,ROUGH-METAL, SHINY-KEY, ICE-
    BLOCK, GLOVES, SABRE, AERIAL, TORCH,
    HEADPHONES, MAGNIFIER, LOCKED-DOOR, BELL,
    SCRATCHES, KEY-CUTTER, HOLE, TRANSMITTER,
    WINDOW, MUD-MAN, WIRE, INSCRIPTION, BOULDERS,
    SWARCK
```

However, it is necessary to enter the following lines as well at his stage so that the variables for the objects are initialised, thus preventing any errors from occurring:

```
20 DIM E(22)
30 A=1:T=0:W=0:RESTORE 1390:FOR B=1 TO 22:READ
    E(B):NEXTB
1390 DATA 4,8,22,16,5,12,22,3,9,17,20,1,2,6,15,10,7,18,11,19,13,22
```

I will deal with the initialisation of the variables for the objects first. The location of each object is governed by the values in the dimensioned variable of ‘E’, which is set to be able to hold twenty-two values in line 20. ‘SWARCK’ is a twenty-third objects, and it is in fact a magic word; it therefore cannot at any point be printed out in a room as an object, and so it does not require a location.

Line 30 is typed out again, but in addition to letting “A” equal one, there is a routine for setting the values in the dimension of ‘E’. Firstly, the data pointer is restored to the line of data in line 1390. Next, there is a FOR/NEXT loop from one to the number of objects, which, in this case, is twenty-two. The computer then reads each of the values in the data

statement, and stores them in the corresponding values of ‘E’. Hence E(1) will equal four, E(2) will equal eight, E(3) will equal twenty-two, and so on.

It should be obvious from the above information that most of the numbers in ‘E’ are between one and twenty-one for them to correspond to the appropriate room numbers of the rooms that the objects are in. The reasons for an object having a value which is not between one and twenty-one are as follows: firstly, if the number equals twenty-two (one more than the number of rooms), then that particular object has not yet been brought into the game, and will be brought into it later on; similarly, if an object is to be removed from the game, its value in the dimension of ‘E’ will be set to equal twenty-two. If the number in the dimension of ‘E’ equals zero, then the corresponding object is being carried by the player, and if the number equals negative one, then something extra will be attributed to the object in addition to it being carried. For example, a ‘TORCH’ will have a value of zero if it is being carried and if it is also lit, then its value will be negative one.

Now the printing of the objects in a room can be dealt with. Line 360 prints ‘Objects:-’ in blue alphanumerics, the two apostrophes putting the cursor down two lines before printing, and the semi-colon allowing the names of the objects to be printed on the same line.

Line 370 sets the variable ‘H’ equal to zero, and restores the data pointer to line 137 — ‘H’ will contain the number of objects in the room, and line 1370 contains the data for the names of the objects in the adventure.

Line 380 works out if an object is in the same-room as the adventurer, and if so, this object is printed out on the screen. This routine is based around a FOR/NEXT loop from one to twenty-two, twenty-two, as previously stated, being the number of objects. Each of the twenty-two object names are read and stored in C\$, and if the value in the dimension of ‘E’ for that object is not equal to ‘A’, the room number (the object is therefore not in the room), or if ‘H’ equals four (there would already be four objects printed out, and so no more would be printed under this category), then the computer goes back for the next object — when the last object is reached the computer continues with the rest of the program. On the other hand, if the object is in the room and there is space to print it out, then the machine does so, incrementing the value in ‘H’ before going back for the next object. If C\$ equals ‘GRENADE’ (assuming that it is also in the room and there is space to print it), then ‘PRINT’ ;“C\$;” ;” will print out “GRENADE:”. The semi-colon allows the next object to be printed on the same line, but if there have been exactly two objects already printed out, then there will not be enough room for a third object to be printed on the same line, and as the cursor is moved onto the next line and spaces are inserted so that the third object is printed under the first object.

4) *A Display of the Constant Inventory* — The inventory is printed out in a similar manner as the objects. However, there are less lines to be typed in here since the dimension of ‘E’ has already been assigned values, and the data for the object names need not be typed in a second time for the inventory —the lines that do have to be typed in are as follows:

```
390   PRINT' ' CHR$133"Inventory:- ";
400   F=0:RESTORE 1370
410   FOR G=1 TO 22:READ C$:IF E(G)<>0 AND E(G)<>-1 OR
      F=4 NEXTG ELSE PRINT":";C$;":";:F=F+1:IF F<>2 NEXTG
      ELSE PRINT"  CHR$133"      ";;NEXTG
```

Line 390 prints “Inventory:-” in magenta alphanumeric after putting the cursor two lines down the screen — from now on I will no longer refer to the use of the apostrophe and the semi-colon, since you should by now realise their uses in the program.

Line 400 sets variable ‘F’ equal to zero, and restores the data pointer to line 1370 — ‘F’ will contain the number of objects being carried, and line 1370 contains the data for the object names.

Line 410, like line 386, is built around a FOR/NEXT loop which covers the numbers between one and twenty-two for each of the objects concerned. The difference between the two categories lies in that it is the objects that are being carried that have to be printed out, and not the objects in the room. Again, C\$ contains the room name, and ‘G’ contains the object number. This time, if the value in ‘E’ is not equal to zero (the object is not being carried), and it is also not equal to negative one (the object is not one which is carried and has a special attribute), or if four objects are already printed out (if ‘F’ equals four), then the computer goes back for the next object. The rest of this part of the program is virtually the same as for line 386 apart from that the variable ‘F’ is concerned instead of the variable ‘G’, magenta alphanumeric are printed instead of blue alphanumeric, and a different number of spaces are required to put the third object in its correct place.

5) *A Display of the Input Line* — This is the final aspect of the initial display. Although the computer asks for an input, the replies to whatever is inputted will be dealt with later on in this chapter. The relevant lines for the input are as follows:

```
420   VDU 31,0,13,134:PRINT"[-----]
      -----]"
```



```

430   VDU 31,0,17,134:PRINT"[-----]
-----]:VDU 31,0,15,135
440   INPUT"Command? "B$
450   CLS:VDU 31,0,19,130

```

Line 420 moves the cursor to the first column and thirteen lines down from the top of the screen and prints CHR\$134, making the rest of the characters in the line become cyan in colour (“VDU” is more versatile than “PRINT”, in its control of various things to do with the screen although it is very similar in function — “PRINT” is used in most cases when printing things on the screen since its function is easier to understand). A line of minus signs is then printed on the screen with side arrows at each end which point away from the centre of the screen.

Line 430 is the same as line 426, with the exception that the line of minus signs in this case is printed seventeen lines down from the top of the screen, instead of fourteen. The cursor is then moved to the fifteenth line, and the colour of the characters on that line are set to white.

Line 440 asks for the input of a command, and stores whatever is typed in, in the variable “B\$”. The two preceding lines of program cordoned off (above and below) the line on the display which asks for the command.

Line 450 clears the screen and moves the cursor down to the nineteenth line, setting the character colour as green, ready for the printing out of the replies to whatever has been inputted.

WORKING OUT A ROUTINE SO THAT THE CHARACTER CAN MOVE FREELY BETWEEN ROOMS

The next stage in the adventure, now that the display has been taken care of, is the movement between rooms. It is necessary for the character to be able to move in the directions implied by the “Exits” from the room, and if a direction is typed in which is not implied, then “No exit” will be printed on the screen. The following lines containing the routines for movement “North”, “South”, “East” and “West”:

```

550   IF LEFT$(B$,1)<>"N" THEN 560 ELSE RESTORE
      1330:FOR C=1 TO A:READ D:NEXTC:IF D=0 THEN 600
      ELSE 590
560   IF LEFT$(B$,1)<>"S" THEN 570 ELSE RESTORE 1340:FOR
      C=1 TO A:READ D:NEXTC:IF D=0 THEN 600 ELSE 590
570   IF LEFT$(B$,1)<>"E" THEN 580 ELSE RESTORE 1350:FOR
      C=1 TO A:READ D:NEXTC:IF D=0 THEN 600 ELSE 590
580   IF LEFT$(B$,1)<>"W" THEN 610 ELSE RESTORE
      1360:FOR C=1 TO A:READ D:NEXTC:IF D=0 THEN 600
      ELSE 590

```

```
590   A=A+D:GOTO 160
600   PRINT"No exit!":GOTO 160
610   M=0:N=0:O=0
```

The data lines 1338 to 136{the ones regarding movement, and they have already been entered in the routine to print out the exits from a room. The main use for these lines of data will now be described.

Line 550 firstly checks to see if the inputted direction is north, and if not, it goes on to the next program line, which is line 560. However, if the direction entered is north, then the data pointer is restored to line 1330 and the value corresponding to movement north from the room "A", is read into the variable 'D', in the FOR/NEXT loop. If the value in 'D' equals zero, then there is no exit to the north, and so the computer jumps to line 60, where "No exit" is printed on the screen; there is then another jump which is back to line 16} to refresh the display for the room.

If there is an exit to the north then the value in "D" will not equal zero, and the computer will jump to line 596. At this line, the value in "D" is added to the value in "A", and the result stored in "A", so that the variable "A" equals the room number of the room which lies to the north of the room from which the direction was commanded. For instance, when one goes north from room "11" to room "8", the value in "D" will equal "-3" — this value of -3 is then added to "A", which equals "11", so that the final value of "A" will be "8", for $11 + (-3) = 8$. In short, "D" contains the number that must be added to "A" so that the new value of "A" corresponds to that of the room immediately north of that room corresponding to the previous value of "A". The computer then jumps to line 166 to refresh the screen display.

Lines 560, 570, and 580 perform the same function as line 550, except that they instead operate for south, east, and west respectively, and use the data in lines 1340, 1350, and 1360 the , which correspond to these movements.

Lines 590 and 600 have been explained with line 55}; line 610 is put in here so that a "no such line" error does not arise — if a direction is not typed in when the program is running at this stage, then the computer will jump to line 610 from line 580. The computer would become confused if it did not find this line, so it is just put in to keep the machine happy. In this line, the variables "M", "N", and "O" are defined as being equal to zero.

WORKING OUT ROUTINES FOR A RESPONSE TO THE INPUT AND CODING IN NUMERICAL VARIABLES, OF THIS INPUT

1) *Commands* — The first word of the two-word command to be entered is the actual verb, or command. The following routine works out the number of the inputted command from a line of data at line 1380, and stores the value in the variable “M” — there is also an error trap if the command entered is not recognised by the computer. The program lines are:

```
470     IF LEFT$(B$,3)="WEA" OR LEFT$(B$,3)="EXA" OR  
        LEFT$(B$,3)="SAY" THEN 610  
620     RESTORE 1380:FOR I=1 TO 16:READ C$:IF  
        LEFT$(B$,3)=C$ M=I  
630     NEXT I:IF M<>0 THEN 650  
640     PRINT"I do not understand you.":GOTO 160  
650     PRINT M  
1380    DATA GET,DRO,WEA,KIC,RIN,REA,CUT,EXA,KIL,LIG,  
        OPE,THR,SAY,TAK,UNL,LOO
```

Line 470 checks for those commands which start with the letters “N”, “S”, “E”, or “W”, jumping to the “COMMAND” routine directly, and bypassing the movement routine if such a command is found; if it is a recognised command which starts with any of these four letters, then there is no need to bother about the movement routine. However, if it is not a registered command, then it is interpreted as a direction, for if the first letter of a command typed in is “N”, “S”, “E”, or “W”, it is read by the movement routine to be one of these four directions, and hence a line is required to check that no recognisable commands are mistaken for directions.

In line 620 the data pointer is restored to line 1380, where the command names are stored. Note that it is the first three letters of each command that are stored in this line, so that only the first three letters of each command need to be typed in when playing the game. After this, there is a FOR/NEXT loop which reads each name in the data for command names, comparing each name with the first three letters of the inputted command. If the two correspond, then “M” is given the value, as a number between one and sixteen, for the command — the range of possible values for the commands will vary for different adventures.

Line 630 contains the NEXT statement of the FOR/NEXT loop, and checks to see whether or not “M” equals zero, for if no command has been matched with that inputted, then “M” will still contain the value of zero, as assigned in line 610. If “M” does not equal zero, then the computer goes on to line 650 for the routine to match up the object entered, which will be dealt with after this — at the moment the value in “M” is printed out on the screen in line 650, showing that the command number has been correctly

matched, once the computer has gone through the routine; this line will be overwritten by the next routine.

If no command has been matched with that entered, then the computer will continue onto line 640 where "I do not understand you" is printed on the screen; there is then a jump to line 168 to refresh the screen format.

Line 130 contains the data for the commands, but to avoid confusion about what the commands are in full and what their functions are, then look up Appendix I. At present, the computer should be able to accept any command that it recognises, and convert it into a numerical form in the variable 'M'. If it does not understand the command, then it should tell you.

2) *Objects* — The second word in the two-word command is an object — this is what the verb, or command, acts upon. The object is also assigned a numerical value, depending on what it is; the value is then stored in the variable 'O'. There is, as usual, a method for trapping possible errors. Type in the following lines:

```
650   RESTORE 1370:D$=RIGHT$(B$,3):FOR J=1 TO 23:READ
      C$:C$=LEFT$(C$,3)
660   FOR K=4 TO 12:IF LEFT$(D$,1)<>" " AND
      C$=MID$(D$,2,3) N=1
670   IF C$=MID$(D$,2,3) O=J:K=12:J=23:GOTO 680 ELSE
      D$=RIGHT$(B$,K)
680   NEXT K:NEXT J:IF O<>0 THEN 690 ELSE
      PRINT"Pardon?":GOTO 160
690   IF N=1 PRINT"Learn to type." CHR$130;
700   PRINT M,O
```

In line 650, the data pointer is restored to line 137J, where the object names are stored. D\$ is set equal to the first seven characters of B\$, minus the first three characters. There then follows the first part of a FOR/NEXT loop which encloses a good part of the next few lines. Each object is in turn read from the list in line 1370, and the first three letters are stores in C\$; this means that only the first three letters of each object need be typed in as well as for each command.

A routine is not required to find the object name in the entered string, for the command may have contained more than three letters, and so the computer has to look for where the object name starts; this is done with a FOR/NEXT loop in the lines 660 to 686. The computer initially checks to see if a space has been missed out by accident between the command and the object, and then checks to see if this object is valid. 'D\$' initially contains the four characters to the right of the second character in 'B\$', and

a check is made to see if the first of the four characters is not a space. If this is so, and the next three characters are the first three characters of an object name, then a space has been missed out, and ‘N’ is set equal to one. If the object name has not been found, then the fourth character onwards is taken of ‘B\$’ and so on, until ‘K’ equals ten, which is a suitably high number so that the length in characters of a command will not mask what the object is. When ‘K’ equals ten, the computer jumps back for the next object and continues with it.

If the object name matches up with that typed in, then ‘O’ is set equal to the object number, depending on the value of ‘J’ in the FOR/NEXT loop. If the computer comes out of this loop without having recognised your object, then ‘O’ will equal zero, and ‘Pardon?’ will be printed out before there is a jump back to line 168. If an object has been matched up, then ‘K’ is set equal to twelve, and ‘J’ is set equal to twenty-three, so that the computer will not have to continue through the loops unnecessarily.

If ‘N’ equals one — a space was missed out between the command and the object — then in line 690, ‘Learn to type’ will be printed, although the computer will still recognise your command and object.

In line 700, the values in ‘M’ and in ‘O’ are printed after the routines, since both commands and objects have been gone through, and the inputted command and object have been recognised. This allows one at this stage to check that these words have been correctly recognised. This line will be overwritten by another routine later on.

THE FILLING FOR THE ADVENTURE

Now that values have been assigned to variables for the inputted command and object, it is necessary for these values to be enacted upon. In doing so, problems are formulated for the adventure player to solve by inputting the relevant words. An adventure should be filled with problems to which there are logical solutions — each room should have at least one function apart from being simply a passage, although in some cases no functions suitably coincide with the room name, and so it just remains a ‘link’ room. This is probably the hardest part of writing adventures, since there are often many cross references which must be dealt with, and so it is necessary to gradually build up the actual adventure from the skeleton adventure, to which it has so far been built up to.

There now follows a number of program lines which contain the replies which are most frequently used when replying to commands which have been entered — each of these lines contains a PRINT statement with the relative information inside quotation marks, and this is followed by ‘GOTO 160’ which makes the computer jump back to refresh the display:

```

720 PRINT"I cannot do that.":GOTO 160
730 PRINT"O.K.":GOTO 160
740 PRINT"I am carrying too much.":GOTO 160
750 PRINT"I do not see it here.":GOTO 160
760 PRINT"I am not carrying it.":GOTO 160
770 PRINT"I do not see a place to put it.":GOTO 160
780 PRINT"I do not have them.":GOTO 160
790 PRINT"I do not see them here.":GOTO 160

```

When the above lines are typed in no apparent difference can be seen since the routines which use these lines have not yet been dealt with. The first of these routines is the ability to pick up objects, or ‘GET’ them. Type in the following lines:

```

700 ON M GOTO 800
800 IF O>10 THEN 720
810 IF F=4 THEN 740
830 IF E(O)<>A THEN 750
840 E(O)=0:GOTO 730

```

In line 700, if ‘M’ equals one, then the computer will jump to line 800 — ‘GET’ is the first command. However, if ‘M’ has any other value, then the computer will break out of the program with an ‘ON range’ error, since it has interpreted the entered command and has found nowhere to go with it; it then becomes confused and returns to the command mode. This means that while the program is running and all the commands have not yet been dealt with, the only commands that can safely be entered are those which have already been covered — the computer would be unable to provide a reply to such commands which cause these errors at this stage anyway. Line 700 will be updated every time another command is dealt with.

Line 800 checks to see whether an object is moveable or not. It is wise to have all moveable objects below a certain value, and those which are not moveable should be above that value to help separate them from each other; although this may work in theory, in practise it is hard to have such a fine line separating them, for at one point an unmoveable object may be dealt with, and at another point, one which can be moved may be dealt with, and so there is a tendency for an intermingling of the objects as the object vocabulary is dealt with. An example of an unmoveable object is a ‘DOOR’, and an example of a moveable object is a ‘TORCH’.

Line 810 sees if four objects are already being carried, for the variable ‘F’ contains the number of objects which are being carried by the player. If the player is unable to carry any more, then the computer will make a jump to line 740 to print out the relevant information depending on what the situation is.

In line 830, if the value in the dimension of ‘E’ for the variable ‘O’ does not equal the room number — the object is not in the same room as the player — then a jump will be made to line 756 to print out a reply to this. An object may only be picked up if the conditions are met to enable it to be picked up.

The value in the dimension ‘E’ for the variable ‘O’ is set equal to zero line 840, for since the conditions have been met, the player is allowed to carry the object he/she wanted to pick up. There is then a jump to line 730 for the affirmative ‘O.K.’ to be printed on the screen.

The next command to be analysed is ‘DROP’, which allows the player to drop an object, which has been picked up, in a room — not necessarily the room in which it was picked up. The lines that need to be typed in for this command are:

```
700    ON M GOTO 800,850
850    IF E(O)<>0 AND E(O)<>-1 THEN 760
860    IF H=4 THEN 770
870    E(O)=A:GOTO 730
```

If the command inputted is ‘DROP’, then ‘M’ will contain the value of two, and so the computer will go to line 856 from line 786, since 856 is the second number in the ‘ON M GOTO’ statement.

Line 850 checks to make sure that the object is being carried. Remember that an object may be carried under normal circumstances, or else it may have a special attribute, like a lit ‘TORCH’, for example; hence it could have its value in ‘E’ equal to zero or negative one. An object must be carried before it may be dropped, and so if it is not being carried then the computer jumps to line 700 to say so.

If there are already four items in the room then ‘H’ will equal four, and one would not be able to drop anything else in that room until something is picked up, and if this is the case, a jump is made in line 860 to line 770. ‘I do not see a place to put it’ would then be printed before the screen format is updated.

In line 870 the value in ‘E’ of ‘O’ is set equal to the room number so that if the player moves to another room, the object dropped would remain in the room in which it was dropped. The computer ends up by going to line 730 for a response in the affirmative.

Now that the more general commands of picking things up and dropping them down again, have been dealt with, the more specialised commands can now be covered — I will deal with them in the order in which they appear

in the program. The first of these commands is “WEAR” which allows the adventurer to wear any objects that are suitable for wearing; in this adventure two objects which may be worn are the “GLOVES” and the “HEADPHONES”. The following lines add this command to the commands “GET” and “DROP”, and the commands of movement:

```
700      ON M GOTO 800,850,880
880      IF O<>5 AND O<>9 THEN 720
890      IF E(O)=-1 PRINT"I am already wearing them.":GOTO 160
900      IF E(O)<>0 THEN 780
910      E(O)=-1:GOTO 730
```

The line number for the third command is added to line 7 so that when “M” equals three — i.e. the command is “WEAR” — the computer jumps to line 880 to check the context at that stage of the adventure and decide what should appear on the screen as a result.

Line 880 selects which objects are allowed with the command “WEAR”; remember that “O” contains the object number. Only objects five and nine — the “GLOVES” and the “HEADPHONES” — will be accepted. If any other object is typed in then the computer will jump to line 720, rejecting this object.

Line 890 checks to see if the object is already being worn, for if the command had been typed in previously for the “GLOVES” or the “HEADPHONES” they would already have the special attribute of being worn, and the value corresponding to them in “E” would equal negative one. This line is inserted to provide a conformation that the object is being worn.

If the object is not being carried at all, then this is taken care of in line 900, for if this is the case the “E” of “O” would not equal zero, since the possible value of negative one for carrying an object has already been dealt with. It therefore follows from this that it is only possible to wear an object if it is firstly in one’s possession.

Once the parameters have been met, the value in “E” of “O” can equal negative one giving the special attribute to the object. There is then a jump for a reply in the affirmative.

The fourth command on the list is “KICK” which forms the first real problem to be met by the adventurer, for in the first room, a prison cell, the only exit is through a door, and the problem lies in how to go through it. After fumbling about with various verbs, the adventurer may finally come across “KICK” — what else would you do to a door once you have become really frustrated with it? The following lines should enable you to try kicking the door yourself:


```

540   IF LEFT$(B$,1)="N" AND E(12)=A PRINT"The door is in the
      way.":GOTO 160
700   ON M GOTO 800,850,880,920
920   IF O<>12 THEN 720
930   IF E(O)<>A THEN 750
940   E(O)=22:PRINT" The hinges were weak and the door has "
      CHR$130"collapsed into a pile of dust.":GOTO 160

```

Line 540 is positioned before the movement routines so that no movement north can be made until the “DOOR” has been removed. The first character of the input is taken, and if it equals “N” for north along with “E” of twelve, the number corresponding to the object “DOOR”, equalling the room number that the adventurer is in, then “The door is in the way” will be printed out. The only room that the door can be found in, is the first room, and so if it is there then “E(12) = A”, where “A” has to equal one. There is then a jump to refresh the display, and the provision for movement is not allowed.

The line number corresponding to “M” equalling four is line 920, and in this line, any other object than “DOOR” will be rejected, since no other object has the value twelve.

In line 930 the computer makes sure that the door has not already been removed, for if “E” of twelve does not equal one — from the above “O” must equal twelve, and “A” must be one — then the “DOOR” is nowhere in sight and need not be bothered with.

Once the conditions have been satisfied, the “DOOR” is put out of action by setting its “E” value to be twenty-two, the number one greater than the number of room in the adventure. An explanation is then given for the disappearance of the door before the computer returns to the main program for the next command and object.

The next command I will deal with is “READ”, for I intend to leave out “RING” until later as it concerns a possible death for the adventurer, and all the deaths will be dealt with later on. The purpose of the command “READ” is to read information which is available for reading by the adventurer. In the model adventure it is possible to read “SCRATCHES” and also an “INSCRIPTION”. However, when reading the latter object it is necessary to have a “MAGNIFIER” in your possession, for otherwise “The writing is too small too read” will be printed out. Anyway, type in the following lines of programme:

```

700   ON M GOTO 800,850,880,920,720,980
980   IF O<>14 AND O<>21 THEN 720

```

```

990      IF A=6 THEN 1020 ELSE IF A<>13 THEN 790
1000     IF E(10)<>0 PRINT"The writing is too small to read.":GOTO
        160
1010     PRINT"The magic word is ' swarck' .":GOTO 160
1020     PRINT" A transmitted signal will allow a door"CHR$130"from
        the ' air lock'  to be opened.":GOTO 160

```

Since the sixth command is dealt with before the fifth, the line number corresponding to “M” equalling five is line 720 — at this line, “I cannot do that” is printed out — and this will be changed when “RING” is finally brought into the vocabulary. If “M” equals six then the computer will jump to line 980

In line 980 all other objects apart from “SCRATCHES” and “INSCRIPTION” will be rejected since no other object can be read. “SCRATCHES” will have its value in “O” as fourteen, and “INSCRIPTION” will have this value equal to twenty-one.

The next line, line 990, checks to make sure that the adventurer is in a position to read the objects and therefore be in the same room as the objects. If “A” equals six then the computer jumps to line 1020 to deal with what the “SCRATCHES” say, and any other value than thirteen, which is the room number of the “INSCRIPTION” will be discarded.

If the “MAGNIFIER” is not in the possession of the adventurer, then “E” of ten will not equal zero, and hence the “INSCRIPTION” will not be read. However, if this is carried, then the “magic word” will be divulged — what has to be worked out when playing the game is what this word should be used for, but that will be covered later.

The seventh command is “CUT”, and in this adventure, this allows a piece of “ROUGH-METAL” to be “CUT” into a “SHINY-KEY” provided that a “KEY-CUTTER” is in the room. However, in other adventures, “CUT” may have different uses, but this is the first time that I have used, or seen use of, this particular use of the command “CUT”. When writing an adventure you may find considerable pleasure in thinking up plausible and original uses of commands. I only have this one use of this command in this adventure, and this is detailed below:

```

700      ON M GOTO 800,850,880,920,720,980,1030
1030     IF O<>2 THEN 720
1040     IF E(O)<>0 THEN 740
1050     IF A<>15 PRINT"I see no place where it can be cut.":GOTO
        160
1060     E(2)=22:E(3)=0:PRINT" The piece of metal has been cut into
        a"CHR$130"key.":GOTO 160

```

Line 700 is again repeated, but this time it has an extra jump, on the condition that ‘M’ equals seven, to line 1030. In line 1030, if the object is not the ‘ROUGH-METAL’ — ‘O’ would not equal two — then the action would be rejected.

The computer then checks to see if the ‘ROUGH-METAL’ is being carried, for if it is not being carried then it cannot be cut. If ‘E’ of ‘O’ does not equal zero then this object is not being carried.

Next, the machine makes sure that the player is in the same room as the ‘KEY-CUTTER’, for if there is nowhere in sight where a key can be cut, then there is no way that it can be done. Therefore, if this is the case, then the action is dismissed.

In line 1060, once the conditions have been met, the ‘ROUGH-METAL’ is removed from the game and the ‘SHINY-KEY’ is brought into the possession of the player — ‘E’ of two is set equal to twenty-two, and ‘E’ of three is set equal to zero. The appropriate result is then printed on the screen before the computer returns for the next command.

The next and eighth command is ‘EXAMINE’ which allows the adventurer to see something in greater detail. In this adventure, if a ‘WINDOW’ is examined, then the reply that a spacecraft can be seen outside and is ready to take off, will be printed out. This is quite a versatile command which may reveal objects that were otherwise not visible to the player, but this use of it is not detailed in this adventure. Type in the following lines:

```
700      ON M GOTO 800,850,880,920,720,980,1030,1070
1070     IF O<>16 AND O<>18 THEN 720
1080     IF A=10 THEN 1100 ELSE IF A<>18 THEN 750
1090     PRINT" A space ship can be seen outside. It is"CHR$130"ready
        to take off.":GOTO 160
```

Line 700 adds the line number 1070 to the list of GOTO lines, and this corresponds to ‘M’ equalling eight. If any other object than the ‘WINDOW’ is inputted, then it will be rejected in line 1070. If the window is not in sight — the adventurer is not in the appropriate room — then the action would be rejected this time in line 1080. Finally, in line 1090, the result to the command would be printed out about the space ship.

Command number nine is ‘KILL’ which may be useful if any nasty creatures are in your way. Although you may not always be allowed to kill things, a ‘MUD-MAN’ in this adventure may be killed, but to do so it is imperative that the ‘SABRE’ is carried, for otherwise it will kill you, but

that will be dealt with later on. The necessary lines are:

```
700      ON M GOTO 800,850,880,920,720,980,1030,1070,1110
1110     IF O<>19 THEN 720
1120     IF E(O)<>A THEN 750
1130     PRINT"You have killed the mud-man.":E(O)=22:GOTO 160
```

Line 700 adds the ninth possible line number to the list, thus catering for the possibility of the value in ‘M’ being nine. Line 1110 makes sure that no other object than the ‘MUD-MAN’ is allowed, and it also certifies that the ‘SABRE’ is being carried. There is a check to make sure that the adventurer is in the same room as the ‘MUD-MAN’ in line 1120, and in line 1130 the ‘MUD-MAN’ is removed from the scene by letting ‘E’ of ‘O’ equal twenty-two. The result is then printed on the screen before a jump to refresh the display.

The next command is ‘LIGHT’ which allows the adventurer to light, for example, a ‘TORCH’ or a ‘LAMP’ — a ‘TORCH’ is used in this adventure — and so enable the player to move about without stumbling into anything in the dark. In the ‘Dimly lit passage’ adventurers must carry a lit ‘TORCH’ or else meet their fate; type in the lines below to allow this implement to be lit:

```
700      ON M GOTO 800,850,880,920,950,980,1030,1070,1110,1140
1140     IF O<>8 THEN 720
1150     IF E(O)=-1 PRINT"It is already lit.":GOTO 160
1160     IF E(O)<>0 THEN 760
1170     E(O)=-1:PRINT"It is now lit.":GOTO 160
```

Line 700 adds line 1140 as the tenth line number for the computer to jump to, since ‘LIGHT’ is the tenth command. The machine makes sure that no other object than the ‘TORCH’ is accepted in line 1140, and in 1150, a check is made to see whether or not the ‘TORCH’ has already been lit or not, and if it has not, then the computer will go on to the next line. If the ‘TORCH’ is not being carried then it cannot be lit, and so this would be rejected in line 1160. In line 1170 ‘E’ of ‘O’ is set equal to negative one to show that it has the special attribute of being lit. The appropriate reply is made before returning for the next command.

The eleventh command is ‘OPEN’, which allows a ‘LOCKED-DOOR’, or a ‘BOX’, or some similar object to be opened, but here, the only object that may be opened is the ‘LOCKED-DOOR’. Since there is already a ‘DOOR’, the prefix ‘LOCKED-’ is put in front of the word ‘DOOR’ to form another object by the name of ‘LOCKED-DOOR’. The program lines are:

```

700      ON M GOTO 800,850,880,920,720,980,1030,1070,
        1110,1140,1180
1180      IF O<>11 THEN 720
1190      IF E(O)<>A THEN 750
1200      IF E(3)<>0 PRINT"I have no key.":GOTO 160
1210      E(O)=22:E(22)=20:PRINT"The door came away in your
        hands,but" CHR$130"the exit is now blocked by
        boulders" CHR$130"which had been behind the door.":GOTO
        160

```

The line number corresponding to ‘M’ equalling eleven is added to the list of line numbers in line 76. This line number, line 1180, makes sure that no other object than the ‘LOCKED-DOOR’ is allowed. Line 1190 checks to see that the player is in the same room as the ‘LOCKED-DOOR’ and in a position to open it, provided that he/she is carrying the ‘SHINYKEY’ (this is checked for in line 1200). Now that the conditions have been met, the ‘LOCKED-DOOR’ is removed from the scheme, but the ‘BOULDERS’ are brought into the action.

The next commands is ‘THROW’ which allows an object to be thrown, the object in this adventure being a ‘GRENADE’. However, in other adventures, a popular use of this command is the ‘THROW’ a ‘ROPE’ — this ‘ROPE’ would then attach itself to a suitable point high up and become climbable; another way of manipulating a ‘ROPE’ in this fashion would be to ‘TIE’ it to a suitable point, but instead of being able to climb up, one would be able to climb down. Now return to the typing:

```

700      ON M GOTO 800,850,880,920,720,980,1030,1070,
        1110,1140,1180,1220
1220      IF O<>1 OR E(22)<>A THEN 720
1230      IF E(O)<>0 THEN 760
1250      E(1)=22:E(22)=22:PRINT" You have cleared a passage through
        the"CHR$130"boulders.":GOTO 160

```

Line 700 adds the line number for the twelfth command, ‘THROW’, to the list of line numbers. In this line to which the computer may jump, a check is made to see that it is the ‘GRENADE’ that is being thrown, and that the player is beside the ‘BOULDERS’ and therefore in a position to clear a passage through them. The next line, line 1236, makes sure that the ‘GRENADE’ is being carried, for if it is not, then it cannot be thrown. Line 125J removes the ‘GRENADE’ and the ‘BOULDERS’ from the scene before printing out the appropriate result.

“SAY” is a command which allows a player to say a magic word which could perform an action which would otherwise be impossible for the player to make. One use of saying such a word could be to reveal a hidden exit from a room, but the use for it in this adventure is to move the player from the outside of the space ship into it and make it take off, provided that the player is outside the spacecraft to begin with. The following lines allow this command to be entered while the program is running:

```
700      ON M GOTO 800,850,880,920,720,980,1030,1070,  
        1110,1140,1180,1220,1260  
1270     IF A<>21 PRINT"Nothing happens.":GOTO 160  
1280     PRINT" You have materialised inside your  
        ship"CHR$130"which has immediately taken off."  
1290     END
```

Line 700 adds, as usual, another line number to the list — the line that it adds is line 126} for the thirteenth command. This line, which the computer may jump to, checks to see if the right object, which is the magic word “SW ARCK”, has been entered. The only other condition that has to be met is that the player is in the right room, and this is dealt with in the next line. The result is printed out and the program ends in line 1290; this will be changed later on as there will be a provision for a score for playing the adventure.

The last three commands on the list are “TAKE”, “UNLOCK”, and “LOOK”, which act as alternatives to the commands “GET”, “OPEN”, and “EXAMINE” — it is useful in an adventure to have several commands which perform the same task since the player would then have a choice of commands, and would be able to choose the commands which he/she prefers to use. For example, one may prefer using the command “GET” to the command “TAKE”. The only line that needs to be typed in is an update of line 700:

```
700      ON M GOTO 800,850,880,920,720,980,1030,1070,  
        1110,1140,1180,1220,1260,800,1180,1070
```

Note that the last three line numbers correspond with those for the commands “GET”, “OPEN”, and “EXAMINE”, since “TAKE”, “UNLOCK”, and “LOOK” are alternatives to these commands.

The next feature of the adventure is the provision of various deaths which may be encountered by the adventurer. The first death that I will deal

with is the one which concerns the command “RING” which has been missed out until now. The player is tempted to “RING” a “BELL” which is in one of the rooms — what else would one do with a bell? This is, however, a red herring, and the noise from the “BELL” aggravates the inhabitants of the maze, and they kill the player who then has to start again. The following lines detail the routine:

```
700      ON M GOTO 800,850,880,920,950,980,1030,1070,  
        1110,1140,1180,1220,1260,800,1180,1070  
710      VDU 23;11,0;0;0;0,31,6,23:PRINT"Press space to start  
        again":IF INKEY$(50)=" " VDU 23;11,255;0;0;0:GOTO 30  
        ELSE VDU 31,6,23:PRINT"                ":IF  
        INKEY$(50)=" " VDU 23;11,255;0;0;0:GOTO 30 ELSE 710  
950      IF O<>13 THEN 720  
960      IF A<>2 THEN 750  
970      PRINT"You have woken the dead who do not  
        like"CHR$130"you too much.":GOTO 710
```

Line 700 is repeated for the last time with the line number corresponding to “M” equalling five and the command being “RING”. In line 950, where the machine would jump to, any other object than the “BELL” would be rejected. In the next line the computer then checks to see if the player is in the same room as the “BELL”. Line 970 the prints out the reason for the player’s death, and then jumps to line 710 which is the main part of the death routine.

The first part of line 710 with the assortment of numbers after the VDU statement removes the flashing cursor from the screen, and moves this now invisible cursor to the position on the screen twenty-three lines down and six lines along. “Press space to start again” is then printed out at this position — the computer waits for the space bar to be pressed, and if it is pressed within the time limit then the flashing cursor will be restored, and the machine will jump back to line 38 to start the game again. If the time limit ends, however, and the space bar has not been pressed, then the message is blanked over, and another time limit is set. The net effect of this is to have the message flashing On and off on the screen until the appropriate condition has been met, and then there will be a jump to the aforementioned line number.

The next few line numbers concern other deaths which are concerned with What commands and objects are typed in. Deaths concerning other situations will be dealt with after all the routines pertaining to the commands have been attended to. The line numbers for the remaining deaths directly related to the commands are as follows:

```

1070 IF O<>16 AND O<>18 THEN 720
1080 IF A=10 THEN 1100 ELSE IF A<>18 THEN 750
1100 PRINT" Something large has fallen through the"CHR$130"hole
and flattened you.":GOTO 710
1240 IF E(9)<>-1 PRINT" The noise from the explosion has
burst"CHR$130"your ear drums.The shock of this
has"" CHR$130"killed you.":GOTO 710

```

The first of these deaths occurs if one examines a ‘HOLE’ in the ceiling of one of the rooms, for something large would then fall out of the hole and flatten the player. Line 1070 makes sure that no other object than the ‘HOLE’ or the ‘WINDOW’ is examined, and if it is the ‘HOLE’ that is examined, then the appropriate message will be printed out before there is a jump to line 710 for the player to start again.

The second and last of these deaths is when the ‘GRENADE’ is thrown at the ‘BOULDERS’, for if the ‘HEADPHONES’ are not worn, then the shock of the noise from the explosion will result in death; if ‘E’ of nine does not equal negative one, then there will be no special attribute of the ‘HEADPHONES’ being worn. The result is printed out and the player will have to start again.

Another routine concerning the commands, but not with the death of the adventurer, is when he/she wishes to pick up a block of ice. The condition for doing this is, however, the wearing of a pair of gloves. The following line should be included in the ‘GET’ statement;

```

820 IF O=4 AND E(5)<>-1 PRINT"It is too cold to carry.":GOTO
160

```

This line checks, first of all, to see if the object entered is the ‘ICEBLOCK’, and then makes sure that the ‘GLOVES’ are being worn. If the conditions are not met then this object cannot be picked up in line 840.

When the player finishes the adventure then it is a good idea for a score to be given along with the best score obtained, and so the next few lines deal with this:

```

10 X=0
30 A=1:W=0:RESTORE 1390:FOR B=1 TO 22:READ
E(B):NEXTB
190 W=W+1
1290 Y=120-W:IF Y>X X=Y
1300 VDU 31,0,16,131:PRINT"Score=";Y;" Best
Score=";X:GOTO 710

```


Line 10 sets the variable “X” equal to zero, for this will contain the next best score for the game, and must be at the lowest score possible at the outset of the game, and therefore at zero. In line 30, the variable “W” is set equal to zero. This variable contains the number of moves made in the adventure, and this is incremented by one every time the computer passes line 190.

The score for a particular game is contained in the variable “Y”, and this is evaluated in line 1290 as the number “120” minus the number of moves made. If the present score is better than the best score, then it become the new best score in this same line. The next line prints out the relative scores before jumping to line 710 to restart the game for the player to try and better his/her score.

Now that all the routines pertaining to the commands in the main routine have been dealt with, I will now deal with another command which does not require an object. This routine is placed before the movement routines, and corresponds to the command “TRANSMIT” which allows a signal to be transmitted. This signal will open up an entrance to an ‘air lock’, but several parameters are required before this may be done, although these will be discussed below these lines which are relevant to this routine:

```

30      A=1:T=0:W=0:RESTORE 1390:FOR B=1 TO 22:READ
      E(B):NEXTB
250     IF A=7 AND E(7)=7 PRINTCHR$130"The transmitter is fully
      operational."
260     IF E(4)=7
      E(4)=22:T=1:E(7)=19:E(20)=22:PRINTCHR$130"The
      transmitter has cooled down,"" CHR$130"but it does not have an
      aerial."
280     IF A=7 AND T=0 PRINTCHR$130"The transmitter is
      overheating."
510     IF LEFT$(B$,3)<>"TRA" THEN 540
520     IF A<>7 OR E(7)<>7 THEN 720
530     T=2:PRINT" An entrance has appeared into the
      ' air"CHR$130"lock' .":GOTO 160

```

To begin with, the variable “T” is set equal to zero in line 30. This variable controls the state of operation of the transmitter. When this contains its Original value of zero, then the transmitter is in a state of overheating, and so Whenever one is in the appropriate room, then the message, as detailed in line 280, will be printed out.

However, wh the “ICE-BLOCK” is dropped in this room, then “E” of four will equal seven, and the transmitter will be cooled down, the ice will be removed from the scheme since it will have taken in the heat from the

transmitter and melted The piece of "WIRE" in room number nineteen is replaced by the "AERIAL", and the variable "T" is set equal to one — the transmitter is now working apart from it requiring the aerial as stated in line 260 of the adventure.

For the transmitter to be in a state of full operation, the "AERIAL" must be in the correct room, as checked for in line 250. The only thing now for the player to do is to transmit the signal. The entered command is checked in line 510, and if it does not correspond, then the computer goes onto the next routine. The machine then makes sure that the player is in the right room to transmit, and that the aerial is in that room. Now that all the conditions have been met, the variable "T" is given the value of two to show that the entrance has been revealed. Once this action has been made, there is then a jump back for the next command.

It is necessary for the players to be allowed to "QUIT" if they consider their relative positions to be desperate. This is quite an easy routine, for if the user simply types out the first three letters of the command "QUIT", then there is a jump to line 710 to restart the game. This is another command that does not require an object, and only the one line of program given below need be typed in:

```
460      IF LEFT$(B$,3)="QUI" THEN 710
```

There often arise several instances in adventures when the player is not allowed to go in certain directions for particular reasons. The two instances in this adventure are contained in the next two lines:

```
490      IF (LEFT$(B$,1)="N" OR LEFT$(B$,1)="S" OR  
          LEFT$(B$,1)="E") AND E(19)=A PRINT "You cannot pass the  
          mud-man.":GOTO 160  
500      IF T<>2 AND A=18 AND LEFT$(B$,1)="E" PRINT "You  
          cannot pass into the ' air lock' .":GOTO 160
```

If the mud-man is in the same room as the adventurer then no movement is allowed until this mud-man has been removed from the scene. The only way of removing the mud-man is by killing it with the "SABRE". If the signal has not been transmitted from the "Signal transmitter room" then "T" will not equal two. If this is the case and the player tries to go east from room eighteen then this will not be allowed until the signal has been transmitted.

There are two deaths which concern the room in which the adventurer is in and what is being carried. The first requires the "SABRE" being carried in the presence of the mud-man, or else death will result. The other has the need for carrying the lit "TORCH" in the "Dimly lit passage" or else the

player will fall down a hole in the poor light. Type in the following two lines:

```
270 IF A=11 AND E(6)<>0 AND E(19)=A PRINT"A mud-man has
just killed you.":GOTO 710
290 IF A=14 AND E(8)<>-1 PRINTCHR$130" You have fallen into
a hole in the" CHR$130"dim light.":GOTO 710
```

Line 270 must have the player not carrying the “SABRE” in the appropriate room, and the mud-man must be in that room before death will result. Line 290 requires “A” equalling fourteen, and “E” of eight not equalling negative one — the “TORCH” would not have the special attribute of being lit — before the adventurer is killed.

The one form of death left to be dealt with is the running out of time. In this adventure the player is allowed 120 moves in which to complete it before the planet, on which the player is situated, blows up. Various warnings are given depending on how many moves have been made. Once 12 moves have been made, the adventurer has to start again. The lines to be typed in are as follows:

```
190 W=W+1:IF W>20 AND W<40 PRINTCHR$130"A rumbling
sound can be heard."
200 IF W>39 AND W<60 PRINTCHR$130"The noise is becoming
louder."
210 IF W>59 AND W<80 PRINTCHR$130"The ground is starting
to shake."
220 IF W>79 AND W<100 PRINTCHR$130"I d advise you to get
out quickly."
230 IF W>99 PRINTCHR$130"The roof is caving in."
240 IF W=120 PRINTCHR$130"The planet has blown up.":GOTO
710
```

After the incrementing of the variable “W”, the computer checks for values between twenty and forty, printing out the message if the value in “W” lies within these limits. If not, then the computer may find a corresponding value and message in the next few lines, but if “W” has the value of 120, then there is no hope for the player and death results.

TIDYING UP THE PROGRAM

1) *Full Instructions* — When writing programs, full instructions are necessary in most cases, since anyone loading a certain program would be able to work out the basis of the program without too much confusion. The adventure is no exception, although most of the instructions are taken up with a storyline of the situation in which the adventurer is in before starting the game. The following lines contain the instructions for this adventure:

```

40  CLS:PRINT' ' ' CHR$129"Do you want the instructions(Y or N)
    ?";Z$=GET$:IF Z$="N" THEN 150 ELSE IF Z$="Y" THEN
    50 ELSE 40
50  CLS:PRINT' ' ' CHR$130" You have been captured by creatures
    on"CHR$130"an uncharted planet."
60  TIME=0:REPEAT UNTIL TIME>400
70  PRINT' ' ' CHR$131" Unfortunately the planet happens to
    be"CHR$131"unstable,and has been evacuated."
80  TIME=0:REPEAT UNTIL TIME>400
90  PRINT' ' ' CHR$132"You therefore have to escape before
    the"CHR$132"planet blows up with you on it."
    100 TIME=0:REPEAT UNTIL TIME>400
110 PRINT' ' ' CHR$133" The computer has a fairly large
    number"CHR$133"of commands,so therefore if one
    command"CHR$133"does not work then try another."
120 TIME=0:REPEAT UNTIL TIME>500
130 PRINT' ' ' CHR$134"The first three letters of each
    command"CHR$134"and object need be typed
    in,although,if"CHR$134"desired,the full word may be entered."
140 TIME=0:REPEAT UNTIL TIME>500

```

Line 40 gives the player the option of receiving the instructions or not: a letter is inputted and stored in the variable ‘Z\$’ — if this letter is ‘Y’ then the computer goes on to print the instructions, and if it is ‘N’ then the instructions are not printed out. However, if neither of these letters is inputted, then the question is asked again, and the process gone through until an appropriate reply is obtained.

Lines 50, 70 and 90 describe the situation that the player is in at the beginning of the adventure — this just provides a story for the player to continue. Lines 110 and 130 provide a short description on how to operate the game, although those people who have played adventures before will have the basic idea on how to play, for the same fundamental principles are transferred across most true adventures.

Lines 60, 80, 100, 120 and 140 are delay loops which give the player sufficient time in which to read each message before the next message is displayed. The size of the number at the end of each line will vary the amount of time before the computer prints out the next message. An alternative to the REPEAT/UNTIL loop could be the FOR/NEXT loop in the form; FOR Z = 1 TO TIME:NEXT Z, where ‘TIME’ is a number corresponding to the duration of time required for the player to read the message.

2) *Lower Case Graphics* — As you may have noticed, lower case text is freely mixed with upper case text. A lot of micros are unable to support this, and so if the one you possess does not, then just type in upper case — lower case is just more pleasing for the eye to read; it is not a terribly great disadvantage in not having this.

3) *Colour* — For those people with colour micros other than the BBC Micro, you may be interested in what the colours are corresponding to the CHR\$ codes that I have used, and so they are as follows:

CHR\$ 129 —red alphanumerics
CHR\$ 130 —green alphanumerics
CHR\$ 131 —yellow alphanumerics
CHR\$ 132 —blue alphanumerics
CHR\$ 133 —magenta alphanumerics
CHR\$ 134 —cyan alphanumerics
CHR\$ 135 —white alphanumerics

If your machine does not have colour, then just miss out these CHR\$ codes from the PRINT statements.

4) *Sound* — Sound is another that may be included in an adventure, but also only if your micro has this feature. Some people may find the sound aggravating, and so it may be missed out if desired. The purpose of the sound is to produce a changing tone which increases in volume and pitch as the player uses up more and more moves. The program lines for the BBC Micro are given below, and so using the information below these lines, along with the syntax for emitting sound on your own micro, the tone should be easily adapted for it:

```
30      A=1:T=0:W=0:RESTORE 1390:FOR B=1 TO 22:READ  
        E(B):NEXTB:SOUND 1,0,1,1  
170     ENVELOPE1,1,-1,1,-1,0,15,30,0,0,0,0,W,0  
180     SOUND1,1,W*2,1
```

Line 30 has a ‘SOUND’ command which has the volume set to zero, so that any lingering sounds from a previous game are cancelled for the start of a fresh game. Remember that the variable ‘W’ contains the number of moves already made: line 170 defines the wavering tone that is to be produced, and ‘W’ controls the volume of this tone. The next line, line 180, is the line that actually produces the sound, and ‘W’, in this case, controls the frequency. Th• means that with every increment of ‘W’, the pitch and volume of the sound increase. For those that understand the attack decay, sustain, and release of notes, there is a release value of zero in the envelope so that the tone is cotinuous.

VARIABLES

To ease the understanding of the program, a list of all the variables and their relative usage are given below:

1) *Numerical Variables*

- (i) A — Number of the room in which the adventurer is situated.
- (ii) B — Variable of a FOR/NEXT loop which is in one instance used in the reading of values for the positions of objects into the dimension of ‘E’, and in another instance for the reading of the room name of the room in which the player is in.
- (iii) C — Variable of a FOR/NEXT loop for reading if certain directions are to be printed on the screen and for seeing if the player can go in particular directions.
- (iv) D — The variable that decides if there is a certain exit from a room, and this depends on whether the value in it is, or is not, equal to zero; it will contain the value which must be added to ‘A’ to go in the chosen direction.
- (v) F — The number of objects that are being carried by the player.
- (vi) G — FOR/NEXT loop variable which counts for each object to see if it should be printed on the screen display format under either ‘Objects’ or ‘Inventory’.
- (vii) H — The number of objects that are lying in a room.
- (viii) I — FOR/NEXT loop variable used to help determine the number of the command entered by the player.
- (ix) J — FOR/NEXT loop variable which aids the determination of the number of the object entered after the command.
- (x) K — FOR/NEXT loop variable which allows for the input of the full number of letters in any command chosen by the adventurer.
- (xi) M — Variable in which the command number is stored.
- (xii) N — Depending on whether this variable contains the value zero or one, a space will either have been entered or missed out between the command and the object by the player.
- (xiii) O — Variable in which the object number is stored.
- (xiv) T — The value in this variable determines the status of the transmitter.
- (xv) W — The number of moves made by the adventurer are stored in this variable.
- (xvi) X — This contains the value of the best score.
- (xvii) Y — The current score is in this variable.
- (xviii) TIME — The variable for the computer’s internal clock.

2) *Dimensioned Variables* — The objects corresponding to the following possible dimensions of ‘E’ are given alongside each variable below:

- E(1) —GRENADE
- E(2) —ROUGH-METAL
- E(3) —SHINY-KEY
- E(4) —ICE-BLOCK
- E(5) —GLOVES
- E(6) —SABRE
- E(7) —AERIAL
- E(8) —TORCH
- E(9) —HEADPHONES
- E(10) —MAGNIFIER
- E(11) —LOCKED-DOOR
- E(12) —DOOR
- E(13) —BELL
- E(14) —SCRATCHES
- E(15) —KEY-CUTTER
- E(16) —HOLE
- E(17) —TRANSMITTER
- E(18) —WINDOW
- E(19) —MUD-MAN
- E(20) —WIRE
- E(21) —INSCRIPTION
- E(22) —BOULDERS

Remember that there is a full list of the commands and the objects in the first two Appendices at the end of the book along with detailed descriptions according to their relative functions.

3) *String Variables*

- (i) A\$ — The variable in which the room name is stored in. This name is displayed on the screen and updated every time the adventurer moves to another room.
- (ii) B\$ — A string which is inputted by the player and contains, or should contain, a command and an object, unless, however, a command which does not require an object is inputted.
- (iii) C\$ — The variable into which is read the names of the objects to be printed under ‘Objects’ and ‘Inventory’. This is also used in the determination of the numerical values for the commands and objects entered, for the first three letters of each command is stored in ‘C\$’, and then compared with the first three letters of the inputted command —the same applies for the objects.
- (iv) D\$ — This is used to help search for the name of the object entered in

‘B\$’, for the three letters starting from one character are compared with those in ‘C\$’ (‘D\$’ is what the characters from ‘B\$’ are stored in). Another three letters are then taken starting from the next character, and so on, until either the object name has been found, or the computer runs out of all possible characters to compare. The first three letters of the next command would then be stored in ‘C\$’.

- (v) Z\$ — This is the input for whether or not the player wishes to receive the instructions at the start of the game, with the acceptable replies being ‘Y’ and ‘N’.

THE PURPOSES CORRESPONDING TO EACH LINE OF DATA

1) *Lines 1310-1320* — In these lines, the data for the names of all the rooms in the adventure are stored.

2) *Line 1330* — the data for movement ‘NORTH’.

3) *Line 1340* — the data for movement ‘SOUTH’.

4) *Line 1350* — the data for movement ‘EAST’.

5) *Line 1360* — the data for movement ‘WEST’.

6) *Line 1370* — the names of all the objects used in the adventure are contained in this line.

7) *Line 1380* — this line contains the first three letters of the names of each command.

8) *Line 1390* — the data for the relative positions of each object at the beginning of the game.

CHAPTER 3

CAPTIVE

```

10 X=0
20 DIM E(22)
30 A=1:T=0:W=0:RESTORE 1390:FOR B=1 T
O 22:READ E(B):NEXTB:SOUND 1,0,1,1
40 CLS:PRINT'''CHR$129"Do you want th
e instructions(Y or N) ?";:Z$=GET$:IF Z$
="N" THEN 150 ELSE IF Z$="Y" THEN 50 ELS
E 40
50 CLS:PRINT'''CHR$130" You have been
captured by creatures on"CHR$130"an unc
harted planet."
60 TIME=0:REPEAT UNTIL TIME>400
70 PRINT'''CHR$131" Unfortunately the
planet happens to be"CHR$131"unstable,an
d has been evacuated."
80 TIME=0:REPEAT UNTIL TIME>400
90 PRINT'''CHR$132"You therefore have
to escape before the"CHR$132"planet blow
s up with you on it."
100 TIME=0:REPEAT UNTIL TIME>400
110 PRINT'''CHR$133" The computer has a
fairly large number"CHR$133"of commands
,so therefore if one command"CHR$133"doe
s not work then try another."
120 TIME=0:REPEAT UNTIL TIME>500
130 PRINT'''CHR$134"The first three let
ters of each command"CHR$134"and object
need be typed in,although,if"CHR$134"des
ired,the full word may be entered."
140 TIME=0:REPEAT UNTIL TIME>500
150 CLS
160 RESTORE
170 ENVELOPE1,1,-1,1,-1,0,15,30,0,0,0,
0,W,0
180 SOUND1,1,W*2,1
190 W=W+1:IF W>20 AND W<40 PRINTCHR$13
0"A rumbling sound can be heard."
200 IF W>39 AND W<60 PRINTCHR$130"The
noise is becoming louder."
210 IF W>59 AND W<80 PRINTCHR$130"The
ground is starting to shake."
220 IF W>79 AND W<100 PRINTCHR$130"I'd
advise you to get out quickly."
230 IF W>99 PRINTCHR$130"The roof is c
aving in."
240 IF W=120 PRINTCHR$130"The planet h
as blown up.":GOTO 710
250 IF A=7 AND E(7)=7 PRINTCHR$130"The
transmitter is fully operational."
260 IF E(4)=7 E(4)=22:T=1:E(7)=19:E(20
)=22:PRINTCHR$130"The transmitter has co
oled down,"'CHR$130"but it does not have
an aerial."
270 IF A=11 AND E(6)<>0 AND E(19)=A PR
INT"A mud-man has just killed you.":GOTO
710
280 IF A=7 AND T=0 PRINTCHR$130"The tr
ansmitter is overheating."
290 IF A=14 AND E(8)<>-1 PRINTCHR$130"

```

```

    You have fallen into a hole in the" 'CHR
$130"dim light.":GOTO 710
    300 FOR B=1 TO A:READ A$:NEXTB
    310 VDU 31,0,3:PRINTCHR$130;A$
    320 PRINT'CHR$131"Exits:- ";:RESTORE 1
330:FOR C=1 TO A:READ D:NEXTC:IF D<>0 PR
INT":North:";
    330 RESTORE 1340:FOR C=1 TO A:READ D:N
EXTC:IF D<>0 PRINT":South:";
    340 RESTORE 1350:FOR C=1 TO A:READ D:N
EXTC:IF D<>0 PRINT":East:";
    350 RESTORE 1360:FOR C=1 TO A:READ D:N
EXTC:IF D<>0 PRINT":West:";
    360 PRINT'CHR$132"Objects:- ";
    370 H=0:RESTORE 1370
    380 FOR G=1 TO 22:READ C$:IF E(G)<>A O
R H=4 NEXTG ELSE PRINT":";C$;":":H=H+1:
IF H<>2 NEXTG ELSE PRINT'CHR$132"
";:NEXTG
    390 PRINT'CHR$133"Inventory:- ";
    400 F=0:RESTORE 1370
    410 FOR G=1 TO 22:READ C$:IF E(G)<>0 A
ND E(G)<>-1 OR F=4 NEXTG ELSE PRINT":";C
$;":":F=F+1:IF F<>2 NEXTG ELSE PRINT'CH
R$133"
";:NEXTG
    420 VDU 31,0,13,134:PRINT"[-----
-----]"
    430 VDU 31,0,17,134:PRINT"[-----
-----]":VDU 31,0,15
,135
    440 INPUT"Command? "B$
    450 CLS:VDU 31,0,19,130
    460 IF LEFT$(B$,3)="QUI" THEN 710
    470 IF LEFT$(B$,3)="WEA" OR LEFT$(B$,3
)="EXA" OR LEFT$(B$,3)="SAY" THEN 610
    480 IF LEFT$(B$,1)="N" AND (E(11)=A OR
E(22)=A) PRINT"You exit is blocked.":GO
TO 160
    490 IF (LEFT$(B$,1)="N" OR LEFT$(B$,1)
="S" OR LEFT$(B$,1)="E") AND E(19)=A PRI
NT"You cannot pass the mud-man.":GOTO 16
0
    500 IF T<>2 AND A=18 AND LEFT$(B$,1)="
E" PRINT"You cannot pass into the 'air l
ock'." :GOTO 160
    510 IF LEFT$(B$,3)<>"TRA" THEN 540
    520 IF A<>7 OR E(7)<>7 THEN 720
    530 T=2:PRINT" An entrance has appeare
d into the 'air"CHR$130"lock'." :GOTO 160
    540 IF LEFT$(B$,1)="N" AND E(12)=A PRI
NT"The door is in the way.":GOTO 160
    550 IF LEFT$(B$,1)<>"N" THEN 560 ELSE
RESTORE 1330:FOR C=1 TO A:READ D:NEXTC:I
F D=0 THEN 600 ELSE 590
    560 IF LEFT$(B$,1)<>"S" THEN 570 ELSE
RESTORE 1340:FOR C=1 TO A:READ D:NEXTC:I
F D=0 THEN 600 ELSE 590
    570 IF LEFT$(B$,1)<>"E" THEN 580 ELSE
RESTORE 1350:FOR C=1 TO A:READ D:NEXTC:I
F D=0 THEN 600 ELSE 590
    580 IF LEFT$(B$,1)<>"W" THEN 610 ELSE
RESTORE 1360:FOR C=1 TO A:READ D:NEXTC:I
F D=0 THEN 600 ELSE 590
    590 A=A+D:GOTO 160
    600 PRINT"No exit!":GOTO 160
    610 M=0:N=0:O=0
    620 RESTORE 1380:FOR I=1 TO 16:READ C$
:IF LEFT$(B$,3)=C$ M=I
    630 NEXT I:IF M<>0 THEN 650

```

```

640 PRINT"I do not understand you.":GO
TO 160
650 RESTORE 1370:D$=RIGHT$(B$,3):FOR J
=1 TO 23:READ C$:C$=LEFT$(C$,3)
660 FOR K=4 TO 12:IF LEFT$(D$,1)<>" "
AND C$=MID$(D$,2,3) N=1
670 IF C$=MID$(D$,2,3) O=J:K=12:J=23:G
OTO 680 ELSE D$=RIGHT$(B$,K)
680 NEXT K:NEXT J:IF O<>0 THEN 690 ELS
E PRINT"Pardon?":GOTO 160
690 IF N=1 PRINT"Learn to type."'CHR$1
30;
700 ON M GOTO 800,850,880,920,950,980,
1030,1070,1110,1140,1180,1220,1260,800,1
180,1070
710 VDU 23;11,0;0;0;0,31,6,23:PRINT"Pr
ess space to start again":IF INKEY$(50)=
" " VDU 23;11,255;0;0;0:GOTO 30 ELSE VDU
31,6,23:PRINT"
":IF INKEY$(50)=" " VDU 23;11,255;0;0;0
:GOTO 30 ELSE 710
720 PRINT"I cannot do that.":GOTO 160
730 PRINT"O.K.":GOTO 160
740 PRINT"I am carrying too much.":GOT
O 160
750 PRINT"I do not see it here.":GOTO
160
760 PRINT"I am not carrying it.":GOTO
160
770 PRINT"I do not see a place to put
it.":GOTO 160
780 PRINT"I do not have them.":GOTO 16
0
790 PRINT"I do not see them here.":GOT
O 160
800 IF O>10 THEN 720
810 IF F=4 THEN 740
820 IF O=4 AND E(5)<>-1 PRINT"It is to
o cold to carry.":GOTO 160
830 IF E(O)<>A THEN 750
840 E(O)=0:GOTO 730
850 IF E(O)<>0 AND E(O)<>-1 THEN 760
860 IF H=4 THEN 770
870 E(O)=A:GOTO 730
880 IF O<>5 AND O<>9 THEN 720
890 IF E(O)=-1 PRINT"I am already wear
ing them.":GOTO 160
900 IF E(O)<>0 THEN 780
910 E(O)=-1:GOTO 730
920 IF O<>12 THEN 720
930 IF E(O)<>A THEN 750
940 E(O)=22:PRINT" The hinges were wea
k and the door has "CHR$130"collapsed in
to a pile of dust.":GOTO 160
950 IF O<>13 THEN 720
960 IF A<>2 THEN 750
970 PRINT"You have woken the dead who
do not like"CHR$130"you too much.":GOTO
710
980 IF O<>14 AND O<>21 THEN 720
990 IF A=6 THEN 1020 ELSE IF A<>13 THE
N 790
1000 IF E(10)<>0 PRINT"The writing is t
oo small to read.":GOTO 160
1010 PRINT"The magic word is 'swarck'."
:GOTO 160
1020 PRINT" A transmitted signal will a
llow a door"CHR$130"from the 'air lock'
to be opened.":GOTO 160

```

```

1030 IF O<>2 THEN 720
1040 IF E(O)<>0 THEN 740
1050 IF A<>15 PRINT"I see no place where
it can be cut.":GOTO 160
1060 E(2)=22:E(3)=0:PRINT" The piece of
metal has been cut into a"CHR$130"key."
:GOTO 160
1070 IF O<>16 AND O<>18 THEN 720
1080 IF A=10 THEN 1100 ELSE IF A<>18 THEN 750
1090 PRINT" A space ship can be seen outside.It is"CHR$130"ready to take off.":GOTO 160
1100 PRINT" Something large has fallen through the"CHR$130"hole and flattened you.":GOTO 710
1110 IF O<>19 THEN 720
1120 IF E(O)<>A THEN 750
1130 PRINT"You have killed the mud-man."
:E(O)=22:GOTO 160
1140 IF O<>8 THEN 720
1150 IF E(O)=-1 PRINT"It is already lit."
:GOTO 160
1160 IF E(O)<>0 THEN 760
1170 E(O)=-1:PRINT"It is now lit.":GOTO 160
1180 IF O<>11 THEN 720
1190 IF E(O)<>A THEN 750
1200 IF E(3)<>0 PRINT"I have no key.":GOTO 160
1210 E(O)=22:E(22)=20:PRINT"The door came away in your hands,but"CHR$130"the exit is now blocked by boulders"CHR$130"which had been behind the door.":GOTO 160
1220 IF O<>1 OR E(22)<>A THEN 720
1230 IF E(O)<>0 THEN 760
1240 IF E(9)<>-1 PRINT" The noise from the explosion has burst"CHR$130"your ears drums.The shock of this has"CHR$130"killed you.":GOTO 710
1250 E(1)=22:E(22)=22:PRINT" You have cleared a passage through the"CHR$130"boulders.":GOTO 160
1260 IF O<>23 THEN 720
1270 IF A<>21 PRINT"Nothing happens.":GOTO 160
1280 PRINT" You have materialised inside your ship"CHR$130"which has immediately taken off."
1290 Y=120-W:IF Y>X X=Y
1300 VDU 31,,16,131:PRINT"Score=";Y;"
Best Score=";X:GOTO 710
1310 DATA Prison cell,Bell tower,Winding staircase,Gunpowder chamber,Place with a rocky floor,Wall with scratches on it,Signal transmitter room,Room of chains,Padded cell,Area with a hole in the ceiling,Muddy area,Altar
1320 DATA Place beside a monolith,Dimly lit passage,Locksmiths,Frozen room,Brightly coloured room,Observation point,Repairs room,Air lock,Outside of ship
1330 DATA 1,0,1,0,2,0,0,2,0,2,-3,0,0,0,-4,0,1,0,0,1,0
1340 DATA 0,-1,0,-1,0,0,-2,3,0,-2,4,-2,0,0,0,0,-1,0,0,-1
1350 DATA 0,0,-1,1,0,-2,1,0,-2,3,5,0,0,-4,0,0,-3,2,-2,0,0

```

```

1360 DATA 0,1,0,2,-1,0,2,-1,0,4,0,0,-3,
3,0,-5,2,0,0,-2,0
1370 DATA GRENADE,ROUGH-METAL,SHINY-KEY
,ICE-BLOCK,GLOVES,SABRE,AERIAL,TORCH,HEA
DPHONES,MAGNIFIER,LOCKED-DOOR,DOOR,BELL,
SCRATCHES,KEY-CUTTER,HOLE,TRANSMITTER,WI
NDOW,MUD-MAN,WIRE,INSCRIPTION,BOULDERS,S
WARCK
1380 DATA GET,DRO,WEA,KIC,RIN,REA,CUT,E
XA,KIL,LIG,OPE,THR,SAY,TAK,UNL,LOO
1390 DATA 4,8,22,16,5,12,22,3,9,17,20,1
,2,6,15,10,7,18,11,19,13,22

```


CHAPTER 4

CONVERSION NOTES

These notes for the conversion of the adventures in this book so that they may work on microcomputers other than the BBC Micro, consist of two main parts: the first part relates to differences which correspond directly to commands on other micro, and the second concerns a major structural disadvantage for those computers which have no “READ” and “DATA” statements and/or the inability to support multi-statement lines, like the ZX81, for example. A reasonable knowledge of programming your micro is desired, but if this does not apply to you, and you are uncertain of what is and is not relevant to your micro, then you should be able to gain information concerning this from your computer manual. It is assumed that you have a reasonable amount of memory — 16 Kilobytes of RAM is considered reasonable. The conversion notes are about those commands used chronologically in the adventure “Captive”; these are the commands which may not be supported by other micros.

I. 1) *LET* — In line 10, the LET statement has been left out. It is optional on the BBC Micro, but it is necessary on some other micros like the ZX81. i.e. “X=0” would become “LET X=0”.

2) *RESTORE/READ* — If your computer does not have these commands then see section II of this chapter.

3) *'(apostrophe)* — This is a control for the position of the cursor on the screen — it moves it down to the beginning of the next line.

4) *CHR\$ (number) before inverted commas* — The colour that the text is to be printed in is controlled by the number. If your computer does not have colour then put an extra space in after the first set of inverted commas. If, on the other hand, your computer does have colour, then look at the manual for your computer if you are unsure about how to set the text colour in a program. Also note that the BBC Micro has the option of leaving out the brackets with CHR\$.

5) *CLS* — This clears the screen and homes the cursor to the top left hand corner. This is standard on most machines, but on the PET, for example, this is obtained by using an inverse heart shape inside inverted commas in a PRINT statement — the cursor movements on this computer are achieved by using other similar shapes which, like the heart shape, are obtainable from the keyboard.

6) *GET\$* — An alternative for this is INKEY\$, which has various syntax for different computers, so look at your manual if you are in doubt

about what to use. If your computer has neither of these, then a simple INPUT statement would be put in place of 'Z\$=GET\$' — i.e. 'INPUT Z\$'. Remember that the carriage return key must be pressed after the 'INPUT' statement while the program is running.

7) *THEN (line number)* — Some computers may require to have 'GOTO' between 'THEN' and the line number to make it become 'THEN GOTO (line number)'.

8) *ELSE* — This is a feature of structured BASIC, so if your computer does not have it, then do not worry. To avoid using it in line 40, for example, a new line would be taken after 'IF Z\$ = 'N' THEN 150':

```
40      IF Z$<> 'Y' THEN 40
```

9) *TIME/REPEAT/UNTIL* — 'TIME' is the variable on the BBC Micro that holds the value of the computer's internal clock. 'REPEAT/UNTIL' is similar to the 'FOR/NEXT' loop which can replace it: for example, 60 TIME=0:REPEAT UNTIL TIME>400 can be replaced with: 60 FOR B=1 to 5000:NEXT B. The number underlined should be changed according to the time delay that you require, so that you can read the information on the screen.

10) *ENVELOPE* — This allows the programmer to change the tone produced by the 'SOUND' statement. Leave this out if your computer does not have sound, or it does not have the ability to change the tone if it does have sound. If, on the other hand, you have a computer which does this, then include here the programming so that a wavering tone is produced whenever you use sound. The variable 'W' controls the volume and frequency of this tone.

11) *SOUND* — Sounds are produced with this command — 'W' controls the frequency at which the note starts, and the 'ENVELOPE' makes it in the form of a wavering tone. If the sound on your micro cannot take this sort of format then it might be best to leave it out completely, since it may not sound very well — 'ENVELOPE' can make the sound ring out continuously while the program is running, whereas 'SOUND' has a set duration, which would stop at some point when the computer is waiting for a command.

12) *IF W=120 PRINT* — The BBC Micro allows the word 'THEN' to be omitted between the '120' and 'PRINT'. Another example of this could be: IF E(4)=7 E(4)=22. If you are in doubt whether 'THEN' should be put in or not, then try applying then syntax 'IF (condition) THEN (expression)', and if it applies, then put it in. By omitting it on the BBC Micro, memory is saved.

13) *VDU 31,x,y* — The purpose of this command is to move the cursor to a location 'x' columns along, and 'y' rows down. The next thing to be printed will be printed at this location. A possible alternative for this on

other micros is the ‘PRINT AT’ statement which also moves the cursor to a selected position on the screen. If, on your computer, the layout looks better without leaving rows of space between the categories, then adjust this until you achieve a display which you like.

14) *VDU 31,0,13,134* — The third number after ‘VDU 31’ is to do with the colour that the text in the ‘PRINT’ statement following it is to be printed in — ‘VDU’ is similar to ‘PRINT’.

15) *LINES 420/430* — The input line is cordoned off by lines above and below it, so the characters in these lines, when printed out, should do this.

16) *INPUT “Command?” B\$* — If your computer does not accept this then it can be changed to: *440 PRINT “Command?”;*

```
440      INPUT B$
```

Some computers print out a question mark when asking for an input, so if this applies to your computer, then miss out the question mark after ‘Command’.

17) *LEFT\$(B\$,3)* — This is standard on most computers, but on the ZX’s, there is what I consider to be a slightly better syntax: instead of ‘LEFT\$’, ‘RIGHT\$’, and ‘MID\$’, it just has the one format. For example:

```
LEFT$(B$,3)becomes B$(TO 3)
RIGHT$(B$,3)becomes B$(3 TO)
MID$(B$,2,4)becomes B$(2 TO 6)
```

‘LEFT\$’ and ‘RIGHT\$’ should be easy enough to understand, since as their names suggest, they deal with the left and right parts of strings respectively. ‘MID\$’ deals with the middle of a string — the first of the number after the string name, is the number of characters from the left of the string which is being dealt with. The second number is the number of characters which will be in the substring.

18) *ON M GOTO* — Many computers have this but others do not. An alternative for this can be achieved by arranging the command routines in steps of line numbers which are of equal distance apart. If the step is fifty, then line 700 could become: *700 GOTO 750+M*50*. ‘If your’s does not accept this either then you will have to do it the long way. i.e.

```
700      IF M = 1 THEN 800
701      IF M = 2 THEN 850
702      . . . . .
```

On the other hand, you could try storing the line numbers in a ‘DATA’ statement and take the values out according to the value in ‘M’: *700 RESTORE (line number where the data is stored):FOR B=1 TO M:READ Z:NEXT B:GOTO Z*.

19) *LINE 710* — This is the line that requests that the game be started

again, and has the appropriate message flashing on and off — VDU 23;11,0;0;0 turns off the flashing cursor, and VDU 23;11,255;0;0 turns it on again. If you are in doubt how to use this on your computer then you can apply the following:

```
710 INPUT "Press carriage return to start again" Z$
712 IF Z$ <> "" THEN 710
714 GOTO 30
```

The computer will only respond if you press return and do not press anything else.

20) *LINE 1310* — With ‘DATA’ statements, many computers need to have words enclosed in inverted commas, whereas the BBC Micro does not.

II. I will now deal with structural problems concerning other computers. To begin with, if your computer does not have lower case text, then you will have no option, but to type things in completely in upper case.

If you are unable to apply the ‘READ’ and ‘DATA’ statements on your micro, then take note: instead of using ‘DATA’ statements you must store the information in strings. To begin with I will deal with the definition of the variables in ‘E’. The method of doing this takes up more memory, but it is simpler to apply, and will take the form of: E(1)=4:E(2)=8:E(3)=22:E(4)=16:E(5)E . . . :E(22)=22.

The information for what rooms the adventurer moves to on particular requested directions may be stored in the strings P\$, Q\$, R\$, and S\$. A suitable character will be chosen to represent zero (say CHR\$(50) on the ZX81) and the characters in the strings correspond to values above, below, or equal to zero which are added to fifty and changed into characters corresponding to these values. As an example for the movements north in ‘Captive’, I will make the values correspond directly for what they would be on the ZX81.

P\$ would equal ‘NMNMOMMOMOJMMMIMNMMNM’ from the DATA of 1,0,1,0,2,0,0,2,0,2,-3,0,0,0,-4,0,1,0,0,1,0.

On the ZX81, the routine to check and print out whether there is or is not a direction to the north is as follows:

```
320 PRINT
321 PRINT
322 PRINT "EXITS:-";
323 IF P$(A) < > "M" THEN PRINT "NORTH";
```

Also, concerned with the actual movement of the player, the routine for movement north could become:

```

550 IF B$(1)<>'N'THEN GOTO 560
551 IF P$(A)='M'THEN GOTO 600
552 LET A=A+CODE(P$(A))-50
553 GOTO 160
600 PRINT 'NO EXIT.'
601 GOTO 160

```

The routines for the other movements would be carried out in a similar manner —instead of dealing with P\$, Q\$, R\$, or S\$ would be dealt with.

You can store the room names in A\$, and have a full stop separating each name — this is CHR\$(27) on the ZX8 I. If the line in which A\$ is stored takes more than the screen size then you should not worry, since the ZX81 will allow this. However, to make editing easier, you may wish to have half of the room names in one string, and the other half in another string so that you will have two strings —A\$, and X\$, for instance:

```

LET A$='PRISON CELL.BELL TOWER. WINDING
STAIRCASE.GUNPOWDER CHAMBER.PLACE WITH A
ROCKY FLOOR.WALL WITH SCRATCHES ON IT.SIGNAL
TRANSMITTER ROOM.ROOM OF CHAINS.PADDED
CELL.AREA WITH A HOLE IN THE CEILING.'
LET X$='MUDDY AREA.ALT AR.PLACE BESIDE A
MONOLITH,DIMLY LIT
PASSAGE.LOCKSMITHS,FROZEN ROOM.BRIGHTLY
COLOURED ROOM.OBSERVATION POINT.REPAIRS
ROOM.AIR LOCK.OUTSIDE OF SHIP.'

```

The program lines to print out the appropriate room names are as follows:

```

300 LET BB=6
301 IF A<10 THEN GOTO 311
302 LET AA=1
303 IF AA=A THEN GOTO 307
304 LET BB=BB+1
305 IF CODE(A$(BB))=27 THEN LET AA=AA+1
306 GOTO 303
307 LET BB=BB+1
308 IF CODE(A$(BB))=27 THEN GOTO 320
309 PRINT A$(BB);
310 GOTO 307
311 LET AA=1
312 IF AA=A THEN GOTO 316
313 LET BB=BB+1
314 IF CODE(X$(BB))=27 THEN LET AA=AA+1
315 GOTO 312

```

```

316 LET BB=BB+1
317 IF CODE(X$(BB))=27 THEN GOTO 320
318 PRINT X$(BB);
319 GOTO 316

```

From the above programming information you will see how inefficient it is to use strings compared with “DATA” statements, because more program lines are required and the strings take up storage memory as well as the memory taken up by these extra lines of program. The full stop divides each piece of data, and the computer counts through this data by incrementing the variable “AA” every time it meets a full stop, and stopping when the value in “AA” corresponds with that in “A”. Each character of the string at that point will then be printed on the string until another full stop is met. When the value in “A” is changed then the different room name corresponding to that value will be printed out.

The next item of data to be sorted out concerns the names of the objects used in an adventure. These could be stored in J\$ as follows, or if there are a large number of objects in your adventure, you could store surplus names in K\$, but it is easier if you try to have them all together:

```

LET J$= 'GRENAD.E.ROUGH-METAL.SHINY-KEY.ICE-
BLOCK.GLOVES.SABRE.AERIAL.TORCH.HEADPHO
NES.MAGNIFIER.LOCKED-DOOR.DOOR.BELL.
SCRATCHES.KEY-CUTTER.HOLE.TRANSMITTER.
WINDOW.MUD-MAN.WIRE.INSCRIPTION.BOULDERS.
SWARCK.'

```

The first thing concerning the objects is printing them out under the categories of “Objects” and “Inventory”. Since the ZX81 has only thirty-two columns in the screen, it is easier for these two categories to be printed in two columns — one for “Objects” and the other for “Inventory” — side by side on the screen, for otherwise, the words would tend to overlap onto the next line. To prevent this when printing across the screen, I limit the number of letters for each object to about twelve — this can be increased to fifteen letters with two columns on the ZX81. The program lines to do this are as follows, and the values in “E” determine the presence or absence of objects:

```

360 PRINT
361 PRINT
362 PRINT AT 5,0; 'OBJECTS'; AT 5,16; 'INVENTORY'
363 PRINT '({7 GRAPHICS 7)'; AT 6,16; '{9 GRAPHICS 7)'
364 LET H=6
365 LET HH=0
366 FOR G=1 TO 22
367 LET HH=HH+1
368 IF E(G)=A THEN GOTO 372

```

```

369 IF CODE(J$(HH)<>27 THEN GOTO 367
370 NEXT G
371 GOTO 378
372 LET H=H+1
373 PRINT AT H+6,0;
374 PRINT J$(HH);
375 LET HH=HH+1
376 IF CODE(J$(HH))<>27 THEN GOTO 374
377 NEXT G
378 LET F=0
379 LET FF=0
380 FOR G=1 TO 22
381 LET FF=FF+1
382 IF E(G)=0 THEN GOTO 386
383 IF CODE(J$(FF))<>27 THEN GOTO 381
384 NEXT G
385 GOTO 420
386 LET F=F+1
387 PRINT AT F+6,16;
388 PRINT J$(FF);
389 LET FF=FF+1
390 IF CODE(J$(FF))<>27 THEN GOTO 388
391 NEXT G

```

- (i) *LINES 369–363* print up the titles for the categories.
- (ii) *LINES 364–377* print out the objects.
- (iii) *LINES 378–391* deal with the inventory.

The above program lines for each category are based around the FOR/NEXT loop of “G” which counts through all the objects, checking to see if they are in the room that the adventurer is occupying, in the first case, and in the second case there is a check to see if the objects are being carried. If an appropriate object has been found, then it is printed out from “J\$”, the character numbers being in the variables “HH” and “FF” — like the routine for the room names, the object names corresponding to the numbers are found through counting the full stops.

The next routines are the last routines which use strings instead of “DATA” statements; the inputted commands are interpreted by the strings, and the appropriate values are stored in the variables “M” and “O”. The program lines follow the string, I\$, in which the first three letters of each command are stored:

```

LET I$="GETDROWEAKICRINREACUTEX
AKILLIGOPETHRSAYTAKUNLLOO"

```

```

440 PRINT AT 21,0; "COMMAND?"
441 INPUT B$
610 LET M=0
611 LET N=0
612 LET O=0
613 LET C$=I$
614 LET D$=J$
615 IF LEN B$<6 THEN GOTO 440
620 FOR I=1 TO 16
621 IF B$(TO 3)=C$(TO 3) THEN LET M=M+1
622 LET C$=C$(4 TO)
623 NEXT I
630 IF M<>9 THEN GOTO 650
640 PRINT AT 16,; "I DO NOT UNDERSTAND YOU."
641 GOTO 160
650 FOR J=1 TO 23
651 LET E$=B$(3 TO)
660 FOR K=1 TO 7
661 IF LEN E$=4 THEN GOTO 681
662 IF E$(1)<>" " AND D$(TO 3)=E$(2 TO 4) THEN LET N=N+1
670 IF D$(TO 3)<>E$(2 TO 4) THEN GOTO 680
671 LET O=J
672 LET K=7
673 LET J=23
680 LET E$=E$(2 TO)
681 NEXT K
682 LET D$=D$(2 TO)
683 IF D$(1)<>" " AND LEN D$>3 THEN GOTO 682
684 LET D$=D$(2 TO)
685 NEXT J
686 IF O<> THEN GOTO 690
687 PRINT AT 16,J; "PARDON?"
688 GOTO 160
690 IF N=1 THEN PRINT AT 16,; "LEARN TO TYPE."

```

Although the above version of the routine looks a great deal longer than the version for the BBC Micro, it is not so much longer as you think since without the ability for several statements to appear on the one line, the number of line numbers increases dramatically. However, the amount of memory taken up does not change greatly.

The string, I\$, which, like the other strings, should be defined early on in the program, in the first few line numbers. There is no need in the variable I\$ for full stops to be inserted between each of the groups of three letters since the intervals between the beginnings of each of the groups is constant.

(i) *LINES 440–441* — This is the input routine.

(ii) *LINES 610–614* This is the definition of the variables for the routines. The variables C\$ and D\$ are set equal to I\$ and J\$ respectively. This is because the beginnings of the strings are deleted in gaining the next bit of data, and so it would not be beneficial to delete pieces of strings which the computer needs to check against every time a command and an object are entered.

(iii) *LINE 615* — This checks for the entered command and object being less than the minimum of six letters. With the ZX81, an error will result if the computer tries to check for more characters in a string than there are in it.

(iv) *LINES 620–641* — A check is made for the first three letters of each command equalling the first three letters of the entered command. The command number is stored in ‘M’, as applies for the version for the BBC Micro. If the command is not recognised then the ‘PRINT’ statement in line 648 comes into action.

(v) *LINES 650–685* — A search is made for the object name entered, and if it is found, then its number is stored in the variable ‘O’ — if not, the word ‘PARDON?’ is printed out in line 687. The method used is very similar to that used for the BBC Micro, the main difference being that the relative information is obtained from a long string with each item separated by a full stop, instead of from a list of data. The ZX81 syntax make it look different.

(vi) *LINES 686–690* — These lines check to see if the object has been recognised and that a space has been inserted between the command and the object, printing out the relative messages depending on the relative conditions.

The second structural problem inherent in the ZX81 is the inability to use more than one statement on a line. The best way to deal with this is to give examples, and from these you can apply the necessary changes to other lines in the programs:

EXAMPLE 1:- Change line 260 from:

```
260      IF E(4)=7 E(4)=22:T=1:E(7)=19:E(20 =22:PRINT  
        CHR$130‘The transmitter has cooled down,’CHR$130” but it  
        does not have an aerial.”
```

To:

```
260      IF E(4)<> 7 THEN GOTO 270  
261      LET E(4)=22  
262      LET T=1  
263      LET E(7)=19  
264      LET E(2)=22  
265      PRINT ‘THE TRANSMITTER HAS COOLED DOWN, BUT  
          IT DOES NOT HAVE AN AERIAL.’
```


You will see from the above example, the advantage of numbering line numbers in steps of ten, since other lines can be slotted in without much trouble. This is particularly necessary on the ZX81, where several lines may be taken up, compared with only the one line on other computers. Also, from this example, you will see the conversion notes coming into use: in line 260, "THEN GOTO" is used; the command "LET" is required in lines 261, 262, 263, and 264; and the "PRINT" statement in line 265 is in block capitals without colour, and it fits into the thirty-two characters per line, therefore corresponding to the ZX81 screen format. You may have noticed that where the assimilation of a condition spans over several lines, instead of checking for the condition being met, the computer checks for the condition not being met — this is particularly noticeable where the condition consists of two or more parameters, for otherwise memory would be taken up checking it in each line. This can be seen in the next example.

EXAMPLE 2:- Change line 820 from:

```

      820    IF O=4 AND E(5)<>-1 PRINT "It is too cold to carry.":
to either:      GOTO160
      820    IF O=4 AND E(5)<>-1 THEN PRINT "IT IS TOO COLD TO
                CARRY."
      821    IF O=4 AND E(5)<>-1 THEN GOTO 160
or:
      820    IF O<>4 OR E(5)=-1 THEN GOTO 830
      821    PRINT "IT IS TOO COLD TO CARRY."
      822    GOTO 160

```

Although the first version of line 820 for the ZX81 is contained in two lines, it is more clumsy than the second version, which requires three lines. There are several rules for the changing of a condition being met to the condition not being met, which are as follows: "AND" should be changed to "OR"; "OR" should be changed to "AND"; something equalling something else should be changed to the first thing not equalling the second; and where there is an inequality to begin with, an equality will take its place.

EXAMPLE 3:- Change line 270 from:

```

      270    IF A=11 AND E(6)<>0 AND E(19)=A PRINT CHR$130 "A
to:          mud-man has just killed you":GOTO 710
      270    IF A<>11 OR E(6)=0 OR E(19)<>A THEN GOTO 280
      271    PRINT "A MUD-MAN HAS JUST KILLED YOU."
      272    GOTO 710

```

This example is mainly just to concrete the points made in the two previous examples. You will soon find that it is not very difficult to convert multistatement lines to single statement lines.

A way of gaining some skill in converting programs, which are originally written for other computers, to work on your own computer, is to look for interesting programs in computer magazines, to make the necessary alterations so that the syntax is correct, and then to type them in. By trying this, you will help to develop your own skills as a programmer, since a greater effort will be required, than simply typing in programs which are written for the micro you use. Instead of keeping the programs just as they are in the magazines, you will find that there are bits in them which you think can be improved upon, and so you will have a greater impulse to modify them, since you will already be changing parts of them to suit your own micro. If your computer happens to have more features than the computer that the program was intended for, then you may wish to try and modify it to include your own computer's features — for example, if the program does not include colour, and you would be able to include this feature into it, then you would do so, provided that the resulting screen format turns out to be pleasing.

By following out such techniques you will gain a more general knowledge of programming in BASIC which you will be able to apply to virtually any computer that can be programmed in this language. If you decide to buy a better computer in the future, then you will be able to apply your old programming techniques almost from the word 'GO', and then expand your knowledge to include the features provided by that computer.

I trust that if you do not have a BBC Microcomputer, then the conversion notes in this chapter are sufficient for you to be able to use the programs with your own computer. The first section, which includes various smaller syntax details, and the second section, which includes structural details, deal with the possible ways in which your micro may differ from the BBC Micro. I have found that the BBC Micro was a good computer to base this book around, since it has many features, like colour and sound, which are becoming a standard in the more recent computers, and it also has a version of BASIC which is well structured and easy to use — the ZX81 is suitable for basing the structural details around, because it is at the lower end of the market, and lacks the features which the more expensive computers have: two major difficulties about it which I have tackled are the absence of 'READ' and 'DATA', along with the inability for multi-statement lines. In short, most other computers are upwards compatible to it, and so if there are conversion details which are specifically concerned with this computer, then conversion to other computers will not be much of a problem.

CHAPTER 5

DRACULA

```

10 DIM E(50):X=0
20 CLS:PRINT' 'CHR$129"Do you want the
  instructions(Y or N) ?";:Z$=GET$:IF Z$=
"N" THEN 130 ELSE IF Z$="Y" THEN 30 ELSE
20
30 CLS:PRINT' 'CHR$130"You have to esc
ape from the castle with"CHR$130"a previ
ous jewel,which Dracula removed "CHR$130
"from your kingdom."
40 TIME=0:REPEAT UNTIL TIME>500
50 PRINT' 'CHR$131"Dracula's guards th
rew you into a cell "CHR$131"on catching
you entering the castle."
60 TIME=0:REPEAT UNTIL TIME>400
70 PRINT' 'CHR$132"Your time is limite
d before your escape"CHR$132"from the ce
ll will be noticed."
80 TIME=0:REPEAT UNTIL TIME>400
90 PRINT' 'CHR$133" The computer has a
fairly large number"CHR$133"of commands
,so therefore if one command"CHR$133"doe
s not work then try another."
100 TIME=0:REPEAT UNTIL TIME>500
110 PRINT' 'CHR$134"The first three let
ters of each command"CHR$134"and object
need be typed in,although,if"CHR$134"des
ired,the full word may be entered."
120 TIME=0:REPEAT UNTIL TIME>500
130 RESTORE 2000:FOR B=1 TO 50:READ E(
B):NEXT B
140 CLS:A=1:W=0:R=1:B$="":S=0:T=0:U=0:
SOUND 1,0,1,1
150 RESTORE 840
160 ENVELOPE 1,1,-1,1,-1,0,15,30,0,0,0
,0,W/2.5,0
170 SOUND 1,1,W/2,1
180 IF A=55 THEN 270
190 W=W+1
200 IF W>34 AND W<70 PRINT CHR$130"Dra
cula has detected your presence."
210 IF W>69 AND W<105 PRINT CHR$130"Dr
acula is out to destroy you."
220 IF W>104 AND W<140 PRINT CHR$130"I
'd advise you to hurry up."
230 IF W>139 AND W<175 PRINT CHR$130"Y
ou do not have much time left."
240 IF W>174 AND W<210 PRINT CHR$130"D
racula is coming nearer."
250 IF W>209 PRINT CHR$130"He does not
intend to let you escape."
260 IF W=245 THEN 1620
270 IF A=54 AND E(9)<>0 PRINT CHR$130"
The dwarf kills you.":GOTO 1010
280 IF A=53 AND E(7)<>0 AND E(48)<>47
AND E(47)<>34 PRINT CHR$130"Dracula has
hypnotised you.":GOTO 1010
290 IF A=53 AND E(16)<>0 THEN 1620
300 IF A=11 AND B$<>"Y" PRINT CHR$130"
A computer asks if you think that this "

```

```

CHR$130"program is good(answer Y or N)."  

310 IF E(36)=49 AND E(43)=49 E(36)=56:  

PRINT CHR$130"The serpent has killed the  

dragon."  

320 IF A=28 AND E(44)<>-1 PRINT CHR$13  

0"You have fallen down the pit.":GOTO 10  

10  

330 IF A=28 PRINT CHR$130" Your rubber  

-boots give you grip on the"CHR$130"slip  

pery surface."  

340 IF A=23 AND E(14)=0 and E(2)<>0 E(  

14)=56:PRINT CHR$130"The jewel has falle  

n into the pool."  

350 IF A=14 AND R<4 PRINT CHR$130"The  

werewolf has awoken and killed you.":GOT  

O 1010  

360 IF A=55 AND E(14)<>0 PRINT CHR$130  

"You have failed to recover the jewel.":  

GOTO 1010  

370 IF E(40)=27 AND E(20)=27 E(40)=56:  

E(20)=56:E(21)=27:PRINT CHR$130"A skelet  

on key has been formed."  

380 IF E(23)=42 AND E(39)=42 E(23)=56:  

PRINT CHR$130"Your acid has dissolved th  

e web."  

390 IF A=35 AND E(4)<>0 OR A=24 AND E(  

10)<>56 PRINT CHR$130"You have been atta  

cked by wolves.":GOTO 1010  

400 IF A=35 AND E(10)=A PRINT CHR$130"  

Your lit-candles frighten the wolves."  

410 IF A<>32 E(31)=32  

420 IF A=7 AND E(1)<>0 PRINT CHR$130"A  

robot has killed you.":GOTO 1010  

430 IF A=7 AND E(30)=56 E(30)=7:PRINT  

CHR$130"A robot,seeing your gun,dropped  

a key."  

440 IF A=22 PRINT CHR$130"The ceiling  

has collapsed.":GOTO 1010  

450 IF A=26 AND E(13)<>0 PRINT CHR$130  

"You are hit by laser beams.":GOTO 1010  

460 IF A=26 PRINT CHR$130"Your shield  

deflects some laser beams."  

470 FOR B=1 TO A:READ A$:NEXT B  

480 VDU 31,0,3,130:PRINT A$  

490 PRINT'CHR$131"Exits:- ";:RESTORE 7  

70:FOR C=1 TO A:READ D:NEXT C:IF D<>0 PR  

INT":North:";  

500 RESTORE 780:FOR C=1 TO A:READ D:NE  

XT C:IF D<>0 PRINT":South:";  

510 RESTORE 790:FOR C=1 TO A:READ D:NE  

XT C:IF D<>0 PRINT":East:";  

520 RESTORE 800:FOR C=1 TO A:READ D:NE  

XT C:IF D<>0 PRINT":West:";  

530 PRINT''CHR$132"Objects:- ";  

540 H=0:RESTORE 810  

550 FOR G=1 TO 50:READ C$:IF E(G)<>A O  

R H=6 NEXT G ELSE PRINT": ";C$;": ";:H=H+1  

:IF H<>2 AND H<>4 NEXT G ELSE PRINT'CHR$  

132 " ";:NEXT G  

560 PRINT''CHR$133"Inventory:- ";  

570 F=0:RESTORE 810  

580 FOR G=1 TO 50:READ C$:IF E(G)<>0 A  

ND E(G)<>-1 OR F=6 NEXT G ELSE PRINT": ";  

C$;": ";:F=F+1:IF F<>2 AND F<>4 NEXT G EL  

SE PRINT'CHR$133 " ";:NEXT G  

590 VDU 31,0,15,134:PRINT"[-----  

-----]"  

600 VDU 31,0,19,134:PRINT"[-----  

-----]":VDU 31,0,17  

,135

```

```

610 IF A=55 E(9)=56:GOTO 1790
620 IF A=11 AND B$<>"Y" INPUT "Command?"
" B$:CLS:VDU 31,0,21:GOTO 150
630 INPUT "Command?" B$
640 CLS:VDU 31,0,21,130
650 IF E(22)=0 AND B$="X*Y+Z-5=W/V" E(
22)=56:E(40)=0:PRINT "A glue has been for
med.":GOTO 150
660 IF E(29)=56 AND LEFT$(B$,1)="N" PR
INT "The guard will not let you pass.":GO
TO 150
670 IF LEFT$(B$,3)="EXA" OR LEFT$(B$,3
)="WEA" OR LEFT$(B$,3)="EMP" THEN 900
680 IF LEFT$(B$,1)<>"N" THEN 690 ELSE
RESTORE 770:FOR C=1 TO A:READ D:NEXT C:I
F D=0 THEN 760 ELSE 720
690 IF LEFT$(B$,1)<>"S" THEN 700 ELSE
RESTORE 780:FOR C=1 TO A:READ D:NEXT C:I
F D=0 THEN 760 ELSE 720
700 IF LEFT$(B$,1)<>"E" THEN 710 ELSE
RESTORE 790:FOR C=1 TO A:READ D:NEXT C:I
F D=0 THEN 760 ELSE 720
710 IF LEFT$(B$,1)<>"W" THEN 900 ELSE
RESTORE 800:FOR C=1 TO A:READ D:NEXT C:I
F D=0 THEN 760
720 IF A=32 AND E(31)=A PRINT "The plan
t will not let you pass.":GOTO 150:ELSE
IF A=42 AND E(23)=A PRINT "You are caught
in the web.":GOTO 150
730 IF A=44 AND E(43)=44 AND LEFT$(B$,
1)="S" PRINT "The serpent is blocking the
exit.":GOTO 150:ELSE IF A=47 AND E(12)<
>56 AND LEFT$(B$,1)="S" PRINT "You cannot
pass the flames.":GOTO 150
740 IF A=49 AND E(36)<>56 AND LEFT$(B$
,1)="S" PRINT "The dragon will not move.":
GOTO 150:ELSE IF A=17 AND E(46)=A AND (
LEFT$(B$,1)="N" OR LEFT$(B$,1)="E") PRIN
T "The exit is sealed.":GOTO 150
750 IF E(15)=45 AND A=45 AND LEFT$(B$,
1)="W" OR E(15)=48 AND A=48 AND LEFT$(B$,
1)="N" OR A=50 AND (E(17)=0 OR E(18)=0
OR E(19)=0) PRINT "You cannot pass the gh
ost.":GOTO 150:ELSE A=A+D:GOTO 150
760 PRINT "No exit!":GOTO 150
770 DATA 1,0,2,3,0,-3,3,-4,2,0,4,0,4
,0,5,5,0,-5,-5,0,0,7,7,7,-7,0,-8,0,0,4,0
,0,0,0,0,3,0,0,0,-4,3,-4,0,0,-3,-3,2,-2,
0,-2,0,0,0,1
780 DATA 0,-1,3,4,-2,0,-3,0,0,-3,-2,-2
,0,5,5,-4,0,-4,7,8,-5,0,0,0,0,0,0,-7
,-7,-7,0,0,-4,0,4,0,4,-3,0,0,3,-3,0,2,
0,2,-2,0,0,0,1,0
790 DATA 0,2,-1,0,0,0,0,0,-4,4,2,6,-3,
0,-4,7,-5,7,8,0,0,0,0,-7,0,7,0,6,0,6,0,5
,5,0,-6,0,0,4,-4,4,0,0,0,0,0,0,0,-3,0,0,
0,-2,0,0,0
800 DATA 0,1,0,-2,4,0,0,0,0,3,4,0,-2,-
4,0,6,7,-6,0,0,0,0,-7,0,-7,0,-8,0,6,0,0,
0,-7,-6,4,-6,-5,-5,0,0,0,-4,0,-4,3,0,0,0
,0,2,0,0,0,0,0
810 DATA GUN,WALLET,CONTAINER,LIT-CAND
LES,COFFIN,WINE-LABEL,CRUCIFIX,LASER-GUN
,SILVER-COIN,WOLVES,COMPUTER,BUCKET,SHIE
LD,JEWEL,GHOST,STAKE,SWORD,DAGGER,AXE,BR
OKEN-BONES,SKELETON-KEY,CHEMICALS,SPIDER
'S-WEB,CLUE,MUSIC,RECORDER,WRITING
820 DATA GUARD,DEAD-GUARD,KEY,PLANT,CU
PBOARD,ROPE,DWARF,DOOR,SAD-DRAGON,DRACUL
A,MIRROR,ACID,GLUE,CLOCK,WEREWOLF,SERPEN

```

```

T, RUBBER-BOOTS, SWITCH, SEALED-EXITS, GARLIC,
VAMPIRE, MALLETT, MEAT-CHOPPER
830 DATA GET, DRO, KIL, REA, PLA, KIC, OPE, M
IX, EXA, WEA, EMP, PRE, BLA, TIE, LOW, CLI, CUT, L
IF, HAM, CHO
840 DATA Cell, Corridor, W.End of corrid
or, E.End of corridor, Library, Cell, Dining
room, Outside of a locked door, Study
850 DATA S.End of main hall, Computer r
oom, Main hall, Living room, Werewolf's cha
mber, Candlelit room, N.End of main hall, C
hamber of suspense, Chamber of horror
860 DATA Weapon room, Wine room, Music r
oom, Chamber of death, Room with a pool, Bu
rial chamber, Armour room, Torture chamber
, Skeleton chamber
870 DATA Room with a pit, Room with sol
dier ants, Room with a clock, Room with a
sarcophagus, Bright room, Workroom, Dracula
's bedroom, Room with cages, Kitchen
880 DATA Room of mirrors, Laboratory, Se
rvants' quarters, Echo chamber, Electric ge
nerator room, Room with a giant spider, Ch
amber of fear, Room with a serpent, Dull r
oom
890 DATA Room with a clue, Room with a
fire in it, Dim room, Room with a dragon, D
ark room, Treasure chamber, Room which is
pitch black, Room with a hole, Room with a
dwarf, Outside of castle
900 M=0:N=0:D=0:IF LEFT$(B$,3)="QUI" T
HEN 1010
910 RESTORE 830:FOR I=1 TO 20:READ C$:
IF LEFT$(B$,3)=C$ M=I
920 NEXT I:IF M<>0 THEN 940
930 PRINT"I do not understand you.":GO
TO 150
940 RESTORE 810:D$=MID$(B$,3,4):FOR J=
1 TO 50:READ C$:C$=LEFT$(C$,3)
950 FOR K=4 TO 10:IF LEFT$(D$,1)<>" "
AND C$=MID$(D$,2,3) N=1
960 IF J=1 AND LEFT$(D$,1)="-" K=10:GO
TO 980
970 IF C$=MID$(D$,2,3) O=J:K=10:J=50 E
LSE D$=MID$(B$,K,4)
980 NEXT K:NEXT J:IF O=0 PRINT"Pardon?"
:GOTO 150
990 IF N=1 PRINT"Learn to type." 'CHR$1
30;
1000 ON M GOTO 1050,1110,1150,1220,1280
,1320,1350,1430,1460,1510,1550,1590,1630
,1670,1720,1760,1810,1850,1890,1950
1010 VDU 23;11,0;0;0;0;0,31,6,10:PRINT"Pr
ess space to start again":IF INKEY$(50)=
" " VDU 23,11,255;0;0;0;0:GOTO 20 ELSE VDU
31,6,10:PRINT"
":IF INKEY$(50)=" " VDU 23;11,255;0;0;0
:GOTO 20 ELSE 1010
1020 PRINT"I cannot do that.":GOTO 150
1030 PRINT"O.K.":GOTO 150
1040 PRINT"I am not carrying it.":GOTO
150
1050 IF O=5 OR O=10 OR O=11 OR O=15 OR
O=20 OR O=23 OR O=24 OR O=25 OR O=27 OR
O=28 OR O=31 OR O=32 OR O=34 OR O=36 OR
O=37 OR O=39 AND E(3)<>0 OR O=41 OR O=42
OR O=45 OR O=46 OR O=47 AND T=0 OR O=48
OR O=16 AND E(48)=34 THEN 1020
1060 IF F=6 PRINT"I am carrying too muc
h.":GOTO 150

```

```

1070 IF E(O)=A E(O)=0:GOTO 1030
1080 PRINT"I do not see it here.":GOTO
150
1090 PRINT"I do not see them here.":GOT
O 150
1100 PRINT"I am not carrying them.":GOT
O 150
1110 IF E(O)<>0 AND E(O)<>-1 THEN 1040
1120 IF H=6 OR H=5 AND O=3 AND E(39)=0
PRINT"I do not see a place to put it.":G
OTO 150
1130 IF O=3 AND E(39)=0 E(39)=A
1140 E(O)=A:GOTO 1030
1150 IF (O<>28 OR E(29)<>56) AND O<>10
AND O<>15 THEN 1020
1160 IF O=15 AND A=50 PRINT"I cannot do
that.":GOTO 150
1170 IF O=10 AND A<>35 OR E(O)=56 THEN
1090
1180 IF O=10 AND E(18)<>0 OR O=15 AND E
(17)<>0 PRINT"I have nothing to kill wit
h.":GOTO 150
1190 IF A=45 AND O=15 E(O)=48:GOTO 1030
1200 IF A=48 AND O=15 E(O)=50:GOTO 1030
1210 E(O)=56:E(29)=1:GOTO 1030
1220 IF O=27 AND A=6 PRINT"Escape befor
e Dracula kills you.":GOTO 150
1230 IF O=24 AND A=46 PRINT"An exit fr
om this castle lies behind a"CHR$130"loc
ked door.":GOTO 150
1240 GOTO 1020
1250 IF O=6 AND E(O)<>0 THEN 1040
1260 IF O=6 PRINT"It gives the formula:
- X*Y+Z-5=W/V":GOTO 150
1270 GOTO 1020
1280 IF O<>25 AND O<>26 THEN 1020
1290 IF A<>21 AND A<>40 THEN 1080
1300 IF A=40 PRINT"Wolves fear lit-can
dles, and laser-guns"CHR$130"can blast se
aled-doors.":GOTO 150
1310 A=29:PRINT"You have been thrown t
hrough a door in"CHR$130"the west wall."
:GOTO 150
1320 IF O<>31 THEN 1020
1330 IF A<>E(O) THEN 1080
1340 E(O)=56:GOTO 1030
1350 IF O<>35 AND O<>32 THEN 1020
1360 IF A<>8 AND A<>52 THEN 1080
1370 IF E(30)=0 AND A=8 PRINT"You have
fallen through a trapdoor.":GOTO 1010
1380 IF E(30)=0 AND E(33)<>56 PRINT"It
is already open.":GOTO 150
1390 IF E(30)=0 AND E(33)=56 E(33)=52:G
OTO 1030
1400 IF E(21)=0 AND A=8 A=53:GOTO 1030
1410 IF E(21)=0 PRINT"The key does not
fit.":GOTO 150
1420 PRINT"I have no key.":GOTO 150
1430 IF O<>22 THEN 1020
1440 IF E(O)<>0 THEN 1100
1450 PRINT"With what formula?":GOTO 150
1460 IF O<>12 AND O<>38 AND O<>41 THEN
1020
1470 IF O=41 AND A=30 R=RND(6):PRINT"It
is ";R;" o'clock.":GOTO 150
1480 IF E(O)<>0 THEN 1040
1490 IF O=12 PRINT"There is water in th
e bucket.":GOTO 150
1500 IF O=38 PRINT"You must escape with
in ";245-W;" moves.":GOTO 150

```

```

1510 IF O<>44 THEN 1020
1520 IF E(O)<>0 AND E(O)<>-1 THEN 1100
1530 IF E(O)=-1 PRINT"I am already wear
ing them.":GOTO 150
1540 E(O)=-1:GOTO 1030
1550 IF O<>12 THEN 1020
1560 IF E(O)<>0 THEN 1040
1570 IF A<>47 PRINT"I see no place to e
mpty it.":GOTO 150
1580 E(O)=56:PRINT" A passage has been
cleared through the"CHR$130"flames.Your
bucket has melted.":GOTO 150
1590 IF O<>45 THEN 1020
1600 IF A<>41 THEN 1080
1610 PRINT"You have switched off the li
ghts."
1620 PRINT CHR$130"Dracula has killed y
ou.":GOTO 1010
1630 IF O<>46 THEN 1020
1640 IFA<>17 OR E(46)=56 THEN 1090
1650 IF E(8)<>0 PRINT"I have nothing to
blast with.":GOTO 150
1660 E(46)=56:GOTO 1030
1670 IF O<>33 THEN 1020
1680 IF E(O)<>0 THEN 1040
1690 IF A<>53 PRINT"I see no place to a
ttach it to.":GOTO 150
1700 IF E(37)<>56 PRINT"Dracula will no
t let you.":GOTO 150
1710 S=1:PRINT"It is attached to a conv
enient post.":GOTO 150
1720 IF O<>33 THEN 1020
1730 IF A<>53 PRINT"I see no place to l
ower it to.":GOTO 150
1740 IF S<>1 PRINT"It has fallen down t
he hole.":E(O)=56:GOTO 150
1750 S=2:GOTO 1030
1760 IF O<>33 OR A<>53 THEN 1020
1770 IF S<>2 PRINT"You have fallen down
the hole.":GOTO 1010
1780 A=54:GOTO 150
1790 W=245-W:IF W>X X=W
1800 PRINT"Score=";W;" Best Score=";X
:GOTO 1010
1810 IF O<>47 THEN 1020
1820 IF E(19)<>0 PRINT"I have nothing t
o cut with.":GOTO 150
1830 IF A<>46 OR T=1 PRINT"I see nothin
g to cut.":GOTO 150
1840 T=1:GOTO 1030
1850 IF O<>5 THEN 1020
1860 IF A<.34 THEN 1080
1870 IF H=6 OR E(48)<>56 PRINT"It will
not move.":GOTO 150
1880 E(48)=34:GOTO 1030
1890 IF O<>16 THEN 1020
1900 IF E(O)<>0 PRINT"I have no stake."
:GOTO 150
1910 IF E(49)<>0 PRINT"I have no mallet
.":GOTO 150
1920 IF H<>6 AND A=34 AND E(48)=A PRINT
"The vampire is dead,but your mallet has
"CHR$130"disappeared.":E(49)=53:E(16)=34
:U=1:GOTO 150
1930 IF A=53 AND E(37)=A PRINT"You have
killed Dracula.":E(37)=56:GOTO 150
1940 GOTO 1020
1950 IF O<>48 THEN 1020
1960 IF E(50)<>0 PRINT"I have nothing t
o chop with.":GOTO 150

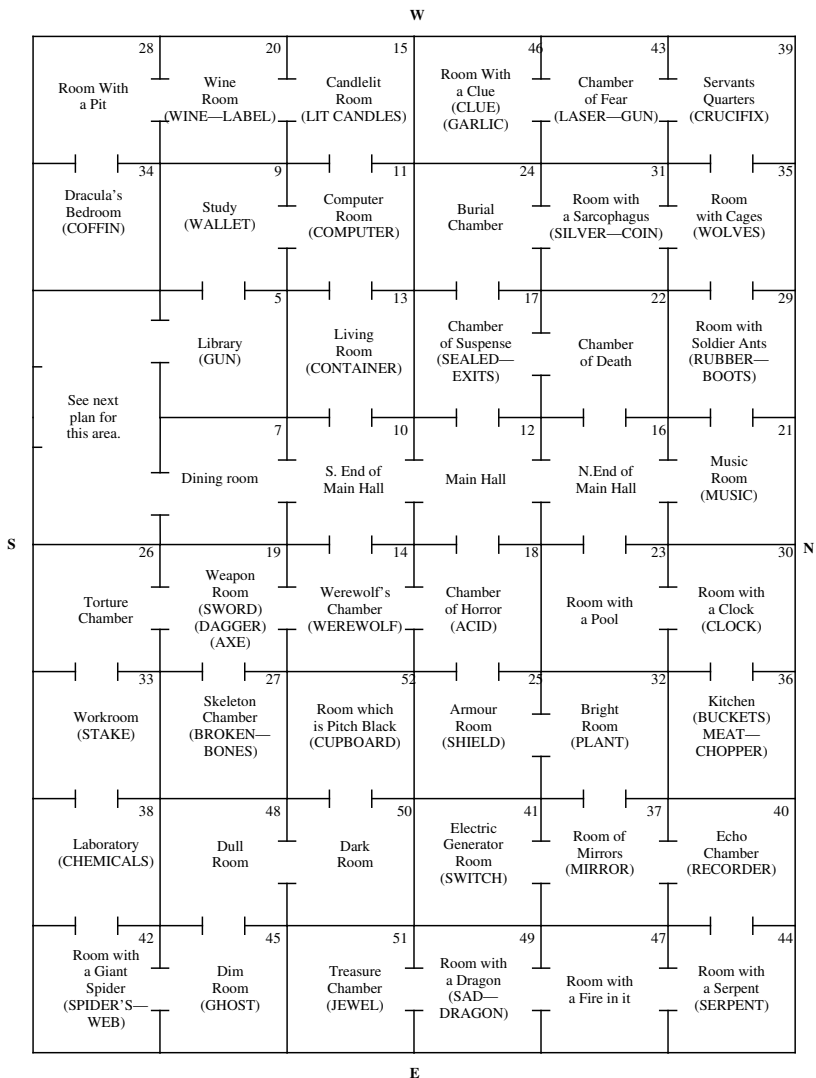
```

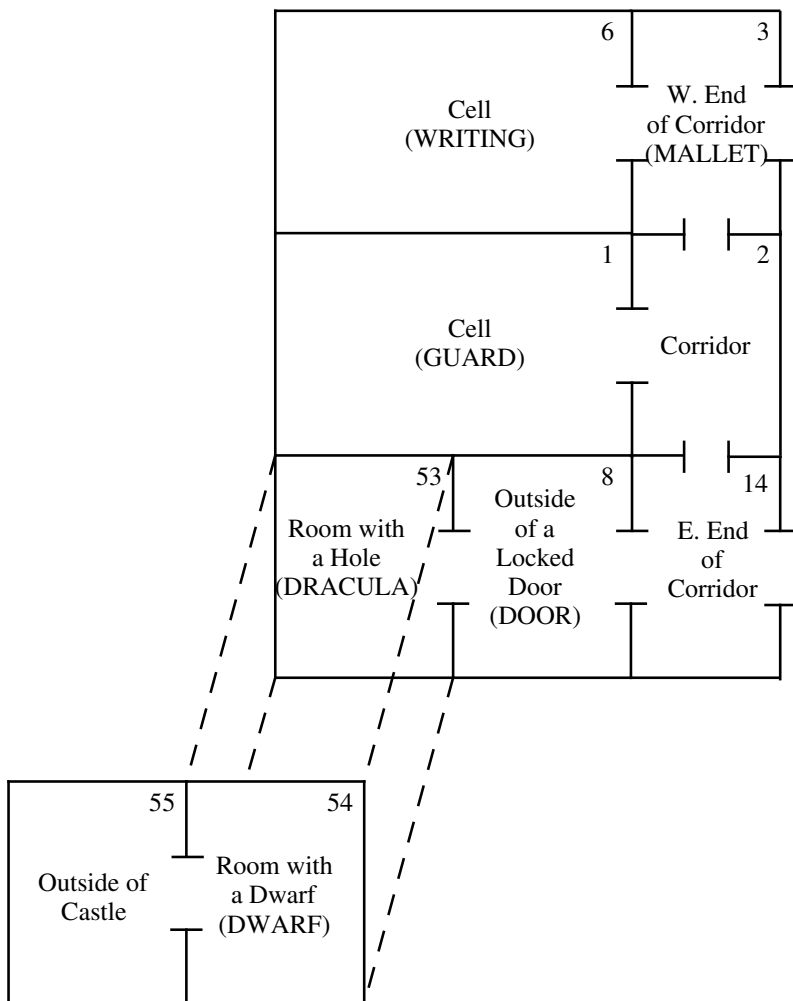


```

1 9 7 0   I F   A < > 3 4   O R   E ( O )   < > 3 4   T H E N   1 0 8 0
1 9 8 0   I F   U = 0   P R I N T " I t   i s   s t i l l   a l i v e . " : G
O T O   1 5 0
1 9 9 0   E ( 4 8 ) = 5 7 : G O T O   1 0 3 0
2 0 0 0   D A T A   5 , 9 , 1 3 , 1 5 , 3 4 , 2 0 , 3 9 , 4 3 , 3 1 , 3 5 , 1
1 , 3 6 , 2 5 , 5 1 , 4 5 , 3 3 , 1 9 , 1 9 , 1 9 , 2 7 , 5 6 , 3 8 , 4 2 , 4 6
, 2 1 , 4 0 , 6 , 1 , 5 6 , 5 6 , 3 2 , 5 2 , 5 6 , 5 4 , 8 , 4 9 , 5 3 , 3 7 ,
1 8 , 5 6 , 3 0 , 1 4 , 4 4 , 2 9 , 4 1 , 1 7 , 4 6 , 5 6 , 3 , 3 6

```





The plan for the adventure, “Dracula’s castle”, is on the two previous pages. In each room, the objects present at the start of the game are given in brackets. If you are wondering why the rooms fifty-three, fifty-four, and fifty-five do not correspond to the method of numbering rooms, this is because when I originally wrote the adventure, I had not fully considered these rooms, and so when I did think of them, they were tagged on at the end. The room numbers were not changed, since “A” is set equal to fifty-three on opening the locked door with the right key, instead of being added to a value in the variable “D”.

DOCUMENTATION

Since many of the variables used in this adventure have the same usages as in the model adventure, these variables will therefore have already been dealt with. Hence, only the variables “new” to this adventure will be explained below.

Numerical Variables

1) R — This variable is used to determine the time on a clock in line 1470:

```
1470      IF O=41 AND A=30 R=RND(6):PRINT“It is “;R;” o’clock.”:
          GOTO 150
```

If the player is in the “Room with a clock” and wishes to “EXAMINE” the “CLOCK”, the time on the clock, “R”, is given a random value between one and six. The time on this clock determines whether or not a “WEREWOLF” is asleep or awake, for if the time is before “4 o’clock”, and the player is in the “Werewolf’s chamber”, he or she will be killed:

```
350      IF A=14 and R<4 PRINT CHR$130“The werewolf has awoken
          and killed you.”:GOTO 1010
```

Note that in this adventure, the line to which the computer jumps to, for the player to press the space bar to restart the game, is line 1010, and not line 710, as in the model adventure.

2) S — The status of a rope is controlled by this variable. If the rope is tied to a post, then “S” will equal one, and if in addition to being tied, the rope is also lowered down a hole, then “S” will equal two. If the rope is lowered without being tied, then it will fall down the hole, and if a player enters “CLIMB ROPE” without it being tied or lowered, the player will fall down the hole. The player must, of course, be in the right room.

3) T — The value in “T” concerns whether a piece of “GARLIC” has been cut or not. If “T” equals zero, the “GARLIC” has still to be cut, and if “T” equals one, it has already been cut, and cannot be cut again. Once it has been cut, the adventurer can pick it up.

4) U — The state of health of a certain vampire is controlled by this

variable. If ‘U’ equals zero, then ‘It is still alive’. To kill it, the ‘STAKE’ must be hammered into it, while the ‘MALLETT’ is being carried, as is usual when killing vampires; to render it harmless, you have to attack it with the ‘MEAT-CHOPPER’ (‘CHOP VAMPIRE’) and ‘DROP’ the ‘GARLIC’ — the value in ‘U’ is incremented when the ‘STAKE’ is used, and the action concerning this must be carried out first.

5) W — This variable, as well as having the same use as in the model adventure, takes over from the variable ‘Y’, in the role of containing the score obtained in the adventure. This is done in line 1790:

```
1790      W=245—W:IF W>X X=W
```

Dimensional Variables

As in the model adventure, the object corresponding to each value of ‘E’ is given:

- E(1) —GUN
- E(2) —WALLET
- E(3) —CONTAINER
- E(4) —LIT-CANDLES
- E(5) —COFFIN
- E(6) —WINE-LABEL
- E(7) —CRUCIFIX
- E(8) —LASER-GUN
- E(9) —SILVER-COIN
- E(10) —WOLVES
- E(11) —COMPUTER
- E(12) —BUCKET
- E(13) —SHIELD
- E(14) —JEWEL
- E(15) —GHOST
- E(16) —STAKE
- E(17) —SWORD
- E(18) —DAGGER
- E(19) —AXE
- E(20) —BROKEN-BONES
- E(21) —SKELETON-KEY
- E(22) —CHEMICALS
- E(23) —SPIDER’S WEB
- E(24) —CLUE
- E(25) —MUSIC
- E(26) —RECORDER
- E(27) —WRITING
- E(28) —GUARD
- E(29) —DEAD-GUARD
- E(30) —KEY

E(31) —PLANT
E(32) —CUPBOARD
E(33) —ROPE
E(34) —DWARF
E(35) —DOOR
E(36) —SAD-DRAGON
E(37) —DRACULA
E(38) —MIRROR
E(39) —ACID
E(40) —GLUE
E(41) —CLOCK
E(42) —WEREWOLF
E(43) —SERPENT
E(44) —RUBBER-BOOTS
E(45) —SWITCH
E(46) —SEALED-EXITS
E(47) —GARLIC
E(48) —VAMPIRE
E(49) —MALLET
E(50) —MEAT-CHOPPER

String Variables

There is no difference between the string variables in this adventure and those in the model adventure, and so there is no need to analyse any of them a second time.

Line Number Analysis

Since I have already made a detailed analysis of an adventure, it is only necessary in this adventure to point out where the various routines are in the program:

- 1) *LINES 10–140* — Initialisation of the variables and the printing out of the instructions if they are asked for.
- 2) *LINE 150* —Resetting of the data pointer.
- 3) *LINES 160–170* —Sound for the adventure.
- 4) *LINES 190–260* — Bypassing of incrementing of the number of moves, when the player is in the last room.
- 5) *LINES 90–260* — Incrementing of the number of moves, and the printing out of the relative messages, depending on the number of moves made.
- 6) *LINES 270–460* — Various deaths which occur depending on the room in which the adventurer is in, and what is being carried. There are also various other things that may happen concerning the position of the player:

for example, if one is in the ‘Dining room’ without the ‘GUN’, then a robot will do some killing; on the other hand, if the ‘GUN’ is carried, then the robot drops a key:—

```
420    IF A=7 AND E(1)<>0 PRINT CHR$130"A robot has killed  
      you.":GOTO 1010  
430    IF A=7 AND E(30)=56 E(30)=7:PRINT CHR$130"A  
      robot,seeing your gun,dropped a key."
```

Line 430 does the killing, and line 440 sees that a key is dropped if the gun is possessed, and the key has not already been dropped.

7) *LINES 470–480* — The finding of the name of the room in which the adventurer is in, and printing it on the screen.

8) *LINES 490–520* — The printing out of the exits from the room in which the player is situated. These may be any configuration of ‘North’, ‘South’, ‘East’, and ‘West’.

9) *LINES 530–550* — Working out of the objects in the room in which the adventurer is in, and printing them on the display. In this adventure there are fifty objects, and hence the ‘G’ FOR/NEXT Loop starts with: FOR G = 1 TO 50.

10) *LINES 560–580* — Prints out the objects that are presently being carried by the adventurer . Note that six objects are allowed in this adventure, compared with four in the model adventure, because of the difference in size — the variable ‘H’ is for the objects in the room, and is checked for equalling six, and not four, in line 550 the variable ‘F’, for the objects carried, is checked in line 580.

11) *LINES 590–640* — Creation of the display for the input, as well as the inputting of the commands and objects into B\$. Line 610 checks for the player being in the last room, and removes the ‘SILVER-COIN’ from the scene as a result of this, as a payment to the dwarf for passing it. Line 620 checks for the player being in the ‘Computer room’ — the player is only allowed to enter a command if ‘Y’ is the reply to the question asked. Line 640 sets the display for the replies to the command and object.

12) *LINES 650–670* — Error trapping for commands which may otherwise be considered as requests for movement. Movements which are not allowed under the conditions are also trapped here.

13) *LINES 680–710* — Movement routines for movement ‘North’, ‘South’, ‘East’, and ‘West’, and the storage of the difference between room numbers in the variable ‘D’.

14) *LINES 720–750* — Additional confinement of movement by the player under certain conditions. Line 750 ends by adding ‘D’ to ‘A’ if the player is allowed to move freely.

- 15) *LINE 760* — If ‘D’ equals zero then there is no exit in the chosen direction, and the computer jumps to this line.
- 16) *LINES 770–800* — Data for movement in each of the four possible directions which the adventurer may attempt to move in.
- 17) *LINES 810–820* — Data for the names of all the fifty objects used in the adventure.
- 18) *LINE 830* — Data for the commands which the player may try using when playing the game.
- 19) *LINES 840–890* — Data for all the fifty-five room names, the layout of which may be seen in the adventure plan.
- 20) *LINE 900* — Definition of the three variables ‘M’, ‘N’, and ‘O’, along with the allowance for the adventurer to ‘QUIT’ if his/her situation is hopeless.
- 21) *LINES 910–920* — Working out of the command entered, and the storing of its number in ‘M’.
- 22) *LINE 930* — If the command is not recognised, then ‘I do not understand you’ is printed out.
- 23) *LINES 940–980* — Finding out of what the object to the command is, and storing its number in ‘O’. If whatever is entered is not recognised, then ‘Pardon?’ will be printed out. In line 96{ is a point which must be mentioned, but so that it is taken care of properly, it is dealt with in ‘Journey to freedom’, which has better examples of this point.
- 24) *LINE 990* — If a space has been missed out between the command and the object, then ‘N’ will equal one, and ‘Learn to type’ will be printed by this line.
- 25) *LINE 1000* — List of line numbers where the computer may jump to depending on the value of ‘M’.
- 26) *LINE 1010* — Line which deals with the player starting again.
- 27) *LINES 1020–1040* — Three quite commonly used messages may be printed by these lines depending on which line number is after ‘GOTO’.
- 28) *LINES 1050–1070* — Routine which corresponds to the command ‘GET’. Line 1050 is an example of something which may work alright in theory, but not always in practice, for the movable objects are not distinctly separated from those which may not be moved. This is because when one works on adventures, all the objects and what happens to them, have not been clearly thought of, and so the moveable objects are often mixed up with those which may not be moved — this means that each object which may not be moved has to be accounted for and be prevented from being picked up. This may be one of the more obvious things that does not work

quite as planned.

29) *LINES 1080–1100* — Three other commonly used messages are dealt with here.

30) *LINES 1110–1140* — The routine for dropping objects in rooms is dealt with in these lines.

31) *LINES 1150–1210* — This is the routine which allows the adventurer to try and kill things.

32) *LINES 1220–1270* —The routine for ‘READ’.

33) *LINES 1280–1310* —The routine for ‘PLAY’.

34) *LINES 1320–1340* —This concerns the command ‘KICK’.

35) *LINES 1350–1420* —‘OPEN’ is dealt with here.

36) *LINES 1430–1450* — ‘MIX’ is the command that these lines deal with.

37) *LINES 1460–1500* —Routine for examining objects.

38) *LINES 1510–1540* — Lines which enable the player to ‘WEAR’ something.

39) *LINES 1550–1580* —The routine for the command ‘EMPTY’.

40) *LINES 1590–1620* — This allows the player to ‘PRESS’ something.

41) *LINES 1630–1660* —The command dealt with here is ‘BLAST’.

42) *LINES 1670–1710* —These lines are for the command ‘TIE’.

43) *LINES 1720–1750* —‘LOWER’ is dealt with within these lines.

44) *LINES 1760–1800* — The routine which allows the player to ‘CLIMB’.

45) *LINES 1810–1840* —Enables the player to ‘CUT’ something.

46) *LINES 1850–1880* — The routine for lifting something (not the same as ‘GET’).

47) *LINES 1890–1940* — These lines concern the hammering of something.

48) *LINES 1950–1990* — The routine for the last command, which is ‘CHOP’.

49) *LINE 2000* — This line contains the data for the status of the objects at the start of the game.

Since me computer memory is just about completely filled, if its limit is 16K, there is insufficient memory to comfortably add alternative commands.

Although there is no problem with a 32K BBC Microcomputer, on the 16K version, to make the program run, it may be necessary to miss out unnecessary spaces. For example, line 400 would be changed from:

```
400      IF A=35 AND E(10)=A PRINT CHR$130"Your lit-candles  
        frighten the wolves."
```

to:

```
400      IFA=35ANDE(10)=A PRINTCHR$130"Your lit-candles  
        frighten the wolves."
```

Note that the space is kept in between ‘E(10)=A’ and ‘PRINT . . .’, since the computer would otherwise think that ‘APRINT . . .’ is a variable, and come up with the error message ‘No such variable’. Repeated over a number of lines, this will save sufficient memory for the program to run on a 16K BBC Micro.

If you have another micro and you find that there is not enough memory left within your 16K (assuming that your micro has 16K), then you will probably require to delete pieces of information. The instructions are optional although preferable, but they may be deleted to save memory — they are not essential for the program to run smoothly; likewise, the sound routine can also be deleted. If there is still insufficient memory, then the ‘PRINT’ statements should be pruned to the bare essentials. If further memory saving is required then the user will have to study the listing and the documentation, and delete routines, which, to them, seem the least significant to the running of the program, until it fits within the memory. For instance, in my original version of this program for the ZX80, I did not include the necessity to kill the vampire in the coffin.

When playing this larger adventure, you may notice that there is a difference in speed. This is because, with the increase in size, the computer has more things to check, and hence will take longer to come to its decisions.

Although it is preferable to have the listing and structure of the adventure as neat as possible, it often happens that complications arise, and the neatness suffers: this may occur through the laziness of the programmer — a great deal of time may be required to iron out any problems in structuring an adventure, since when actually writing a program, it is hard to keep to the model version for that program; if your micro does not have the facility for renumbering the line numbers, then structural difficulties may also arise. However, the main objective when writing a program, is to make it look good on the screen, and the main benefits arising from a good

structure are a slight increase in speed, and a saving in memory. In short, try and work as closely as possible to the structure of the model adventure, and if any difficulties arise, like, for example, having the moveable objects mixed with those objects which may not be moved — this is not a pleasant thing to sort out in a program — then attempt to fix it so that any player would not notice any obvious difference when playing the game.

I have included the adventure ‘Dracula’s castle’ in the book, as it is the first one that I have written, and I have improved upon the standard to which it was originally written. My first adventures were basically written around a six by eight matrix of rooms, but in my more recent ones, I try to devise more irregular matrices.

CHAPTER 6

JOURNEY

```
10 DIM E(55):X=0
20 RESTORE 770:FOR B=1 TO 55:READ E(B)
):NEXT B
30 CLS:PRINT CHR$129"Do you want the
instructions(Y or N) ?";:Z$=GET$:IF Z$="
N" THEN 120 ELSE IF Z$="Y" THEN 40 ELSE
30
40 CLS:PRINT''CHR$130"You have escap
ed from the castle with a"CHR$130"valuab
le jewel which you keep hidden."
50 TIME=0:REPEAT UNTIL TIME>400
60 PRINT''CHR$131"You must escape as
quickly as possible"CHR$131"from the ar
ea around the castle,as your"CHR$131"tim
e is limited before Dracula's guards"CHR
$131"find you and then kill you."
70 TIME=0:REPEAT UNTIL TIME>500
80 PRINT''CHR$132"The computer has a
fairly large number"CHR$132"of commands
,so therefore if one command"CHR$132"doe
s not work then try another."
90 TIME=0:REPEAT UNTIL TIME>500
100 PRINT''CHR$133"The first three let
ters of each command"CHR$133"and object
need be typed in,although,if"CHR$133"des
ired,the full word may be entered."
110 TIME=0:REPEAT UNTIL TIME>500
120 CLS:A=1:R=0:S=0:T=0:W=0:SOUND &11,
0,0,0
130 RESTORE
140 ENVELOPE 1,1,-1,1,-1,0,15,30,0,0,0
,0,W/3,0
150 SOUND 1,1,W,1
160 W=W+1:IF POS=0 PRINT CHR$130;
170 IF W>34 AND W<70 PRINT"Guards have
discovered Dracula's death."
180 IF W>69 AND W<105 PRINT"They have
now been alerted to find you."
190 IF W>104 AND W<140 PRINT"Your pres
ence has been detected."
200 IF W>139 AND W<175 PRINT"You have
been noticed."
210 IF W>174 AND W<210 PRINT"The guard
s want to kill you."
220 IF W>209 AND W<245 PRINT"They are
closing round you."
230 IF W=245 PRINT"Dracula's guards ha
ve killed you.":GOTO 890
240 IF POS=0 PRINT CHR$130;
250 IF A=26 AND E(53)<>0 PRINT"You hav
e fallen down into a hole in the"CHR$130
"dark.":GOTO 890
260 IF A=42 AND E(31)<>-1 PRINT"The ai
r is too thin for you to breathe.":GOTO
890
270 IF A=8 AND E(20)=A AND E(15)=A E(1
5)=50:E(16)=A:PRINT"The anteater has eat
en the termites,and"CHR$130"has revealed
a harpoon."
280 IF E(31)<>50 AND S=1 AND A=43 PRIN
```

```

T" Your air-cylinder has exploded in you
r"CHR$130"face.":GOTO 890
290 IF A=35 AND T=1 PRINT"You have bee
n stung by the scorpion.":GOTO 890
300 IF A=43 S=1
310 IF A=35 AND E(41)<>50 THEN T=1
320 IF E(51)=0 PRINT"You have sunk int
o the sinking sand.":GOTO 890
330 IF E(44)<>0 AND A=36 PRINT"The cob
ra has killed you.":GOTO 890:ELSE IF A=3
6 PRINT"The cobra keeps away from your g
arlic."
340 IF A=13 AND E(6)<>-1 PRINT"You hav
e sunk in the swamp.":GOTO 890
350 IF E(40)=0 AND E(39)<>51 PRINT"Dra
cula has killed you.":GOTO 890
360 IF E(47)<>-1 AND A=32 PRINT"The mi
notaur has killed you.":GOTO 890
370 IF A=21 AND E(26)<>50 PRINT"You ha
ve fallen down a pit.":GOTO 890
380 IF A=30 AND E(34)<>50 OR A=29 AND
E(34)=50 PRINT"You have drowned in the r
iver.":GOTO 890
390 IF A=11 AND E(19)<>-1 PRINT"A guar
d has killed you.":GOTO 890
400 IF POS=0 PRINT CHR$130;
410 FOR B=1 TO A:READ A$:NEXT B
420 VDU 31,0,3:PRINT CHR$130;A$
430 PRINT'CHR$131"Exits:- ";:RESTORE 7
00:FOR C=1 TO A:READ D:NEXT C:IF D<>0 PR
INT":North:";
440 RESTORE 710:FOR C=1 TO A:READ D:NE
XT C:IF D<>0 PRINT":South:";
450 RESTORE 720:FOR C=1 TO A:READ D:NE
XT C:IF D<>0 PRINT":East:";
460 RESTORE 730:FOR C=1 TO A:READ D:NE
XT C:IF D<>0 PRINT":West:";
470 PRINT'CHR$132"Objects:- ";
480 H=0:RESTORE 740
490 FOR G=1 TO 55:READ C$:IF E(G)<>A O
R H=6 NEXT G ELSE PRINT":";C$;":":H=H+1
:IF H<>2 AND H<>4 NEXT G ELSE PRINT'CHR$
132" ";:NEXT G
500 PRINT'CHR$133"Inventory:- ";
510 F=0:RESTORE 740
520 FOR G=1 TO 55:READ C$:IF E(G)<>0 A
ND E(G)<>-1 OR F=6 NEXT G ELSE PRINT":";
C$;":":F=F+1:IF F<>2 AND F<>4 NEXT G EL
SE PRINT'CHR$133" ";:NEXT G
530 VDU 31,0,15,134:PRINT"[-----
-----]"
540 VDU 31,0,19,134:PRINT"[-----
-----]":VDU 31,0,17
,135
550 IF A=49 THEN 2020
560 INPUT"Command? "B$
570 CLS:VDU 31,0,21,130
580 IF LEFT$(B$,3)="EXA" OR LEFT$(B$,3
)="WEA" OR LEFT$(B$,3)="EAT" OR LEFT$(B$
,3)="SHO" OR LEFT$(B$,3)="SPE" OR LEFT$(
B$,3)="STA" THEN 780
590 IF LEFT$(B$,1)<>"N" THEN 600 ELSE
RESTORE 700:FOR C=1 TO A:READ D:NEXT C:I
F D=0 THEN 640 ELSE 630
600 IF LEFT$(B$,1)<>"S" THEN 610 ELSE
RESTORE 710:FOR C=1 TO A:READ D:NEXT C:I
F D=0 THEN 640 ELSE 630
610 IF LEFT$(B$,1)<>"E" THEN 620 ELSE
RESTORE 720:FOR C=1 TO A:READ D:NEXT C:I
F D=0 THEN 640 ELSE 630

```

```

620 IF LEFT$(B$,1)<>"W" THEN 780 ELSE
RESTORE 730:FOR C=1 TO A:READ D:NEXT C:IF
D=0 THEN 640 ELSE 630
630 A=A+D:GOTO 130
640 PRINT"No exit!":GOTO 130
650 DATA Outside of castle,Part of a p
ath,Entrance to outhouse,Woodcutter's hu
t,Cloakroom,Path beside a river,W.End of
corridor,Glade,Corridor,Bank of river,G
uards' quarters,E.End of corridor
660 DATA Swamp in forest,Fuel room,Foo
d room,Dense part of forest,Money room,S
tore room,Cleared area of forest,Supply
room,Lion pits,Large tree in forest,Cair
n in forest,Thin forestry,Clearing
670 DATA Mouth of a cave,Bear's cave,D
imly lit part of cave,Exit from cave,Ent
rance to a maze,Part of a maze with a le
ver,Minotaur's section of maze,Centre of
maze
680 DATA Thick undergrowth,Ditch in ma
ze,Area guarded by a cobra,Thick bushes
in maze,Exit from maze,Burnt area,Rubbis
h dump,Steep sides of a mountain
690 DATA Pass in a mountainous region,
Large valley,Sides of a lake,Rocky part
of a route,Sinking sand,Old windswept ro
ad,Outside of city gates,Home city
700 DATA 0,0,-1,0,0,-2,0,-2,0,-2,-2,3,
3,0,0,3,-3,0,2,0,0,0,0,-2,-1,-1,-1,1,0,0
,-1,1,1,0,1,1,0,1,0,0,0,-1,0,-1,0,0,1,0,
0
710 DATA 0,1,0,2,0,0,0,2,2,0,0,0,0,3,-
3,-3,0,0,-3,0,-2,2,0,1,0,1,0,0,-1,1,0,0,
-1,-1,0,-1,-1,0,-1,0,1,0,1,0,0,0,-1,0
720 DATA 0,-1,2,-2,2,0,2,0,3,0,0,0,-3,
0,0,0,0,-3,0,-2,0,-3,-2,0,0,0,1,0,1,0,1,
0,2,2,0,2,2,3,0,-3,0,0,-1,1,2,-2,0,1,0
730 DATA 1,2,0,0,-2,0,-2,0,-2,0,0,-3,0
,-3,3,0,0,2,3,0,2,0,0,0,0,0,0,-1,0,-1,0,
-1,0,0,-2,-2,-3,-2,-2,0,-3,1,0,2,-1,0,-2
,0,-1
740 DATA PARCHMENT,POST,WALL,MATCHES,C
HAIN-SAW,BOOTS,DOG,TERMITES,HARPOON,GLAS
SES,GUARD,TAPER,CANS,FOOD,TREES,SPADE,CH
EST,DUSTER,BOXING-GLOVE,ANTEATER,FUEL-TA
NK,OIL,LARGE-TREE,CUBE,RUSTY-KEY,LION,DO
OR,CORD,BEAR,SIGN,AIR-CYLINDER
750 DATA CRUCIBLE,LEVER,MINOTAUR,FOUNT
AIN,BULLET,UNDERGROWTH,GRAVE,DRACULA,CRO
SS,SCORPION,COBRA,BUSHES,GARLIC,SHOTGUN,
SPEAR,DIAMOND,SNOW,WHALE,WEREWOLF,SAND,G
ATES,LIT-TAPER,KEY,FLICK-KNIFE
760 DATA GET,DRO,WEA,EAT,REA,LIG,PAT,E
XA,TUR,FEE,DUS,OPE,PUN,PUL,FIL,CUT,DIG,H
IT,THR,CLI,MEL,KIL,SHO,SPE,STA
770 DATA 1,2,3,3,4,5,6,8,50,10,11,11,1
4,15,16,50,17,18,18,19,20,20,22,50,23,24
,25,25,27,27,28,30,31,32,33,50,34,50,50,
50,35,36,37,50,38,39,40,41,44,45,46,48,5
0,50,50
780 M=0:N=0:O=0:IF LEFT$(B$,3)="QUI" T
HEN 690
790 RESTORE 760:FOR I=1 TO 25:READ C$:
IF LEFT$(B$,3)=C$ M=I
800 NEXT I:IF M<>0 THEN 820
810 PRINT"I do not understand you.":GO
TO 130
820 RESTORE 740:D$=MID$(B$,3,4):FOR J=
1 TO 55:READ C$:C$=LEFT$(C$,3)

```

```

830 FOR K=4 TO 10:IF LEFT$(D$,1)<>" "
AND C$=MID$(D$,2,4) N=1
840 IF J=8 AND D$="STER" OR J=12 AND D
$="-TAP" OR J=28 AND D$="SCOR" K=10:N=0:
GOTO 860
850 IF C$=MID$(D$,2,3) O=J:K=10:J=55 E
LSE D$=MID$(B$,K,4)
860 NEXT K:NEXT J:IF O<>0 THEN 870 ELS
E PRINT"Pardon?":GOTO 130
870 IF N=1 PRINT"Learn to type."CHR$1
30;
880 ON M GOTO 940,990,1040,1080,1130,1
180,1220,1260,1300,1320,1360,1400,1450,1
480,1510,1570,1620,1680,1720,1780,1810,1
860,1890,1940,1980
890 VDU 23;11,0;0;0;0,31,6,10:PRINT"Pr
ess space to start again":IF INKEY$(50)=
" " VDU 23;11,255;0;0;0:GOTO 20 ELSE VDU
31,6,10:PRINT"
":IF INKEY$(50)=" " VDU 23;11,255;0;0;
0:GOTO 20 ELSE 890
900 PRINT"I cannot do that.":GOTO 130
910 PRINT"O.K.":GOTO 130
920 PRINT"I am not carrying it.":GOTO
130
930 PRINT"I do not see a place to put
it.":GOTO 130
940 IF O=3 OR O=7 OR O=8 OR O=11 OR O=
15 OR O=21 OR O=23 OR O=26 OR O=27 OR O=
29 OR O=30 AND E(29)=27 OR O=32 OR O=33
OR O=34 OR O=35 OR O=37 OR O=38 OR O=39
OR O=41 OR O=42 OR O=43 OR O=49 OR O=50
OR O=52 OR O=2 AND R=0 THEN 900
950 IF F=6 PRINT"I am carrying too muc
h.":GOTO 130
960 IF E(O)=A THEN 980
970 PRINT"I do not see it here.":GOTO
130
980 E(O)=0:GOTO 910
990 IF E(O)<>0 AND E(O)<>-1 THEN 920
1000 IF H=6 THEN 930
1010 IF O=22 OR O=13 AND E(22)=0 AND H=
5 OR O=5 AND E(22)=-1 AND H=5 OR A=20 AN
D H=5 THEN 900
1020 IF O=13 AND E(22)=0 OR O=5 AND E(2
2)=-1 E(22)=20
1030 E(O)=A:GOTO 910
1040 IF O<>6 AND O<>10 AND O<>19 AND O<
>31 THEN 900
1050 IF E(O)<>0 AND E(O)<>-1 THEN 920
1060 IF E(O)=-1 PRINT"I am already wear
ing it.":GOTO 130
1070 E(O)=-1:GOTO 910
1080 IF O<>14 AND O<>41 THEN 900
1090 IF O=41 AND A<>35 THEN 970
1100 IF O=14 AND E(O)<>0 THEN 920
1110 IF O=41 T=0
1120 E(O)=50:PRINT"Burp...that was good
.":GOTO 130
1130 IF O<>1 AND O<>30 THEN 900
1140 IF E(O)<>0 THEN 920
1150 IF O=1 PRINT" You must escape quic
kly before you are"CHR$130"caught by Dra
cula's guards.":GOTO 130
1160 IF E(10)<>-1 PRINT"It is necessary
for glasses to be worn "CHR$130"before
this sign may be read.":GOTO 130
1170 PRINT"It is dangerous to cross the
stream at "CHR$130"a point next to the
exit from the cave."

```

```

1180 IF O<>12 THEN 900
1190 IF E(O)<>0 THEN 920
1200 IF E(4)<>0 PRINT"I do not have any
matches.":GOTO 130
1210 E(O)=50:E(53)=0:GOTO 910
1220 IF O<>7 THEN 900
1230 IF A<>6 THEN 970
1240 PRINT"The dog lets you past it.":A
=8:GOTO 130
1250 PRINT"See what has been revealed."
:GOTO 130
1260 IF O<>3 AND O<>23 AND O<>35 AND O<
>37 AND O<>43 AND O<>52 THEN 900
1270 IF E(O)<>A THEN 970
1280 IF O=3 OR O=52 PRINT"This could be
climbed with some cord.":GOTO 130
1290 E(O)=50:E(O+1)=A:GOTO 1250
1300 IF O<>24 THEN 900
1310 IF E(O)<>0 THEN 920 ELSE PRINT"It
says:- (C) COPYRIGHT, IAN R. WATT":GOTO 1
30
1320 IF O<>26 THEN 900
1330 IF E(O)<>A THEN 970
1340 IF E(14)<>0 PRINT"I do not have an
y food.":GOTO 130
1350 E(O)=50:E(14)=50:PRINT" The lion d
isappeared after eating your"CHR$130"foo
d.":GOTO 130
1360 IF O<>25 AND O<>47 THEN 900
1370 IF E(O)<>0 THEN 920
1380 IF O=25 PRINT"The rust has been du
sted off your key.":E(O)=50:E(54)=0:GOTO
130
1390 PRINT"Your diamond is now a lethal
weapon.":E(O)=-1:GOTO 130
1400 IF O<>27 THEN 900
1410 IF A<>25 THEN 970
1420 IF E(25)=0 PRINT"The key does not
fit.":GOTO 130
1430 IF E(54)<>0 PRINT"I do not have a
key.":GOTO 130
1440 PRINT"You have passed through the
door.":A=26:GOTO 130
1450 IF O<>11 THEN 900
1460 IF A<>11 THEN 970
1470 PRINT"You have been thrown by the
guard into "CHR$130"an adjacent room.":A
=14:GOTO 130
1480 IF O<>33 THEN 900
1490 IF A<>31 THEN 970
1500 PRINT"You have pulled a lever, and
by doing so"CHR$130"have alerted Dracula
's guards.":GOTO 890
1510 IF O<>5 AND (O<>13 OR F=6) THEN 90
0
1520 IF E(O)<>0 THEN 920
1530 IF O=13 AND E(22)<>A THEN 970
1540 IF O=13 E(22)=0:GOTO 910
1550 IF E(22)<>0 PRINT"I have nothing t
o fill it with.":GOTO 130
1560 E(22)=-1:GOTO 910
1570 IF O<>15 THEN 900
1580 IF E(O)<>A THEN 970
1590 IF E(5)<>0 PRINT"I have nothing to
cut with.":GOTO 130
1600 IF E(22)<>-1 PRINT"I need some oil
in the chain-saw.":GOTO 130
1610 E(22)=20:E(O)=50:E(16)=A:GOTO 1250
1620 IF O<>2 AND (O<>38 AND H<>6) THEN
900

```

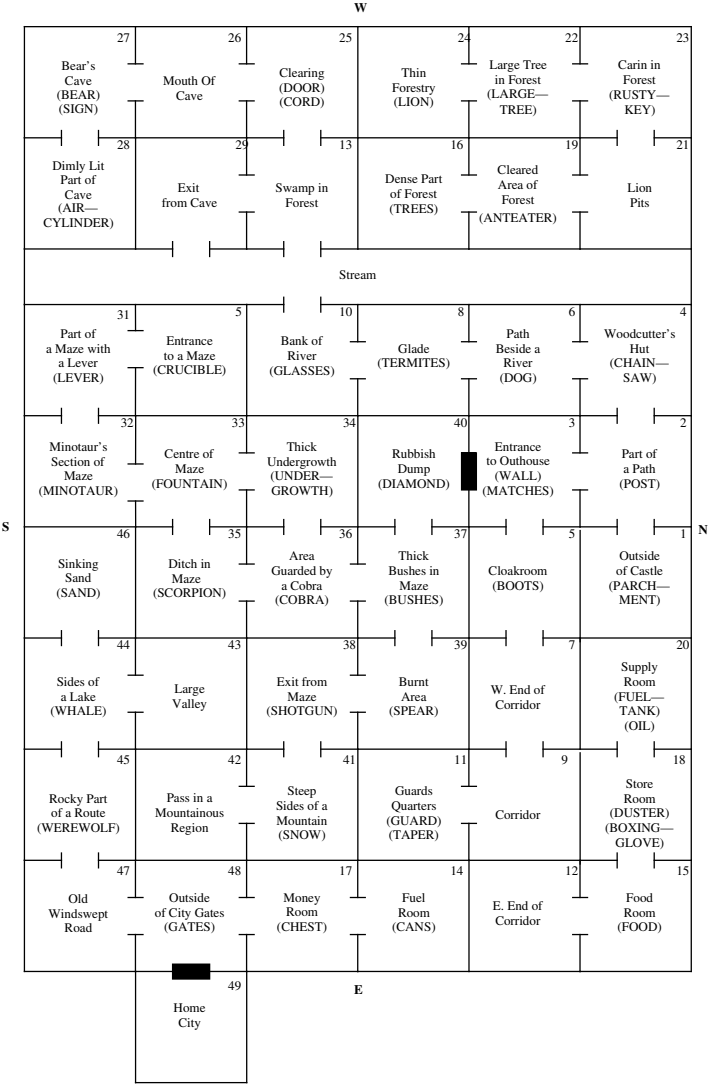
```

1630 IF E(O)<>A THEN 970
1640 IF E(16)<>0 PRINT"I have nothing t
o dig with.":GOTO 130
1650 IF O=38 E(39)=A:E(40)=A:PRINT"Drac
ula is here in an advanced state of"CHR$
130"reincarnation." 'CHR$130;:GOTO 1250
1660 IF R=1 PRINT"I have already dug it
out.":GOTO 130
1670 R=1:GOTO 910
1680 IF O<>29 THEN 900
1690 IF E(O)<>A THEN 970
1700 IF E(2)<>0 PRINT"The bear kills yo
u.":GOTO 890
1710 E(O)=50:PRINT"The bear has run awa
y.":GOTO 130
1720 IF O<>9 AND O<>17 AND O<>28 AND O<
>31 OR O=9 AND A<>44 OR O=17 AND A<>10 O
R O=28 AND A<>3 AND A<>48 OR O=31 AND A<
>43 THEN 900
1730 IF E(O)<>0 AND E(O)<>-1 THEN 920
1740 IF O=9 PRINT"You have been assassi
nated by the 'save"CHR$130"the whale' or
ganisation.":GOTO 890
1750 IF O=17 PRINT"The chest acts as a
platform across the"CHR$130"river.":E(17
)=10:A=13:GOTO 130
1760 IF O=28 PRINT"It has caught onto s
omething.":E(O)=-1:GOTO 130
1770 PRINT"It has exploded in mid air."
:E(O)=50:GOTO 130
1780 IF O<>28 OR A<>3 AND A<>48 THEN 90
0
1790 IF E(O)<>-1 PRINT"It is not attach
ed to anything.":GOTO 130
1800 IF A=3 A=40:E(O)=0:GOTO 130:ELSE A
=49:E(O)=0:GOTO 130
1810 IF A<>30 OR O<>40 AND O<>48 THEN 9
00
1820 IF E(O)<>0 THEN 920
1830 IF O=40 AND E(48)<>50 PRINT"The cr
ucible is not clean.":GOTO 130
1840 IF O=48 PRINT"The crucible is now
clean.":E(O)=50:GOTO 130
1850 PRINT"A flick-knife has been forme
d.":E(O)=51:E(55)=0:GOTO 130
1860 IF O<>34 THEN 900
1870 IF E(O)<>A THEN 970
1880 PRINT"The force resulting from the
minotaur's"CHR$130"destruction has proj
ected you into the "CHR$130"next room.":
A=31:E(O)=50:GOTO 130
1890 IF O<>39 THEN 900
1900 IF E(O)<>A THEN 970
1910 IF E(45)<>0 PRINT"I have no shotgu
n.":GOTO 130
1920 IF E(36)<>0 PRINT"I have no bullet
.":GOTO 130
1930 PRINT"Dracula is really dead now."
:E(O)=51:E(36)=51:E(45)=51:GOTO 130
1940 IF O<>50 OR A<>45 THEN 900
1950 IF E(O)<>A THEN 970
1960 IF E(46)<>0 PRINT"I do not have a
pear.":GOTO 130
1970 E(50)=47:PRINT"It has run away wou
nded.":GOTO 130
1980 IF O<>50 OR A<>47 THEN 900
1990 IF E(O)<>A THEN 970
2000 IF E(55)<>0 PRINT"I do not have a
flick-knife.":GOTO 130
2010 E(50)=50:PRINT" It has shrivelled

```



```
up,disappearing into "CHR$130"the ground.
":GOTO 130
2020 W=245-W:IF W>X X=W
2030 VDU 31,0,17,130:PRINT "Score=";W;"
      Best Score=";X:GOTO 890
```



In the plan on the previous page, the room number is given in the top right hand corner, and the room name is given at the top of the room square. Any objects in a room at the beginning of the game are in brackets below the room name. A stream separates the western side of the plan from the larger eastern side, but does not act as a room — it is fatal to fall into the stream. At the two instances where —■■■■— is encountered, the two rooms on either side are separated by an obstacle which must be climbed with the cord.

DOCUMENTATION

A list of all the variables — along with comments beside them — that have not been previously encountered, are listed below:

Numerical Variables

1) R — This variable concerns whether a post has, or has not been dug out of the ground. If “R” equals zero then the post is still in the ground, and therefore cannot be picked up. On the other hand, if “R” equals one, then the post will no longer be in the ground. The adventurer would then be able to “GET” it.

2) S — The state of stability of an air-cylinder is controlled by this variable. When “S” equals zero there is no chance of the air-cylinder exploding, but “S” is set equal to one when the player enters the “Large valley”, and it must be thrown away immediately, or else it will explode in the player’s face. It is necessary to carry this object at this stage since the adventurer would otherwise die in the thin air in the “Pass in a mountainous region”, which is encountered prior to entering the “Large valley”. The logic behind the air cylinder exploding, is that the drop in altitude makes it unstable (it is pressurised).

3) T — If this variable equals one, then a scorpion is in a position to kill the adventurer the next time a move is made, if this move is the incorrect one. “T” will equal zero when it has been removed from the game, or if it has not yet been encountered. When it is encountered, “T” is set equal to one, and the one way to render it harmless, is to “EAT” it.

4) POS — This variable contains the number of the column which the cursor is in.

Dimensioned Variables

As usual, the objects corresponding to each value in “E” are different from those already encountered, so the list for this adventure is given below:

- E(1) —PARCHMENT
- E(2) —POST

E(3) —WALL
E(4) —MATCHES
E(5) —CHAIN-SAW
E(6) —BOOTS
E(7) —DOG
E(8) —TERMITES
E(9) —HARPOON
E(10) —GLASSES
E(11) —GUARD
E(12) —TAPER
E(13) —CANS
E(14) —FOOD
E(15) —TREES
E(16) —SPADE
E(17) —CHEST
E(18) —DUSTER
E(19) —BOXING-GLOVE
E(20) —ANTEATER
E(21) —FUEL-TANK
E(22) —OIL
E(23) —LARGE-TREE
E(24) —CUBE
E(25) —RUSTY-KEY
E(26) —LION
E(27) —DOOR
E(28) —CORD
E(29) —BEAR
E(39) —SIGN
E(31) —AIR-CYLINDER
E(32) —CRUCIBLE
E(33) —LEVER
E(34) —MINOTAUR
E(35) —FOUNTAIN
E(36) —BULLET
E(37) —UNDERGROWTH
E(38) —GRAVE
E(39) —DRACULA
E(40) —CROSS
E(41) —SCORPION
E(42) —COBRA
E(43) —BUSHES
E(44) —GARLIC
E(45) —SHOTGUN
E(46) —SPEAR
E(47) —DIAMOND
E(48) —SNOW

E(49) —WHALE
E(50) —WEREWOLF
E(51) —SAND
E(52) —GATES
E(53) —LIT-TAPER

String Variables

As in ‘Dracula’s castle’, there are no additions to the list of string variables from the model adventure. This points towards a fixed list, which will apply to any adventure which you may try and write.

Line Number Analysis

I feel that there is again a need to analyse line numbers in this adventure, so that the structure of a third adventure may be compared with two others, and thus allow a greater understanding for other adventures to be written from the information within the chapters of this book. The line numbers or groups of line numbers, are given below with comments alongside them:

1) *LINES 10–120* — Initialisation of the numerical variables, and the values in the dimensioned variable of ‘E’. The option for instructions is given, and the screen is cleared in line 120, ready for the display format.

2) *LINE 130* —Resetting of the data pointer.

3) *LINES 140–150* — Lines which develop the sound and produce it. The sound is updated every move in both pitch and volume, with the changing value of the variable ‘W’.

4) *LINE 160* — Incrementing of the variable ‘W’, an checking to see if the cursor is in the first position on the line, and if so, the character concerning the colour is printed at this location.

5) *LINES 170–230* — Checking for various ranges which ‘W’ may be within, and printing of the appropriate message which corresponds to the value in ‘W’. Line 230 checks for ‘W’ reaching its maximum value, and hence the death of the player.

6) *LINE 240* — Concerns the setting of the text colour, depending on the position of the cursor in columns along the screen.

7) *LINES 250–390* — Various deaths which may occur, depending on what is being carried by the adventurer, and the situation encountered by him/her. The messages printed on the screen are not always to do with the player’s death, and may be messages resulting from escaping death, or from dropping something in a room and revealing something else. Variables are also defined through the above criterion.

8) *LINE 400* —This is to do with the text colour.

- 9) *LINES 410–420* — Extracting of the room name of the room in which the adventurer is situated, from the list of room names in the data lines. The room name is then printed out on the screen in the chosen colour.
- 10) *LINES 430–460* — Printing out on the screen of the exits from the room. As usual, there is the provision for any configuration of “North”, “South”, “East”, and “West”.
- 11) *LINES 470–490* — Lines for the screen format of the objects in the room. Remember that in an adventure of this size, six objects per room, and six objects carried at any one time are allowed.
- 12) *LINES 500–520* — Lines for the “Inventory” of what the player is carrying. Like the lines for the “Objects”, the data is read from the list of objects in the adventure.
- 13) *LINES 530–560* — Setting of the display for the player’s input. Line 550 checks to see if the player is in the last room and therefore the adventure is completed. The computer would then jump to line 2020 for the score obtained by the adventurer in that particular game. Line 560 asks for the input.
- 14) *LINE 570* — Clearing of the screen and setting of the display for the response to the entered command.
- 15) *LINE 580* — Checking for those commands which start with any of the letters which would produce a movement in one of the four ‘directions.
- 16) *LINES 590–640* — Checking for a request for movement in any direction, and seeing if this is possible — if this is so, then the value that must be added to “A” to land in the required room, is stored in “D”. “D” would then be added to “A” in line 630. On the other hand, if the desired movement is not possible, then “No exit” would be printed out in line 640.
- 17) *LINES 650–690* — Lines of data which correspond to the names of all the forty-nine rooms in the adventure.
- 18) *LINES 710–730* — Lines of data for the movement in the four directions which may chosen.
- 19) *LINES 740–750* — Data for all the fifty-five object names in the adventure.
- 20) *LINE 760* — Data for the first three letters of each of the commands which may be used while playing the game.
- 21) *LINE 770* — Data for the positions of all the objects at the beginning of each game.
- 22) *LINE 780* — Defining of the variables “M”, “N”, and “O”. A check is made to see if the player wishes to “QUIT” the present game.
- 23) *LINES 790–800* — Finding of the number corresponding to the

command entered, and storing of this number in the variable ‘M’.

24) *LINE 810* — Printing out of a message if the command entered is not within the computer’s vocabulary.

25) *LINES 820–860* — Finding of the object entered along with the command, and giving it a number which is stored in the variable ‘O’. If the object is not within the vocabulary, then ‘Pardon?’ is printed out in line 860. Line 840 checks for any other object being accepted instead of the object typed in — when typed out in full, an object may contain letters which appear as the first three letters in the same order as in the other object. The method for checking this is as follows: the object that is being taken in preference is checked for, along with D\$ equalling the four characters in that object name which have the second, third, and fourth characters equalling the first three letters of the object that is not accepted. The first letter in the string that D\$ is checked against, is the letter in the accepted name which precedes the other three characters. For example, the object, ‘CORD’, could be mistaken for a ‘SCORPION’ — therefore ‘J’ is checked for equalling ‘28’, which corresponds to the ‘SCORPION’, along with D\$ equalling ‘SCOR’. The second, third, and fourth letters correspond to the first three letters in ‘CORD’; ‘S’ precedes ‘COR’ in ‘SCORPION’.

26) *LINE 870* — If a space is missed out between the command and the object, then ‘N’ will equal one, and so ‘Learn to type’ will be printed on the screen.

27) *LINE 880* — List of the line numbers to which the computer may jump to depending on the value of ‘M’, the command number.

28) *LINE 890* — Line where the computer requests the player to start again.

29) *LINES 900–930* — Four lines containing messages which the machine may have to jump to quite often.

30) *LINES 940–980* —‘GET’ statement.

31) *LINES 990–1030* —‘DROP’ statement.

32) *LINES 1040–1070* —‘WEAR’ statement.

33) *LINES 1080–1120* —‘EAT’ statement.

34) *LINES 1130–1170* —‘READ’ statement.

35) *LINES 1180–1210* —‘LIGHT’ statement.

36) *LINES 1220–1250* —‘PAT’ statement.

37) *LINES 1260–1290* —‘EXAMINE’ statement.

38) *LINES 1300–1310* —‘TURN’ statement.

39) *LINES 1320–1350* —‘FEED’ statement.

- 46) *LINES 1360–1390* —“DUST” statement.
- 41) *LINES 1400–1440* —“OPEN” statement.
- 42) *LINES 1450–1470* —“PUNCH” statement.
- 43) *LINES 1480–1500* —“PULL” statement.
- 44) *LINES 1510–1560* —“FILL” statement.
- 45) *LINES 1570–1610* —“CUT” statement.
- 46) *LINES 1620–1670* —“DIG” statement.
- 47) *LINES 1680–1710* —“HIT” statement.
- 48) *LINES 1720–1770* —“THROW” statement.
- 49) *LINES 1780–1800* —“CLIMB” statement.
- 50) *LINES 1810–1850* —“MELT” statement.
- 51) *LINES 1860–1880* —“KILL” statement.
- 52) *LINES 1890–1930* —“SHOOT” statement.
- 53) *LINES 1940–1970* —“SPEAR” statement.
- 54) *LINES 1980–2100* —“STAB” statement.
- 55) *LINES 2120–2030* —Working out of the adventurer’s score.

This program is a sequel to “Dracula’s castle”, the objective being to return to your home city again with the jewel, and without being killed. Like “Dracula”, it takes up virtually the full amount of memory, and so you may need to make a few adjustments if you have a 16K machine.

The plan, along with the necessary documentation, should be sufficient to give you an idea of what is happening in the program. After all, you always have Chapter Two and the Appendices to refer to if you have any queries about either the structure or the content of the program.

CHAPTER 7

POSSIBLE IMPROVEMENTS FOR ADVENTURES IN THE FUTURE

At present, a typical layout for an adventure is as follows: the room name is printed out at the top of the screen followed by the “Objects” and the “Inventory” below it. There is then the routine to input commands, and then the reply to each command entered is worked out and printed on the screen. The 16K BASIC Adventure is limited to about fifty rooms with at least one incident per room — a greater number of rooms can be included if the adventure is in machine code which is much more compact than BASIC, but it is harder to program in such a language, and it would require a great deal more time and effort.

The immediate difference in an adventure with an increase in the amount of memory available would be in the actual size, for by doubling your memory capacity from 16K to 32K, many more rooms can be added, and more routines can be allocated for each room. Now that a large number of the newer computers on the market have a potential for graphics, a greater memory capacity will allow limited graphics in adventures for these computers: this could be either as a bird’s eye view of the room, or as a 3D representation. Obviously, with 32K of memory, the pictures on the screen cannot be too complicated, but when future computers come out with a standard memory capacity of 256K instead of 16K which it is now, some quite detailed pictures could be drawn on the screen, although it would be made more efficient if disk drives were connected. However, when such large amounts of memory are being moved around, it will not be feasible to use BASIC since it would be too slow; so therefore the only real solution to programming adventures like these will be to use machine code. This could mean that it would no longer become reasonable to write your own adventures, because such a large amount of work, in the drawings and everything else, would be required — the likes of the adventures in this book would therefore not be comparable with this. However, the redeeming factor will be that the new breed of adventures, which will arrive along with next generation’s computers, will be more expensive to buy than the traditional adventures, since you will basically be paying for the amount of time spent on writing each adventure. I consequently foresee that the traditional adventures will still be around with the newer adventures on the market, simply because of the price difference.

Two other recent features on micros that can be incorporated into adventures are colour and sound. In the future the colour will be

incorporated into the 3D representations, as may be expected, and the sound will be used to reflect the surroundings — there are also possibilities for speech synthesis whereby you hear the results to your commands as well as seeing what happens on the screen. A typical format for one location could be as follows — the picture may be that of a gorge with a river flowing through it; the player would be able to see the movement of the river and also hear the sounds made by the river. Even further into the future, animated graphics may be used for continuous movement, where the perspective changes smoothly as you move between locations. However, this would not only require a large amount of memory, but also a great increase in the speed at which pixels are moved around the screen.

Along with speech synthesis, the converse of this, namely speech recognition, may be brought into adventures. Instead of having to type in commands and objects, you would be able to say them, thus saving time in typing. In a graphics screen the objects would be seen, but an inventory may not be given, since an enquiry by speech to the computer would prompt the response of what is being carried at any one point in a game. On the other hand, the graphics may not take up all of the screen, and room may be left for such things as an inventory or results of commands.

As well as using speech for input and output, the actions required could be carried by using an advanced joystick: by pushing it forward, the intention would be for movement in that direction, and by pulling it back there would be the expected effect of movement backwards — movement of the joystick right or left would produce a rotation in the selected direction, and by pulling it up, or by pushing it down from its centre point, the player would be able to move up or down. “GET” and “DROP” could be implemented by grasping the joystick more tightly for one, and by pressing a release button for the other, along with a method in each for deciding which object is to be picked up or dropped. Attacking a creature could be made possible by grasping it tightly while pushing it forward, and likewise, retreating could be done by the same method, with the exception of moving it backwards. By twisting it one way, the player could “TIE” something, and by twisting it the other way, this object could be untied. There are various possibilities for joysticks being used in adventures, but Obviously they would require to be very tough for such vigorous movement.

Possibly the final step that I can foresee in adventures is linking up the same game to two or more computers and have several players battle it out against each other, each player seeing the game from their own viewpoint. They would be able to make treaties with each other or gang up against each other, while at the same time they would be trying to solve the problems which the computers may give. This has been done before on mainframe computers, but only in the traditional adventure format. The

advanced adventure will not have the time measured by the number of moves made, but instead it will be measured in seconds.

From this chapter you should see that there are great possibilities in the advancement of adventures, but at present, the main prevention of these advances is in the technology of the computers which are now available on the market.

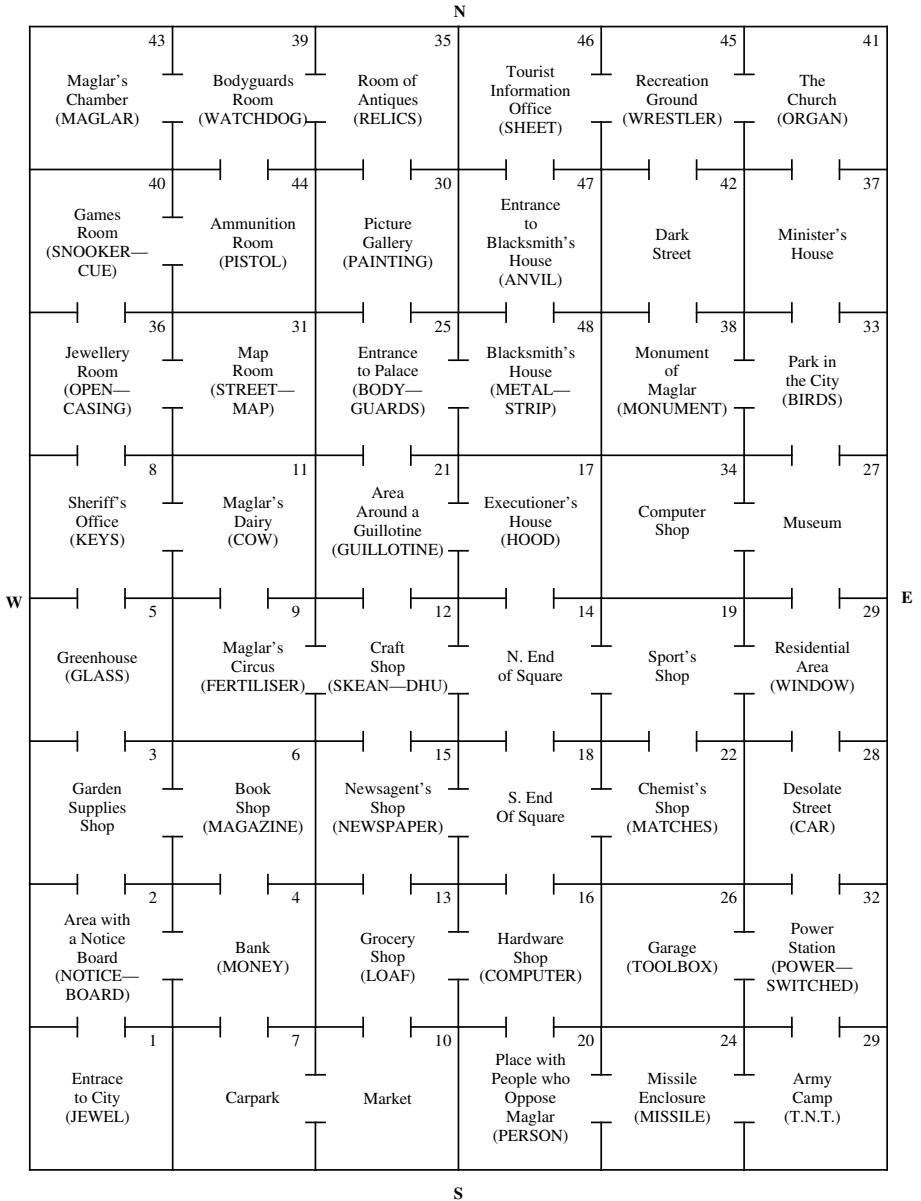
CHAPTER 8

NOW WRITE YOUR OWN ADVENTURES

To give you a start in writing your own adventures, I will give you the plan of an adventure along with a list of what should happen in each of the rooms. The adventure is called “Tyranny of Maglar” and is the third of the trilogy - of adventures starting with “Dracula’s castle”. The main purpose of this adventure is to return the “JEWEL” to its correct place, and to kill “MAGLAR” who took control of your kingdom on hearing of your capture by Dracula. As well as this, you will have to return the situation to what it was before, for “MAGLAR”, in his greed, took over your palace, and took various items from their proper place for this palace, and so there has to be a way for the player to find out which objects must be returned before the adventure can be completed. An interesting feature of this adventure is the ability for the player to trade objects for some form of money and so be able to gain new objects which will be useful. You may wish to include the ability ; for the player to find out his/her score while playing the game, and this can be worked out according to the number of objects in their correct places — the final score will be as in previous adventures, with the exception that added to it will be the values for the objects (say 16 points for each object sorted out). For this you will have to remember to prevent the command “SCORE” from being accepted as the command “SOUTH”, and also remember that it does not take an object.

You should try and write the adventure according to the format in Chapter Two, using the data given in the rest of this chapter. However, if you have any novel ideas about things which could happen in the adventure, then by all means use them, for the situations which I suggest are not inflexible, and the purpose of writing an adventure is to use your own imagination, and creative instincts to challenge other people to win in the hypothetical situations in which you place them — remember that it is you, the programmer, who is in control of what happens to the players in adventures. The more you decide to use your own ideas, the greater will be the pleasure that you will gain out of writing adventures, provided that these ideas are logical.

Plan



ROOMS

The room names are given below in numerical order along with their expected functions in the adventure:

1) *ENTRANCE TO CITY* — This is the starting location for the adventure, and the first action of the adventurer should be to pick up the “JEWEL” that is lying on the ground, for it must be returned to the appropriate place.

2) *AREA WITH A NOTICE BOARD* — On the “NOTICE-BOARD” can be information pertaining to the recent removal of a “PAINTING” and some “RELICS” from the “Museum” to be put in the palace. This will be a small piece of helpful information for the player to start off with, for these are two of the objects which must be returned to their correct place.

3) *GARDEN SUPPLIES SHOP* — If you “DROP” the “FERTILISER” here, which you can pick up from the circus, then you will receive a “TOKEN”. This “TOKEN” may be used to buy some “MATCHES” from the “Chemist’s shop”.

4) *BANK* — The money here will be sufficient to allow you to buy a “LOAF” of bread from the “Grocery shop”. It is lying around and nobody will notice you taking it.

5) *GREENHOUSE* — The extra piece of “GLASS” left here from the building of the greenhouse, will allow you to repair a broken “WINDOW” in the “Residential area” — this would count as score when the repairs have been done.

6) *BOOK SHOP* — You will find a “MAGAZINE” here; it is free and need not be paid for with anything. If you read it, then you will find an advertisement for the “Craft shop”, and in particular, about a “SKEAN-DHU”, which is a type of dagger.

7) *CARPARK* — If you “DROP” the repaired “CAR” here, then a “CREDIT-CARD” will fall out of a slot in the metalwork. It turns out to be the one which you had lost a long time ago, before your journey to “Dracula’s castle”.

8) *SHERIFF’S OFFICE* — Some “KEYS” can be found here and these may be picked up. The purpose of them is to “LOCK” up the casing in the “Jewellery room” after putting the “JEWEL” back into it, so that it is kept safe. Points are scored for the return of the “JEWEL”.

9) *MAGLAR’S CIRCUS* — Some “FERTILISER” can be found here, which was kindly left by one of the elephants. If you take this to the “Garden supplies shop”, then you will be given a token in return for it.

10) *MARKET* — You are able to “SELL” a “COW” here which you are able to steal from “Maglar’s dairy”. In return for selling the “COW”, you

will receive a “GUINEA”.

11) *MAGLAR'S DAIRY* — A “COW” is in here when you enter, and you are allowed to take it. As you have found out from the information about the “Market” above, you can “SELL” this “COW” for some form of money.

12) *CRAFTSHOP* — You will be able to “BUY” a “SKEAN-DHU” from this shop with the “GUINEA” which you will receive from selling the “COW”. The “SKEAN-DHU” is the object that you will require to kill “MAGLAR”.

13) *GROCERY SHOP* — A “LOAF” is here, and you will be able to “BUY” it with the “MONEY” which you can obtain from the “BANK”. The “LOAF” is used to “FEED” some “BIRDS” which you will find in the “Park in the city”.

14) *N. END OF SQUARE* — This room has no function apart from acting as a connecting room. Perhaps you will want to think up some sort of other purpose for this location.

15) *NEWSAGENT'S SHOP* — A “NEWSPAPER” is lying on the counter, and the newsagent recognises you as you enter, and allows you to take this object. The “NEWSPAPER” tells you about a new “MONUMENT” which Maglar has erected for himself. To gain score from this you will have to knock this “MONUMENT” down.

16) *HARDWARE SHOP* — When you receive the “CREDIT-CARD” from dropping the repaired “CAR” in the “Carpark”, you can use it to “BUY” the “COMPUTER” which is lying about in this shop.

17) *EXECUTIONER'S HOUSE* — You will find a “HOOD” here which will disguise you from the “BODYGUARDS” who will kill you if they recognise you. This is an object which must be worn before it will have the desired effect.

18) *S. END OF SQUARE* — Like the “N. End of square”, this room acts as a connecting room, without a function, so you have the option of putting something here yourself if you wish.

19) *SPORT'S SHOP* — If you take the “SNOOKER-CUE” from the “Games room” and “DROP” it here, then you will be given the option of picking up a “JAVELIN” which will appear.

20) *PLACE WITH PEOPLE WHO OPPOSE MAGLAR* — When you come into this room, then a “PERSON” will come up to you and tell you that you must be wary because there have been muggings recently in the “Dark street”, so you will have to carry some form of light with you when you enter this location.

21) *AREA WITH A GUILLOTINE* — You will be given one move in which you will have the chance to dispose of the “GUILLOTINE”. To do this, you will have to “THROW” the “T. N. T.” at it, and with a bit of luck,

it will be blown up before it chops your head off. If you attempt to leave this room before carrying out this action, then you will be killed.

22) *CHEMIST'S SHOP* — You are able to ‘BUY’ some ‘MATCHES’ here with the ‘TOKEN’ which you can obtain from the ‘Garden supplies shop’. To light them, you can use either the command ‘LIGHT’, or the command ‘STRIKE’, and you will find that they stay alight for a long time.

23) *RESIDENTIAL AREA* — There is a ‘WINDOW’ here, and if you ‘EXAMINE’ it, then you will discover that it is broken. If you ‘REPAIR’ it with the ‘TOOLBOX’ from the ‘Garage’, and the ‘GLASS’ from the ‘Greenhouse’, then you will receive points to be added to your score.

24) *MISSILE ENCLOSURE* — You will find a ‘MISSILE’ here and if you try and ‘THROW’ it, or ‘EXPLODE’ it in any way, then you will be killed, for it is a red herring.

25) *ENTRANCE TO PALACE* — You have to pass Maglar’s ‘BODYGUARDS’ here, and to do so, you will need to be wearing the ‘HOOD’ from the ‘Executioner’s house’, for otherwise they will recognise you and kill you.

26) *GARAGE* — A ‘TOOLBOX’ is here which can be picked up. It has two functions in this adventure: in one instance it can be used to help ‘REPAIR’ a ‘WINDOW’, and in the other, you are able to ‘REPAIR’ the ‘CAR’ which you can obtain from the ‘Desolate street’.

27) *MUSEUM* — This is where you can leave two items, to gain two lots of points. One object is the ‘PAINTING’, and the other is the ‘RELICS’, both of which you are able to find in the palace.

28) *DESOLATE STREET* — A ‘CAR’ can be picked up from here. On examination of it, you will find out that it requires several repairs, so you have to find something with which you can ‘REPAIR’ it.

29) *ARMY CAMP* — Some ‘T. N. T.’ can be found here, and it should be used to remove the ‘GUILLOTINE’ from your path before it kills you. Be careful, for if you simply ‘DROP’ it, then it will explode, and you will be blown up with it.

30) *PICTURE GALLERY* — This is a location inside the palace, and in it is a ‘PAINTING’, which Maglar stole from the ‘Museum’, so you will need to return it there to gain points.

31) *MAP ROOM* — In this room is a ‘STREET-MAP’, which you must pick up and return to the ‘Park in the city’, from where Maglar stole it for his own personal use. By doing so, you will receive points to add to your score.

32) *POWER STATION* — Whenever you enter this room, you should switch off the ‘POWER-SWITCH’, for it will switch off the lights to the palace. If you enter the palace when the lights are still on, then you will be

killed by one of the “BODYGUARDS”.

33) *PARK IN THE CITY* — There will be some “BIRDS” here, and if you decide to “FEED” them with the “LOAF” of bread, then they will tell you that Maglar stole the “STREET-MAP” from this location, and they will ask you to retrieve it for them.

34) *COMPUTER SHOP* — By dropping the “COMPUTER”, which you had bought from the “Hardware shop”, here, a “SLEDGEHAMMER” will appear in its place. This object is required to knock down the “MONUMENT” which was erected by Maglar, in the location entitled “Monument of Maglar”.

35) *ROOM FULL OF ANTIQUES* — Some “RELICS” can be found here, and they must be picked up, and then dropped again in the “Museum” for you to gain points.

36) *JEWELLERY ROOM* — You have to replace the “JEWEL” in the “OPEN-CASING”, which you will find in here. The door to this casing must be locked with the “KEYS” from the “Sheriff’s office” to gain the points you deserve for retrieving it from Dracula.

37) *MINISTER’S HOUSE* — In here you will find nothing if you enter without having previously played the “ORGAN” in “The church”. If you have played it then you will find a “BIBLE” here. With this in your possession when you try and kill “MAGLAR” with the “SKEAN-DHU”, your action will be rewarded, but, on the other hand, if you do not have this object, then the “BODYGUARDS” will intervene at the crucial moment and kill you.

38) *MONUMENT OF MAGLAR* — The “MONUMENT” here must be knocked down with the help of the “SLEDGEHAMMER”, and by doing so, you will gain some points for your score.

39) *BODYGUARDS’ ROOM* — Instead of finding some “BODYGUARDS” here, you will find a “WATCHDOG”, which will not let you past it. To remove it from your way, you will have to “SHOOT” it with the “PISTOL” which you will find in the “Ammunition room”.

40) *GAMES ROOM* — A “SNOOKER-CUE” can be found here. This is linked with the “Sport’s shop”, and if you “DROP” it there, then a “JAVELIN” will appear in its place.

41) *THE CHURCH* — An “ORGAN” is here, and cannot be moved. However, you are able to “PLAY” it, and the result of doing so is for a “BIBLE” to appear in the “Minister’s house”.

42) *DARK STREET* — To pass here without being mugged and killed by some layabouts, you have to be carrying the “LIT-MATCHES” which you will have obtained from the “Chemist’s shop”.

43) *MAGLAR’S CHAMBER* — Maglar can be found here, and you must

‘KILL’ him with the ‘SKEAN-DHU’ from the ‘Craft shop’. By killing him, you will gain points.

44) *AMMUNITION ROOM* — You will need the ‘PISTOL’ from here so that you will be able to pass the ‘WATCHDOG’, which you will ‘SHOOT’ in the ‘Bodyguards’ room”.

45) *RECREATION GROUND* — A ‘WRESTLER’ is here and will not let you pass to the east. To pass him you have to ‘THROW’ the ‘JAVELIN’ from the ‘Sports shop’ at him. This will distract him, and you will be able to pass into the next room.

46) *TOURIST INFORMATION OFFICE* — A ‘SHEET’ can be picked up from here and it will give details about the ‘JEWEL’ being returned to the ‘OPEN-CASING’, if you decide to ‘EXAMINE’ it, or ‘READ’ it.

47) *ENTRANCE TO BLACKSMITH’S HOUSE* — There is an ‘ANVIL’ at this location, and you can use it to make a ‘HORSESHOE’, which will give you a little good luck.

48) *BLACKSMITH’S HOUSE* — You will find a ‘METAL-STRIP’ here and will need to use the ‘ANVIL’ in the entrance to the house, to make the ‘HORSESHOE’ out of it. If you are not carrying the ‘HORSESHOE’ when you complete the last point-scoring part of this adventure, then you will receive some bad luck —this will be in the form of some sort of death.

The above information, along with that in Chapter Two, should be sufficient for you to write this adventure. Once you have tried this adventure, . then you will wish to write adventures of your own — you will need to make up a plan, and so on, following the details in Chapter Two. Note that the - objects and commands of this adventure, and the details on how to complete it are not given in the appendices. This is to give you the chance to change it to suit yourself, although pointers are given for the objects and commands to be used, as well as virtually all the situations in each room.

From this point on, you are on your own, regarding the making of adventures, so I must wish you success in making them in the future. I trust that this book has proved useful to you in both learning how to program BASIC adventures, and in improving your programming structure — I, myself, started to program by writing adventures, and through this, my skills as a programmer developed.

APPENDIX I

THE COMMANDS AND THEIR MEANINGS

The Commands used in all three of the adventures in this book are listed below in alphabetical order. The letters in brackets after each command correspond to the adventure, or adventures, that the command comes from: “C” stands for the model adventure, “Captive”; “D” stands for “Dracula’s castle”; and “J” stands for “Journey to freedom”. After this, the meaning of each command is given.

1) *BLAST* (D) — This command is used with connotations of obliteration, with a more destructive force of wiping out than with shooting. Obviously an object must be carried which the player can blast with, and in the context of the adventure the passageway is cleared of obstacles which are in the adventurer’s way.

2) *CHOP* (D) — This may have a similar useage as such commands as “KILL” and “SHOOT”, and various others since it allows the player to take away the life — if a “VAMPIRE” may be said to be alive in the first place — of a character opposing him or her. Again, the adventurer will require a suile implement to carry out the command successfully. you could also use this command to chop such things as wood, for example.

3) *CLIMB* (DJ) — In many cases a “ROPE” or similar object may be required, but in other cases it may not. If a “ROPE” is required then it should be tied or in some way attached to something before it may be climbed. Depending on the context, the player could either climb up or down it. In the cases where a “ROPE” is not required, the player may be in the positoin to climb something like a tree, and in such cases there are usually hints that the tree may be climbed. For example, the computer may print out that a low branch is within easy reach, or on examination of the tree by the adventurer, the machine could print out that it is climbable.

4) *CUT* (CDJ) — In my adventures I have not used this command with any association of death, but instead for various uses. In one instance it is used to form something new out of something existing; another time it allows an object to be freed from whatever it is attached to and thus enable it to be picked up; in the other situation it removes trees from an area, revealing a certain object — that is, it removes something from the surface and renders visible an object which could not previously be seen.

5) *DIG* (J) — This has a use in “Journey to freedom” which is similar to the second use of “CUT”; if a “POST” is dug out of the ground then it

can be picked up and carried. Another possible use of the command could be to reveal something to the adventurer — for example, if ‘DIG GROUND’ is typed in, assuming that ‘GROUND’ is listed under ‘Objects’, then the computer could print ‘Some treasure has appeared’. The word ‘TREASURE’ would then appear under the list of objects in the room. Note, however, that this can only take place on the provision that six objects are not already in the room, since there cannot be seven in a room (the seventh would be the ‘TREASURE’ if there were already six objects). You should also remember than an object like a ‘SPADE’ must be carried before the player can ‘DIG’.

6) *DROP* (CDJ) — This is one of the essential commands in any adventure, since without it and ‘GET’ the player would not be able to transfer objects freely from one room to another. There are not a great deal of conditions surrounding this command apart from that the object must be in the adventurer’s possession to begin with, before it can be dropped. Since the only objects that may be picked up are moveable objects there need not be any conditions about them being dropped, for only such objects are carried.

7) *DUST* (J) — In ‘Journey to freedom’ this has one function which is implemented in two different ways. In one case, with the help of a ‘DUSTER’, some rust may be dusted off a key to make it fit a lock, and in the other case a ‘DIAMOND’, when dusted, becomes a ‘lethal weapon’. The object acted upon changes to the benefit of the player.

8) *EAST* (CDJ) — This is a command of movement and may be used by just typing the first letter, ‘E’, into the computer and not the first three letters as in commands which do not concern movement. The purpose of this command is to move the player to a location, or room, ‘EAST’ by one room square, of the room presently occupied provided that there is an exit to the east. Remember that directions do not require an object.

9) *EAT* (J) — This command, when used, is not often to the player’s benefit — the object, more often than not of food, which the player is allowed to eat, will be required for another part of the adventure. For example, the ‘FOOD’ which may be eaten in the adventure is for feeding a ‘LION’ with. An example of eating being beneficial is where a ‘SCORPION’ is eaten in the same adventure — the creature would otherwise kill the adventurer.

10) *EMPTY* (D) — This is used for some sort of container which has some fluid inside it which is usually water. In ‘Dracula’s castle’ the container is a ‘BUCKET’, the fluid is some water, and the result of emptying this container is to extinguish a fire sufficiently so that it may be passed. A possible alternative to ‘EMPTY BUCKET’ could be ‘POUR

WATER”, but “WATER” is not included in the object list, since examination of the “BUCKET” reveals that there is water in the “BUCKET”. “POUR” is similar to “EMPTY”, but differs in that it refers to the fluid in the container directly and not to the container itself.

11) *EXAMINE* (CDJ) — A great deal of use is made of this command in adventures; although it is not one of the six essential commands, it is resident in most adventure vocabularies. Examining objects can reveal attributes about them which the adventurer may take advantage of. For example, by examining a “DESK”, the machine could say that a “PISTOL” and a “LETTER-OPENER” are in the “DESK”, and then print these two objects under the heading “Objects” in the display format.

12) *FEED* (J) — When an adventurer meets some sort of creature which refuses to move out of the way and sounds rather hungry, by feeding it some food, it could be persuaded to move away and so let the adventurer Ijrass. By doing this, the food will be used up as the programmer Intended, instead of as a meal to a player who types in “EAT FOOD”. After being fed, the creature is usually removed from the game — a useful point to note when programming this type of thing, is that when the creature is in the room, an exit will be barred from the adventurer, and when it has been removed, the player will be allowed freedom of movement (when “E” of the object name equals “A” and the input for movements is in the direction of the barred exit, then the computer prints c.mt that the creature will not move out of the way and jump for the next Input).

13) *FILL* (J) — If an adventurer has some sort of container and a particular type of fluid is available for the taking, but requires something to carry it in, then the command to use is “FILL”. The fluid may then be carried about in the container. In “Journey to Freedom” the fluid is “OIL” and the containers in which it may be carried are some “CANS”. However, when the container is dropped, the fluid must be dropped as well, so there has to be provision for this. In the above mentioned adventure, the “OIL” is transferred into your possession by simply picking it up while carrying the “CANS”. The actual command “FILL” is used to transfer the “OIL” into a “CHAIN- SAW”; the opposite of this command is the command “EMPTY”.

14) *GET* (CDJ) — This is the second of the two essential commands which require objects, the other four being directions which do not take an object. The purpose of this command is to enable the adventurer to pick objects up on the condition that the programmer has decided that the chosen objects are moveable by the player.

15) *HAMMER* (D) — In the context of my adventures this command is used to drive a “STAKE” into “DRACULA” and his “VAMPIRE” colleague, provided that a “MALLET” is in the possession of the

adventurer. A possible usage which I have not yet seen in an adventure could be to put a "POST" or something similar into the ground so that there would be something to attach a "ROPE" to for climbing.

16) *HIT* (J) — A player often has to use force on creatures that are in the way, and "HIT" is a command which allows the player to use violence on such creatures. In most cases a certain implement may be required to do the hitting. In the third adventure, a "POST" must be used against a reluctant "BEAR" before it will move. It is then removed from the adventure by letting "E" of the object number equal fifty, which is one more than the number of rooms.

17) *KICK* (CD) — Like the previous command, this is a command of violence against various unhelpful objects. However it is different in the respect that no objects need be carried since the implement that does the kicking is the player's foot. The common purpose of such commands are either to preserve the life of the adventurer, because it is said that attack is the best form of defense, or to clear the passage so that the player can move past.

18) *KILL* (CDJ) — This is the third of three similar commands. The way in which this one differs from the other two is in that the recipient of the blows suffers more harm to a fatal extent. Some sort of weapon is usually needed, but in the incidence in the second adventure, the player's bare hands are sufficient to the deed.

19) *LIFT* (D) — There is a greater force behind this command than "GET", which does not imply much exertion. In "Dracula's castle", a "COFFIN" may be lifted, and the effect of doing so moves it enough to reveal a "VAMPIRE". The command is associated with the movement of an object which is too heavy to be carried.

20) *LIGHT* (CJ) — This is a popular command, since a great many adventures have torches or some other light-emitting devices which must be lit to show the way in a darkened section of the adventure layout — failure to light the device can result in death. Quite often, only a torch is required, but in several cases batteries may be needed before it may be lit. To light the "TAPER" in "Journey to freedom", the player must be carrying the "MATCHES" as well as the "TAPER".

21) *LOWER* (D) — Often, the adventurer will be required to "LOWER" a "ROPE" before it is possible to "CLIMB" down it — if one starts to climb down a "ROPE" which is still where one started to climb from, then it is more than likely that one will fall. There are other possibilities for this command: for example, in a situation, one may wish to cross a drawbridge, there could be a necessity to lower it first of all, so therefore the command

and object typed in first of all would be ‘LOWER DRAWBRIDGE’, which would be followed by ‘CROSS DRAWBRIDGE’.

23) *MELT* (J) — As it may seem, this command is used for objects to be melted, as long as there is, of course, something to melt them in. In ‘Journey to freedom’, the objects are melted in a ‘CRUCIBLE’ — the first object to be melted must be some ‘SNOW’ to wash the ‘CRUCIBLE’ out and then a ‘CROSS’ may be melted down. Once this has been done, by some sort of miracle a ‘FLICK-KNIFE’ is formed. If an object like the ‘CROSS’ is examined then the computer could print out: ‘This looks as if it can be melted’. The player’s reaction on finding the ‘CRUCIBLE’ would then be one of having found something to melt the cross in. If the computer has more memory then helpful hints like this can be given to aid the adventurer in such situations, since I admit that the solution is not too obvious, but that is how it is with adventures — some problems that you set are more logical than others which may suffer from ideas that are a bit too far fetched. However, if your adventure is completely logical, then it may be too easy for the player and it would not be much of a challenge, so the programmer should always strive for a balance point between credibility and problems to which there are no obvious solutions. When one does meet a problem which is not obvious at first sight then the solution is generally found by trial and error — if, on completing this problem, the solution then appears perfectly logical, then one would curse oneself for not having thought of it sooner, but, on the other hand, if the solution is arrived at by pure chance, and the problem seems completely stupid, then the programmer would be thought of as some idiot who specialises in daft ideas (many people who write adventures fall into this category at some stage or other). Remember that what may seem logical to the programmer could appear illogical to the player.

24) *MIX* (D) — This command, like many others of those under analysis in this section of the book, are specific to just one incidence in one adventure. It may have other applications which could arise under various situations, but the use that I have made of it in ‘Dracula’s castle’ is for mixing some chemicals together along with a made up formula, with the result of forming a glue, which can only be dropped in the one room. I am more pleased with the routine for ‘MIX’ than with the routine for ‘MELT’, since, with the former, the prompt of ‘With what formula?’ comes up, and there is only one given formula in the adventure — the presence of formula points towards the chemicals being mixed under specific conditions.

25) *NORTH* (CDJ) — The purpose of this command of movement, where only the first letter ‘N’ need be typed in, is to move the player to the location immediately north of that room presently occupied, provided that there is an exit in that particular direction from the room that the player is in.

26) *OPEN* (CDJ) — This normally refers to some form of door, but can be used for any object which may have the status of being either open or closed. A key may often be required to open the door, which usually stays open, although it sometimes closes behind the adventurer. A door acts as a barrier and can be counted as an obstacle which has to be cleared out of the way before the adventurer can proceed. The converse of this command, ‘CLOSE’, is not very often used.

27) *PAT* (J) — In the third adventure there is a situation where a dog is in the way of a player, and since a dog is ‘man’s best friend’, it might be more beneficial to be kind to the dog, rather than attack it; so one of the actions that one might do in kindness towards the dog is ‘PAT’ it. I cannot foresee much greater useage than this for such a command in an adventure.

28) *PLAY* (D) — This command is used twice in ‘Dracula’s castle’, in one instance to ‘PLAY MUSIC’, and in the other to ‘PLAY’ a ‘RECORDER’, the latter being used more frequently in adventures. The first use transports the adventurer, by some form of magic, into a concealed passageway, and the second use is to give the player some extra information which would be on the cassette in the ‘RECORDER’. Sometimes it is necessary to find the cassette and insert it before the machine can be played.

29) *PRESS* (D) — When there are any types of switches or buttons lying around, then an attempt should be made to ‘PRESS’ them, since they could disclose a secret. However, they may be equally dangerous, because - the wrong button from a row of several buttons, or buttons pressed in the wrong order, can lead to the death of the player. There are many possibilities for the things that may happen from the pressing of buttons.

30) *PULL* (J) — This is similar to the previous command, but different in - that a greater exertion is required and that the object, often a lever-type object, is pulled, rather than pushed, ‘PUSH’ being a possible alternative for ‘PRESS’. ‘PULL’ may be used for buttons as well, to confuse players who would be expecting ‘PUSH’ or ‘PRESS’.

31) *PUNCH* (J) — Another variation of violence against annoying creatures is contained in the command ‘PUNCH’, in ‘Journey to freedom’. I have made it necessary for a ‘BOXING-GLOVE’ to be worn before the desired effect of being thrown into the next room is obtained — without wearing it, adventurers would meet their deaths. Such necessities before particular actions are carried out add more interest to the game.

32) *QUIT* (CDJ) — Although this is not essential, it is a standard for the adventure game, and only needs one program line to install it. Its purpose is to allow a player to start the game again from the beginning, if a mistake

has been made that cannot be corrected, for example, or if time is running too short and the position is hopeless. No object is required of this command.

33) *READ* (CDJ) — This is another popular command which, like ‘PLAY’ gives some extra information to the player, and it can also produce some ‘magic’ effects if a spell is ‘READ’ from such things as scrolls. You may, however, be given details about spoken words which can affect the adventurer’s position — most of the time the spells in adventures tend to move the player around the network of rooms. Warnings may also be given through the reading of material.

34) *RING* (C) — One should be always careful of the command ‘RING’, although it can be beneficial, for one of the most tempting things to do when one comes across a bell, is to ‘RING’ it — this may produce some sort of catastrophe which kills the adventurer; it may do nothing; or it may be useful. The choice is at the discretion of the programmer.

35) *SAY* (C) — When a magic word has been revealed to the player, then the usual way to use it, is to ‘SAY’ it, but the problem is in working out where it can be said, and having something happen as a consequence, because most of the time such a word is said the computer prints out the words ‘Nothing happens’, so allowance for this must be made by the programmer.

36) *SHOOT* (J) — This is yet another word of violence, and you may be pleased to know that there are only another two such words of direct violence left to deal with, but it is best to have a reasonable vocabulary of these words in an adventure. This command needs the player to have some sort of trigger-fired weapon (it could be a gun or a crossbow, for example) which would have a clean impact without the devastating force of ‘BLAST’. The object being shot at is not always killed, and occasionally stunned — it very often happens that it is removed from the game, and seldom revives.

37) *SOUTH* (CDJ) — The third command, alphabetically, of direction is ‘SOUTH’ and as in all such commands, the first letter only need be typed in, and no object is taken. The purpose is for the player to move one position south provided that an exit lies in that direction.

38) *SPEAR* (J) — The object that the player must carry when using this command is obviously a spear, and the object that is to be printed after the command is the creature that the spear is to be thrown at. The command is fairly self-evident in purpose, and need not kill. It may often injure, but if it is thrown at the wrong beast the programmer could decide to either make it miss, or else let the creature turn on the player. Programmers usually prefer

to use the latter possibility.

39) *STAB* (J) — Like the previous command, this is used in the art of killing, or causing injury to monsters. It gives a clean blow to the creature provided that some form of knife is carried. The use of commands like this and “SPEAR” (“KNIFE” could be an alternative for “S ”) which are specific to their purpose help to build up the vocabulary. A command like “THROW” can replace the more specific commands if there is a shortage of memory.

40) *TAKE* (C) — This command is an alternative for the command “GET” which some people may prefer. I, myself, prefer using “GET”, and have included this alternative as an example of how easy it is to add extra commands which have exactly the same purpose as others — in my earlier adventures I did not develop this feature.

41) *THROW* (CJ) — A variety of uses can be made of “THROW”, since a good many of the moveable objects are able to be thrown. To climb over an obstacle like a “WALL”, a “ROPE” can be thrown and it may catch securely enough for the player to climb. Weapons can be thrown, as well as various other objects, to pave the adventurer’s way — for example, a “GRENADE” can be thrown to remove some boulders, and a “CHEST” may be thrown to form a platform across a river.

42) *TIE* (D) — The main use for this command is to “TIE” a “ROPE” so that one may climb down without falling. The converse, “UNTIE”, could be used to “UNTIE” a “ROPE” that one has just climbed up, if there is the possibility of requiring it a second time.

43) *TRANSMIT* (C) — This is a command that does not require an object, and it is used to “TRANSMIT” a signal from a “TRANSMITTER”. In “Captive”, the conditions required to make this work are cooling it down and giving it an “AERIAL”. The command “TRANSMIT” can then be used, and the result could be from a variety of things that the programmer would choose from — in this case it reveals an exit from a room which was not there previously. It is a very specific command.

44) *TURN* (J) — In “Journey to freedom”, this command refers to a “CUBE” (one of those that aggravates people), and the result of making this action is totally irrelevant to the adventure, as it says who the copyright of the program belongs to. It is good practice in adventures to contain red herrings to aggravate the player into trying to complete the adventure with more determination; however, you should not overdo it too much, for this could make the player question whether anything is relevant. A possibly better use of this command could be in a situation where the player finds an abandoned car, enters it, inserts a key into the ignition, and then turns the

key — the car could then be used to travel across a section over which the player is unable to survive, because of the large distance, if he or she travels by foot.

45) *UNLOCK* (C) — This is an alternative for the command “OPEN”, but it may have a separate function, because a “DOOR” may have to be unlocked before it may be opened. Where there is no sign that the door is locked, then just “OPEN” should be used, and where the unlocking of the door is combined with the opening of it, then “UNLOCK” may be used as well.

46) *WEAR* (CDJ) — This is a popular command as can be seen by the fact that it is used in all three of the adventures of this book. A lot of players can be caught out through the necessity to “WEAR” an object like a “BOXING-GLOVE”, or some “BOOTS”. It is used with any object that can be thought of, as being something that can be worn.

47) *WEST* (CDJ) — This is the fourth and last command of movement, and is typed in, in the same way as the other three. The purpose is to move the adventurer one location west of that location presently occupied, provided that there is an exit in that direction. The present location then becomes that location west of the original one.

The purpose of this section of the book is to detail the purpose of each of the commands used in the three adventures, and to suggest possible alternative usages of these commands. As can be seen, different commands are thought up to suit the situations in each adventure, but also several commands appear time and time again, and the relative popularity of usage of each command can be seen by looking at the number of letters, out of ‘C’, ‘D’, and ‘J’, which are in brackets after it. Obviously, many more commands can be thought of, but that is up to you to think of in your own adventures — the combined number of commands out of the three adventures is sufficient to show a general pattern in the various types of command.

APPENDIX II

THE OBJECTS AND THEIR USES

All the objects used in the three adventures are listed below in alphabetical order. As in “APPENDIX I”, the letters in brackets which are in this case - after an object, correspond to the adventure, or adventures that the object comes from: “C” stands for “Captive”. “D” for “Dracula’s Castle” and “J” for “Journey to Freedom”. The way in which the object is used is then given.

1. *ACID* (D) — Can be carried in a “CONTAINER” and is used in the removal of a “SPIDER’S WEB” which the adventurer may become caught in. Another usage could be for throwing at creatures and thus removing them from one’s way.
2. *AERIAL* (C) — It is necessary for this to be in the same room as the “TRANSMITTER” before the signal can be transmitted — this object is specific to the action of transmitting.
3. *AIR-CYLINDER* (J) — This must be worn so that the player will survive the thin atmosphere in a pass in some mountains. Such an object could also be used in an underwater adventure along with diving equipment.
4. *ANTEATER* (J) — As it sounds, this creature eats insects like ants and termites, the latter are in one adventure and are removed by dropping the creature in the same room. A “HARPOON” is revealed from under them.
5. *AXE* (D) — Although this may sound as though it will be used as a weapon it is actually used to free some “GARLIC” with the command “CUT”. It is more usually used as a weapon along with the command “THROW”.
6. *BEAR* (J) — This is a creature which needs to be driven away with a post, in “JOURNEY”, and when it moves the player is able to pick up a
7. *BELL* (C) — This is a red herring in “CAPTIVE” since it is very tempting to “RING” a “BELL” on seeing one — the result of doing so is death.
8. *BOOTS* (J) — Another object which must be worn before they operate properly. When worn, they prevent the player from sinking into a swamp.

9. *BOULDERS* (C) — These prevent the adventurer from passing into the next room and must be removed by throwing a “GRENADE” at them which easily removes them.
10. *BOXING GLOVE* (J) — This must be worn when punching a “GUARD” or else you will be killed. By telling players that they have been thrown into an adjacent room is another way of saying that they can pass the guard, although the decision is made for them by the guard.
11. *BROKEN-BONES* (D) — When some “GLUE” is dropped in the same room as them then they are glued together to form a “SKELETON-KEY” —bones are part of a skeleton.
12. *BUCKET* (D) — When examined the player will find out that it contains water. If this water is emptied in the room which has a fire in it then the fire is sufficiently extinguished to let the player pass.
13. *BULLET* (J) — A “SHOTGUN” is required before the “BULLET” can be shot at anything. The “BULLET” is obtained by examining a “FOUNTAIN”. The creature that is shot at is the reincarnated Dracula who is completely killed, since the “BULLET” is really made of silver, although the adventure does not say so. This fact does not change the situation surrounding the completion of the adventure, though.
14. *BUSHES* (J) — When these are examined they turn out to be really some clumps of “GARLIC” which can then be picked up and carried.
15. *CANS* (J) — These are used to carry some “OIL” from the “FUELTANK” to the “CHAIN-SAW”. If they are dropped without the “OIL” being transferred then the “OIL” in them returns to the area around the “FUEL-TANK”.
16. *CHAIN-SAW* (J) — This requires some “OIL” for it to function, and it is used to clear an area of trees and thus reveal a “SPADE” from beneath the tree cover.
17. *CHEMICALS* (D) — When mixed together according to a specific formula which must be found, a “GLUE” is formed for the adventurer to use.
18. *CHEST* (J) — Contrary to what one may think, this cannot be opened, but instead, when it is thrown, it acts as a platform across a river.
19. *CLOCK* (D) — In “DRACULA”, when you are in the room with the clock in it and “EXAMINE” the “CLOCK”, the time on it is set to a random between one o'clock and six o'clock. If the time is before four o'clock when you enter the “WEREWOLF'S CHAMBER”, then the “WEREWOLF” kills you.

20. *CLUE* (D) — This is simply something which may be read, and gives information which may be of some use to the adventurer. It is not essential that such pieces of information be read again for the adventure to be completed if the content of them is already known.

21. *COBRA* (J) — In ‘JOURNEY’ an opposing creature is a ‘COBRA’, the only thing that is required to move it is the possession of - ‘GARLIC’ along with its pungent smell — this is not the expected use of ‘GARLIC’, but any player would be quite pleased at removing such a beast with such a simple object.

22. *COFFIN* (D) — In ‘DRACULA’ you should by this time know that it is necessary to kill a vampire, but before you can do that you must ‘LIFT’ up the ‘COFFIN’ that the ‘VAMPIRE’ is in to reveal the creature.

23. *COMPUTER* (D) — When you are asked if you think that the adventure is good by the ‘COMPUTER’ and you do not reply with ‘Y’ then you will not be allowed to move until you do give this reply.

24. *CONTAINER* (D) — Some ‘ACID’ may be carried in this — the ‘ACID’ may only be picked up if the ‘CONTAINER’ is in the possession of the player.

25. *CORD* (J) — This is an alternative for the object ‘ROPE’, and suggests a much thinner piece of material but which is still able to hold the player’s weight.

26. *CROSS* (J) — Instead of the usual warding off of evil spirits, this ‘CROSS’ can be melted down to form a ‘FLICK-KNIFE’. It is melted in a ‘CRUCIBLE’.

27. *CRUCIBLE* (J) — The ‘CROSS’ can only be melted in this if some ‘SNOW’ has first of all been melted. The melted ‘SNOW’ washes out the ‘CRUCIBLE’ ready for melting down the ‘CROSS’.

28. *CRUXIFIX* (D) — The purpose of this object is one of protection against the powers of Dracula, for without it he is in a position to hypnotise you when you come face to face with him.

29. *CUBE* (J) — I suppose somebody at some time would have to include one of these irritating devices in an adventure, so I have made an irritating feature out of it, since it gives useless information when it is turned.

30. *CUPBOARD* (D) — The ‘KEY’ that the robot gives has its proper use in opening the ‘CUPBOARD’ and not the ‘DOOR’ which is reserved for the ‘SKELETON-KEY’. When the ‘CUPBOARD’ is opened a ‘ROPE’ falls out which the player may pick up and carry. A variety of objects could be put behind cupboard doors in adventures.

31. *DAGGER* (D) — A ‘DAGGER’ is used to kill a member of ‘WOLVES’. It is necessary to be careful here since a false step can be fatal

with all those “WOLVES” around.

32. *DEAD-GUARD* — When a “GUARD” is killed in “DRACULA”, a “DEAD-GUARD” takes its position in the room in which it was killed. It is possible for this “DEAD-GUARD” to be carried around although this is a pointless thing for the adventurer to do.

33. *DIAMOND* (J) — When dusted with a “DUSTER” the “DIAMOND” is a lethal weapon. It will then become possible to kill a “MINOTAUR” which would not let you past it.

34. *DOG* (J) — If you “PAT” this creature then you will be automatically transported into the next room. It makes a change from having to kill things to pass them.

35. *DOOR* (CDJ) — This is the only object which is used in all three adventures showing how popular the “BOOK” is. It is simply an obstacle which is in the adventurer’s way and a method must be found for opening it.

36. *DRACULA* (DJ) — Is present in two adventures; in one he is in a state of activity and in the other he is in a state of activity and in the other he is in a state of reincarnation. It is necessary to find a way of killing him in both cases.

37. *DUSTER* (J) — This is the implement which is required to both “DUST” the rust off a “RUSTY -KEY” and to “DUST” a “DIAMOND” so that it becomes a lethal weapon.

38. *DWARF* (D) — One of the popular creatures of the fantasy world is the “DWARF” which I associate with a nasty little creature who likes grabbing your money. “TROLLS” are larger creatures which use force to relieve you of some cash.

39. *FLICK-KNIFE* (J) — When you have the “FLICK-KNIFE” in your possession the creature against which it is used in “JOURNEY” is the “WEREWOLF” along with using the command “S ”.

40. *FOOD* (J) — Sometimes it is necessary to feed creatures with things like “FOOD” before you can pass them, and there is often provision for the foolhardy adventurer who thinks that the “FOOD” is for his/her own consumption.

41. *FOUNTAIN* (J) — When this is examined, a “BULLET” is revealed and the “FOUNTAIN” vanishes into thin air leaving the player to work out what to do with the “BULLET”.

42. *FUEL-TANK* (J) — If you have a fuel like “OIL” in an adventure then you should also have a place in which the fuel is stored and a method for extracting the fuel. In “JOURNEY”, the “FUEL-TANK” is the storage place for the “OIL”.

43. *GARLIC* (D J) — The “GARLIC” in “DRACULA” is used to render a “VAMPIRE” completely harmless and in “JOURNEY” it frightens away a “COBRA” with its pungent smell.
44. *GATES* (J) — This is something similar to the “WALL” in “JOURNEY” since it is an object which must be climbed over with the “CORD”. However, you could make it have to be opened, like a “DOOR” in your own adventures.
45. *GHOST* (D) — There is only one “GHOST” although there appears to be three. When it is killed in the first room with the “MAGIC SWORD” it goes into the second room and when it is killed there it moves to the third room where it cannot be killed. You cannot pass it when it is in the first and second rooms, but you can pass it when it is in the third room, as long as you are not carrying any of the objects from the “WEAPON ROOM”.
46. *GLASSES* (J) — So that you are able to read the sign in “JOURNEY” you must be wearing the “GLASSES”. If you are not wearing them when you try to read the “SIGN” then it tells you that you have to wear them to do so.
47. *GLOVES* (C) — In “CAPTIVE” for an “ICE-BLOCK” to be picked up the “GLOVES” must be worn. You may have noticed, however, that if you drop the “GLOVES” then the “ICE-BLOCK” is not dropped as well — this is because the main objective is for the player to realise that the “GLOVES” should be worn in the first place.
48. *GLUE* (D) — When this is dropped in the “SKELETON CHAMBER” the “BROKEN-BONES” combine together to form a “SKELETONKEY”. You are not able to drop the “GLUE” in any of the other rooms in the adventure.
49. *GRAVE* (J) — When this is dug up with the “SPADE”, the reincarnating Dracula is revealed along with a “CROSS”, and the only way that the “CROSS” can find its way into your possession, is for Dracula to be killed properly with the silver “BULLET”.
50. *GRENADE* (C) — So that some “BOULDERS” are moved out of the way it is necessary to throw the “GRENADE” at them. The explosion is on impact and the player is able to pass.
51. *GUARD* (DJ) — There is one “GUARD” in “DRACULA” and another in “JOURNEY”. The first is a weakling and can be killed with your bare hands but the other can kill you and the best that you can do with it is be thrown into an adjacent room.
52. *GUN* (D)— The purpose of the “GUN” in “DRACULA” is to frighten the robot into giving you a “KEY”. If you enter the “DINING ROOM” without the “GUN” then the robot will kill you.
53. *HARPOON* (J) — This is part of a large red herring along with the

“ANTEATER”, “TERMITES” and “WHALE”. For if you decide to throw the “HARPOON” at the “WHALE” then you will be killed.

54. *HEADPHONES* (C) — If you throw the “GRENADE” at the “BOULDERS” and are not wearing the “HEADPHONES” then the noise from the blast will be sufficient to burst your ear drums — the shock of this will kill you.

55. *HOLE* (C) — This is a red herring as the player is tempted to find out more about it and in examining it something large falls out and flattens the adventurer.

56. *ICE-BLOCK* (C) — From the message that the transmitter is overheating the adventurer should deduce that something is required to cool it down with, and on finding the “ICE-BLOCK” the two things will be linked together. The problem about this object is that some “GLOVES” must be worn for it to be picked up.

57. *INSCRIPTION* (C) — When this is read it will reveal someone which should be useful to the adventurer, but the catch is that a “MAGNIFIER” must be in the possession of the player before the “INSCRIPTION” may be read.

58. *JEWEL* (D) — The “JEWEL” is the object in “DRACULA” that the player has to recover — without having recovered this, player will have failed in the mission. In the sequel, “JOURNEY TO FREEDOM”, the “JEWEL” is still in the possession of the adventurer, but there is no need to print it out under “INVENTOR Y” since it is not manipulated in any way in this adventure.

59. *KEY* (D J) — The “KEY” in “DRACULA” is required to open a “CUPBOARD” and hence reveal a “ROPE”, the “KEY” in “JOURNEY” simply opens a door which is in the way.

66. *KEY-CUTTER* (C) — The purpose of this is to “CUT” from a piece of “ROUGH-METAL” a “SHINY-KEY”. Note that the “ROUGHMETAL” could not be cut into a single “KEY” since the object “KEYCUTTER” already starts with the three letters of “KEY”.

61. *LARGE TREE* (J) — This concerns part of the red herring about the “CUBE”, for, when this is examined the “CUBE” is revealed to the player and printed under “OBJECTS”.

62. *LASER-GUN* (D) — As you may discover from a particular message, laser guns are able to “BLAST” sealed exits. In other words they clear doors and the like out of the way so that the player can pass.

63. *LEVER* (J) — Here is another example of temptation for the player because the natural thing to try first of all is pulling the “LEVER” — this unfortunately alerts Dracula’s guards to your whereabouts.

64. *LION* (J) — A “LION” is an animal that tends to eat a lot when it is ,

hungry, so it is quite possible that if it is fed with the ‘FOOD’ that you have brought along then it will go away. As well as this you no longer fall down a pit in the location labelled ‘LION PITS’.

65. *LIT-CANDLES* (D) — Some help is given in the adventure as to the purpose of these objects: a message reveals that ‘W olves fear lit candles’. This means that as long as you are carrying the ‘LIT-CANDLES’, then you will not be killed by ‘WOLVES’ when you enter the ‘Room of Cages’.

66. *LIT-TAPER* (J) — If you are carrying this object in the ‘Dimly lit part of cave’ then you will not be killed, since the ‘LIT-TAPER’ acts as a light which lights your way in this section and therefore keeps you safe.

67. *LOCKED-DOOR* (C) — In an adventure there is not much of a difference between a ‘DOOR’ and a ‘LOCKED DOOR’ because they both have to be opened in one way or another. The ‘LOCKED-DOOR’ points towards the necessity of a ‘KEY’ of some sort for it to be opened.

68. *MAGNIFIER* (C) — So that the ‘INSCRIPTION’ may be read, this is required. Without it, the computer prints out that the writing is too small to read.

69. *MALLET* (D) — In ‘DRACULA’ this is used to ‘HAMMER’ the ‘STAKE’ into ‘DRACULA’ and the ‘VAMPIRE’. Without it ‘I have nothing to hammer with’ will be printed out.

70. *MATCHES* (J) — As may be expected these are needed so that something may be lit. The object that the ‘MATCHES’ may ‘LIGHT’ is the ‘TAPER’ which then becomes a ‘LIT-TAPER’ (the ‘TAPER’ is taken out of the game, and the ‘LIT-TAPER’ is brought into the game).

71. *MEAT-CHOPPER* (D) — This is used to ‘CHOP’ the ‘VAMPIRE’ and stop it recovering from having a ‘STAKE’ hammered into it. The dropping of the ‘GARLIC’ also helps to stop it from recovering.

72. *MINOTAUR* (J) — To kill this monster the command ‘KILL’ is used along with the possession of a ‘DIAMOND’ which has been dusted with the ‘DUSTER’. When you kill it, you are moved into an adjacent room.

73. *MIRROR* (D) — When you examine this object you will be told how many moves you have left to make before your time runs out — it prints out the number which is ‘W’less than 24 5.

74. *MUD-MAN* (C) — This is another creature that has to be killed so that you can pass it. The requirements for doing so are a ‘SABRE’ and the command ‘KILL’.

75. *MUSIC* (D) — If you come across some ‘MUSIC’ and are reasonably musically adept, then you would naturally try and ‘PLAY’ the ‘MUSIC’. The result of doing so transports you into another room which is hidden beside the ‘MUSIC ROOM’.

76. *OIL* (J) — The “OIL” is obtained from the “FUEL TANK” and put in some “CANS”. The “OIL” is then carried in the “CANS” until the “CHAIN-SAW” is filled with it. The “CHAIN-SAW” will then become operational.

77. *PARCHMENT* (J) — It is common to give the adventurer some sort of useful hint to begin with and this is given in the “PARCHMENT” which may be read.

78. *PLANT* (D) — This is a creature, if it can be called that, that can be passed quite easily, for all that need be typed in is “KICK PLANT” and the “PLANT” moves aside (you will surely not let things like plants defeat you).

79. *POST* (J) — To pick up the “POST” it is necessary to firstly “DIG” it out of the ground and then type in “GET POST”. This object is used to fend off an annoying “BEAR” by simply hitting it when you have the “POST” in your possession.

80. *RECORDER* (D) — When you “PLAY” the “RECORDER” you will receive some useful information about the “WOLVES” and “LIT-CANDLES” and a “LASER-GUN” and sealed exits.

81. *ROPE* (D) — This is usually used for climbing as in “DRACULA” where it must be tied and lowered, first of all; however, it could also be used for typing up the likes of a “GUARD”, although a “CORD” would be more suited to this purpose.

82. *ROUGH METAL* (C) — So that you can make a “SHINY KEY” in “CAPTIVE”, you must pick up the “ROUGH-METAL” and “CUT” it in the same room as the “KEY-CUTTER”.

83. *RUBBER-BOOTS* (D) — These must be worn for the desired effect to be obtained. Since they are made of rubber they tend to have a better grip and so the adventurer will not slip and fall down the pit in the “Room with a Pit”.

84. *RUSTY-KEY* (J) — If the player tries to open the door in “JOURNEY” then he/she will find out that it does not fit the lock. This is because it is necessary to “DUST” it with the “DUSTER” and thus remove the rust. The “RUSTY-KEY” is removed from the game and the “KEY” is brought into it in the player’s “INVENTORY”—the key fits the lock.

85. *SABRE* (C) — If this is not carried and the adventurer enters the same room that the “MUD-MAN” is in, then death results. If the “SABRE” is carried and this monster is in the same room as the player, then movement will not be allowed until the “MUD-MAN” has been killed with the “SABRE”.

86. *SAD-DRAGON* (D) — I have given the dragon the attribute of being sad since it would otherwise begin with the first three letters of

- “DRACULA”. When confronted, the dragon will now move, but when a “SERPENT” is dropped in the same room, the “SERPENT” kills the “SAD-DRAGON” and you can pass.
87. *SAND* (J) — Although it is possible to survive if you happen to be in the location labelled “SINKING SAND”, if you also decide that you want to pick up the “SAND” there, then you sink with the sinking sand.
88. *SCORPION* (J) — If you enter into the same room as the “SCORPION” and by the end of one move you have not disposed of it by eating it, then it will sting you to death.
89. *SCRATCHES* (C) — Instead of the computer printing out that writing can be seen in every adventure you make, it is better to vary the words you use to detail that writing is there to be read. “SCRATCHES” is one such variance.
9. *SEALED-EXITS* (D) — As they may sound, “SEALED-EXITS” are those exits which cannot be passed unless whatever that is sealing them is broken. If you “BLAST” at them with a “LASER-GUN” then this seal will be broken.
91. *SERPENT* (D) — When you are confronted with the “SERPENT” you will be surprised to realise that you can pick it up. By dropping it again in the “Room with a dragon” the “SAD-DRAGON” is killed by it and you can pass into the next room.
92. *SHIELD* (D) — If you enter the “Torture Chamber” and are not protected by a “SHIELD” then some Laser Beams fired at you, kill you. However, if you are carrying the “SHIELD”, the Laser Beams are deflected.
93. *SHINY-KEY* (C) — This is formed by cutting the “ROUGHMETAL” with the “KEY-CUTTER”. The “SHINY-KEY” is used to open a “LOCKED DOOR” in the Air Lock.
94. *SHOTGUN* (J) — If you use this along with the “BULLET” which you may find, you can “SHOOT DRACULA” when you see him again, and if your luck is with you, he will die.
95. *SIGN* (J) — When you have persuaded the “BEAR” to move out of the way, then you can pick up the “SIGN”. You will only be able to read the information on it provided that you are wearing the “GLASSES”.
96. *SILVER-COIN* (D) — If you have thought that you had completed “Dracula” but died at the hands of the “DWARF” who went on about some sort of courage, you could not have picked up the “SILVER COIN” — this is the payment to it which you must make in order to pass it.

97. *SKELETON-KEY* (D) — This is formed by dropping the ‘GLUE’ in the ‘Skeleton Chamber’ where the ‘BROKEN BONES’ are to be found. Its use is to ‘OPEN’ the ‘DOOR’ off the ‘E. End of the Corridor’.

98. *SNOW* (J) — For the ‘CROSS’ to be melted in the ‘CRUCIBLE’, the ‘CRUCIBLE’ must be clean and to clean it the ‘SNOW’ is melted in it first of all.

99. *SPADE* (J) — To obtain this, some trees have to be removed with a ‘CHAIN-SAW’ which is filled with ‘OIL’. The ‘SPADE’ is used to ‘DIG’ up a ‘POST’ as well as to ‘DIG’ up a ‘GRAVE’. The latter use helps you to uncover the reincarnating ‘DRACULA’.

100. *SPEAR* (J) — When the ‘WEREWOLF’ is met in the ‘Rocky part of a route’ the ‘SPEAR’ must be used against it and it runs away injured — it is not possible to kill this creature until it is met for a second time.

101. *SPIDERS-WEB* (D) — If the player enters the ‘Room with a Giant Spider’ then he/she will be caught up in the ‘SPIDERS WEB’. The way to free yourself is to ‘DROP’ the ‘CONTAINER’ which you should be carrying full of ‘ACID’—the ‘ACID’ dissolves the web.

102. *STAKE* (D) — This is hit with the ‘MALLET’ in two instances; in the first case it helps to kill the ‘VAMPIRE’ and in the other it aids the destruction of ‘DRACULA’.

103. *SWARCK* (C) — If you read the ‘INSCRIPTION’ when you have the ‘MAGNIFIER’ in your possession then you will be told that it is a magic word. When you are in the location ‘Outside of ship’ and ‘SAY’ this word, you are transported into the ship which flies off with you in it.

104. *SWITCH* (D) — When you ‘PRESS’ this in ‘Dracula’ the lights are shut down and this gives ‘DRACULA’ the chance to locate you and kill you. This object can have a variety of other uses which the programmer may like to think up.

105. *SWORD* (D) — This allows the adventurer to ‘KILL’ the ‘GHOST’ in the ‘Dim room’ and the ‘GHOST’ in the ‘Dull room’ but the ‘GHOST’ in the ‘Dark room’ cannot be killed and the ‘SWORD’ and any other weapon from the ‘WEAPON ROOM’ must be dropped elsewhere before this ‘GHOST’ can be passed without any problems.

106. *TAPER* (J) — This can be lit with the help of some ‘MATCHES’. When it is lit it is removed from the game and the ‘LIT TAPER’ is brought into it.

107. *TERMITES* (J) — The function of these small creatures is to cover up the ‘HARPOON’. If the ‘ANTEATER’ is dropped on top of them then they will be eaten and the ‘HARPOON’ will be revealed.

108. *TORCH* (C) — When the adventurer passes through the “Dimly lit passage” the “TORCH” must be carried and also be lit or else a hole will be fallen into.

109. *TRANSMITTER* (C) — For this to work, the “ICE-BLOCK” and the “AERIAL” must be dropped in the “Signal Transmitting room”. It will then be possible to “TRANSMIT” a signal which will reveal an entrance into the “Air Lock”.

110. *TREES* (J) — When the player cuts these with the oil filled “CHAIN SAW” sufficient area is cleared for a “SPADE” to be seen — this can then be picked up and taken away.

111. *UNDERGROWTH* (J) — When the player examines the “UNDERGROWTH” a “GRAVE” can be seen which must be dug up so that “DRACULA” may be killed properly.

112. *VAMPIRE* (D) — This must be killed in “DRACULA”. To do so the coffin must first of all be lifted. The “STAKE” is then hammered into it with the “MALLET”. After this, the “MEAT CHOPPER” is used on it and finally the “GARLIC” is dropped.

113. *WALL* (J) — The adventurer must “CLIMB” over this. Some help is given if it is examined, about it being climbable on the condition that the “CORD” is used to “CLIMB” it with.

114. *WALLET* (D) — The purpose of this is to help to keep the “JEWEL” safe once it has been found, because if this is not carried then the gem may fall into unrecoverable places.

115. *WEREWOLF* (D) — This is a suitable creature to have, since it does not tend to like people living too much and so it is a form of opposition against you. In “Dracula” you can be killed but in “Journey” you can revenge this previous encounter.

116. *WHALE* (J) — If you want then you can try throwing the “HARPOON” at the “WHALE” when you come across it, provided that you are carrying the “HARPOON”, but the result will be your death.

117. *WINDOW* (C) — This allows the player to gain a little more information about his/her surroundings — if it is examined details are given about your spacecraft.

118. *WINE-LABEL* (D) — In “Dracula”, when you try to “MIX” the “CHEMICALS” you are asked for a formula. This formula is obtained by reading the “WINE-LABEL” and remembering the details given — when typing out the formula, it must be typed out in full to be accepted.

119. *WIRE* (C) — When you cool down the “TRANSMITTER” in “Captive” this object disappears from the game and the “AERIAL” takes its

place. As the transmitter improves in working order the “WIRE” can be distinguished as actually being an “AERIAL”.

120. *WOLVES* (D) — If you enter the “Room of Cages” and are not carrying the “LIT-CANDLES” then these animals will kill you. As well as the candles to protect yourself, you will need the “DAGGER” to kill them.

121. *WRITING* (D) — This forms a helpful message near the beginning of the adventure “Dracula’s Castle” which the player has the option of reading if he/she considers it worthwhile.

This is a reference section which allows the reader to look up any object out of all those in the three adventures of this book, in its alphabetical order, and see what its function is and its influence on the game. The only object that is in all three adventures is the “DOOR”; you may have observed that most of the objects are unique to the adventure that they are part of. Although virtually a completely new set of objects is required for each new adventure, you should be able to use this section as a rough guideline for the objects that you will come to use in your own adventures.

APPENDIX III

HOW TO SOLVE THE ADVENTURES

The routines mentioned for the commands and objects in appendices I and II, and any other adventure routines in the chapters, are brought together here in possible solutions to the three adventures. If you have completed them, then you may be interested in how the programmer would complete them, or if you have not made an attempt at them, then you may wish to see the adventures completed in practice and gain information from this to make your own. Each command made is given along with remarks corresponding to what is printed on the screen — after directions, the new room name is printed out; the messages which relate to the number of moves made are printed only when each new message appears on the screen. Any remarks give are put inside brackets after this.

“Captive”

Starting Room — Prison Cell

1. *“KICK-DOOR”* — The hinges were weak and the door has collapsed into a pile of dust. (The first obstacle has been removed from the adventurer’s path. Now for some exploration).
2. *“N”* — Bell tower. (Remember not to *“RING”* the *“BELL”*).
3. *“W”* — Winding staircase (A *“TORCH”* is listed under *“OBJECTS”*).
4. *“GET TORCH”* — O.K. (How about lighting it).
5. *“LIGHT TORCH”* — It is now lit. (Some more exploration now).
6. *“N”* — Gunpowder chamber (Leave the *“GRENADE”* till later).
7. *“E”* — Place with a rocky floor. (Some *“GLOVES”* are here).
8. *“GET GLOVES”* — O.K. (How about wearing them).
9. *“WEAR GLOVES”* — O.K. (Move on to next room).
10. *“N”* — Signal transmitter room. The transmitter is overheating. (Search for something which can cool the *“TRANSMITTER”*).
11. *“E”* — Room of chains (Leave the *“ROUGH METAL”* for the moment).
12. *“N”* — Area with a hole in the ceiling (Do not *“EXAMINE”* the *“HOLE”* unless you like being killed).

13. “N”—Altar (A “SABRE” is here —it could be used as a weapon).
14. “*GET SABRE*”—O.K. (Now you have something to fight with).
15. “S”—Area with a hole in the ceiling.
16. “S” — Room of chains (How about picking up the “ROUGH METAL” now —it could be useful).
17. “*GET ROUGH METAL*” O.K. —(Now for a fight with something).
18. “S” — Muddy Area. (The “MUD-MAN” here does not look as though he is going to live much longer with you standing around brandishing a “SABRE”).
19. “*KILL MUD-MAN*” — You have killed the mud-man (Surprise, surprise. There is now no further need for the “SABRE”).
20. “*DROP SABRE*” — O.K. A rumbling sound can be heard (Perhaps there is something to cool the “TRANSMITTER” in one of the rooms off this room).
21. “S” — Locksmiths (The “KEY -CUTTER” looks as though it could be useful).
22. “*CUT ROUGH METAL*” — The piece of metal has been cut into a key (A “LOCKED-DOOR” can now be opened if you can find it.)
23. “N”—Muddy area (there is one more room off this room).
24. “E” — Frozen room (An “ICE-BLOCK” is here — this should cool the “TRANSMITTER”).
25. “*GET ICE BLOCK*” — O.K. — (The “GLOVES” are no longer required now that the “ICE-BLOCK” has been picked up).
26. “*DROP GLOVES*” — O.K. (Now you should take the “ICEBLOCK” along to the “Signal transmitter room” to see if it has any effect).
27. “W”—Muddy Area.
28. “N”—Room of chains 29. “W” — Signal transmitter room. The transmitter is overheating (Try dropping the ice here).
30. “*DROP ICE BLOCK*” — O.K. The transmitter has cooled down, but it does not have an aerial (Go and look for the “AERIAL”).
31. “E”—Room of chains.
32. “N”—Area with a hole in the ceiling.
33. “W”—Dimly lit passage.
34. “W”—Brightly coloured room.
35. “W”—Repairs room (The “AERIAL” is here).

36. “*GET AERIAL*” — O.K. (How about taking it back to the “TRANSMITTER”).
37. “*E*” — Brightly coloured room.
38. “*E*” — Dimly lit passage.
39. “*E*” — Area with a hole in the ceiling. The noise is becoming louder.
40. “*S*” — Room of chains.
41. “*W*” — Signal Transmitter room (‘Drop’ the ‘Aerial’ like you did with the ‘ICE BLOCK’).
42. “*DROP AERIAL*” — O.K. The transmitter is fully operational (Try transmitting a signal now and see what the effect is).
43. “*TRANSMIT*” — An entrance has appeared into the “air lock”. The transmitter is fully operational (Look for other objects which you may require to aid your escape via the “Air lock”)
44. “*W*” — Padded cell. (Some ‘HEADPHONES’ are here).
45. “*GET HEADPHONES*” — O.K. (How about wearing them).
46. “*WEAR HEADPHONES*” — O.K. (You still have not picked up the ‘GRENADE’ so go back and ‘GET’ it).
47. “*E*” — Signal transmitter room. The transmitter is fully operational.
48. “*S*” — Place with a rocky floor.
49. “*W*” — Gunpowder chamber. (The ‘GRENADE’ is here).
50. “*GET GRENADE*” — O.K. (Now to find the “Air lock” and work out how to escape).
51. “*E*” — Place with a rocky floor.
52. “*N*” — Signal transmitter room. The transmitter is fully operational.
53. “*E*” — Room of chains.
54. “*N*” — Area with a hole in the ceiling.
55. “*W*” — Dimly lit passage.
56. “*W*” — Brightly coloured room.
57. “*N*” — Observation point.
58. “*E*” — Air lock (The ‘LOCKED DOOR’ in an obstacle which is in the way and must be removed — try using the ‘SHINY KEY’).
59. “*OPEN LOCKED DOOR*” — The door came away in your hands but the exit is now blocked by boulders which had been behind the door.

The ground is starting to shake (How about using the GRENADE" to remove the "BOULDERS" from your way.)

60. "THROW GRENADE" — You have cleared a passage through the boulders. (Now you are able to pass through into the next room).

61. "N" — Outside of ship. (To enter the ship try saying the magic word "SWARCK" which you discovered on earlier attempts at the adventure).

62. "SAY SWARCK" — You have materialised inside your ship which has immediately taken off.

Score —58 Best Score —58
Press space to start again.

"Dracula's Castle"

Starting room — Cell

1. "KILL GUARD" — O.K. (the first problem is completed quite easily).

2. "N"—Corridor (Searching for something of interest).

3. "W"—W. End of Corridor (A "MALLET" is here).

4. "GET MALLET" — O.K. (This should be moved to a more accessible place where it can be picked up again later).

5. "N"—Library (A "GUN" is here).

6. "GET GUN" — O.K. (This is needed for protection against the Robot).

7. "W"—Study (A "WALLET" is here).

8. "GET WALLET" — O.K. (When you pick up the "JEWEL" this helps you to keep it safe).

9. "N" — Computer room. A computer asks if you think that this program is good (answer Y or N). (There is not much option about what is typed in, since only "Y" for yes is accepted).

10. "Y"—(No remarks are given).

11. "W"—Candlelit room (There are some "LIT CANDLES" here).

12. "GET LIT CANDLES" — O.K. (these, like the "MALLET", are being moved to a more convenient point for use later in the adventure).

13. "E" — Computer room. A computer asks if you think that this program is good (Answer Y or N).

14. "Y"—(No remarks)

15. "E"—Living Room (A "CONTAINER" is here).

16. “*GET CONTAINER*” — O.K. (This will be used for carrying some “ACID”).
17. “*E*” — S. End of main hall.
18. “*S*” — Dining Room. A robot, seeing your gun, dropped a key. (Why not pick up the “KEY”.)
19. “*GET KEY*” — O.K. (This will “OPEN” a “CUPBOARD”).
20. “*N*” — S. End of main hall (There is not any more use for the “GUN”).
21. “*DROP GUN*” — O.K. (this is also a suitable place to “DROP” the “MALLET”).
22. “*DROP MALLET*” — O.K.
23. “*N*” — Main hall.
24. “*N*” — N. end of main hall. (The “LIT CANDLES” can be dropped here).
25. “*DROP LIT-CANDLES*” — O.K.
26. “*E*” — Room with a pool.
27. “*N*” — Room with a clock
28. “*E*” — Kitchen (A “BUCKET” is here).
29. “*GET BUCKET*” — O.K. (The “MEAT CHOPPER” can be moved to a better location).
30. “*GET MEAT CHOPPER*” — O.K. (There is now no further need to enter the “KITCHEN”).
31. “*W*” — Room with a clock. (I wonder what the time is).
32. “*EXAMINE CLOCK*” — It is 6 o’clock (This will vary randomly between one and six every time you “EXAMINE” the “CLOCK”, so if the time is not after three, then “EXAMINE” it until it is, or else you will be at the mercy of the “WEREWOLF”).
33. “*S*” — Room with a pool.
34. “*W*” — N. end of hall, Dracula has detected your presence.
35. “*S*” — Main hall (This seems a good place to “DROP” the “MEAT CHOPPER”).
36. “*DROP MEAT CHOPPER*” — O.K. (Now to go into the Section of the Castle in which the “JEWEL” is hidden).
37. “*E*” — Chamber of horror.
38. “*E*” — Armour room.

39. “*N*”—Bright room (A nasty looking ‘PLANT’ is in your way).
40. “*KICK PLANT*”—O.K. (The ‘PLANT’ temporarily disappears).
41. “*E*”—Room of mirrors.
42. “*N*”—Echo Chamber
43. “*E*” — Room with a serpent. (The ‘SERPENT’ is friendlier than you might expect).
44. “*GET SERPENT*”—O.K.
45. “*S*” — Room with a fire in it (If you examined the ‘BUCKET’ in previous attempts then you would know that it contained ‘WATER’).
46. “*EMPTY BUCKET*” — A passage has been cleared through the flames. Your bucket has melted. (Another obstacle has been removed from your path.)
47. “*S*”—Room with a dragon. (Serpents do not like dragons).
48. “*DROP SERPENT*”—The serpent has killed the dragon.
49. “*S*” — Treasure chamber. (At last you have found the ‘JEWEL’. but that is not nearly the end of the adventure).
50. “*GET JEWEL*” — O.K. (Now to go back to a more central location and try and work out how to escape.).
51. “*N*”—Room with a dragon.
52. “*N*”—Room with a fire in it.
53. “*N*”—Room with a serpent.
54. “*W*”—Echo Chamber.
55. “*S*”—Room of mirrors.
56. “*W*”—Bright room (It is that terrible ‘PLANT’ again).
57. “*KICK PLANT*”—O.K.
58. “*S*” — Armour room (A ‘SHIELD’ is here-this could be used as a form of protection).
59. “*GET SHIELD*”—O.K.
60. “*W*” — Chamber of horror. (Why not fill your ‘CONTAINER’ with ‘ACID’ now).
61. “*GET ACID*”—O.K.
62. “*S*” — Werewolf’s chamber. (You might as well leave the ‘JEWEL’ in a relatively safe place for the time being).
63. “*DROP WALLET*”—O.K.

64. *"DROP JEWEL"*—O.K.
65. *"S"*—Weapon room. (You have a choice of three weapons).
66. *"GET SWORD"*—O.K.
67. *"S"* — Torture Chamber. Your shield deflects some laser beams. (So you did need your *"SHIELD"*, otherwise there would not have been much left of you).
68. *"E"*—Workroom. (Leave the *"STAKE"* till on the way back).
69. *"E"*—Laboratory. Dracula is out to destroy you.
70. *"E"* — Room with a giant spider. (The *"SPIDERS WEB"* is in the way).
71. *"DROP CONTAINER"* — O.K. Your acid has dissolved the web. (You are now able to pass into the next room).
72. *"N"*—Dull room. (A *"GHOST"* is here).
73. *"KILL GHOST"*—O.K. (Is a *"GHOST"* alive in the first place?)
74. *"W"* — Dim room. (Another *"GHOST"* is here — or is it the same one?)
75. *"KILL GHOST"*—O.K.
76. *"DROP SWORD"* — O.K. (You needed to drop your weapon to pass the third *"GHOST"*).
77. *"N"*—Dark room. (You cannot *"KILL"* this *"GHOST"*).
78. *"W"* — Room which is pitch black. (It is a wonder you can see the *"CUPBOARD"*).
79. *"OPEN CUPBOARD"*—O.K. (A *"ROPE"* is inside it).
80. *"GET ROPE"*—O.K.
81. *"DROP KEY"*—O.K. (There is no further use for this *"KEY"*).
82. *"E"*—Dark room.
83. *"S"*—Dim room
84. *"E"*—Dull room
85. *"S"*—Room with a giant spider.
86. *"W"*—Laboratory. (Some *"CHEMICALS"* are here).
87. *"GET CHEMICALS"*—O.K. (How about mixing them).
88. *"MIX CHEMICALS"* — With what formula? (In the previous attempts you may have *"READ"* the *"WINE LABEL"*).

89. “*X*Y+Z-5=W/V*” —A glue has been formed.
90. “*W*” —Workroom. (The ‘*STAKE*’ will come in useful).
91. “*GET STAKE*” —O.K.
92. “*W*” —Torture Chamber. Your shield deflects some laser beams.
93. “*N*” —Weapon room.
94. “*DROP SHIELD*” — O .K. (There is no more need for it since you will not be going back that way again).
95. “*E*” —Skeleton chamber.
96. “*DROP GLUE*” — O.K. A ‘*SKELETON KEY*’ has been formed. (You will be able to ‘*OPEN*’ the locked ‘*DOOR*’ with this).
97. “*GET SKELETON KEY*” —O.K.
98. “*W*” — Weapon room. (The two remaining weapons are to be picked up now since they will be of use in the next section).
99. “*GET DAGGER*” —O.K.
100. “*GET AXE*” —O.K.
101. “*N*” —Werewolf’s chamber.
102. “*W*” —S. End of main hall.
103. “*DROP STAKE*” —O.K. (This will be used later).
104. “*DROP SKELETON KEY*” —O.K. I’d advise you to hurry up.
105. “*DROP ROPE*” —O.K. 106. ‘*N*’ —Main Hall.
107. “*N*” — End of main hall. (The ‘*LIT CANDLES*’ are here — they are required for the section you are about to encounter).
108. “*GET LIT CANDLES*” —O.K.
109. “*N*” —Music Room. (Some ‘*MUSIC*’ is here).
110. “*PLAY MUSIC*” — Room with soldier ants. You have been thrown through a door in the West wall. (Some ‘*RUBBER BOOTS*’ are here).
111. “*GET RUBBER BOOTS*” —O.K.
112. “*WEAR RUBBER BOOTS*” —O.K.
113. “*W*” —Room with cages. Your lit candles frighten the wolves.
114. “*KILL WOLVES*” — O.K. (That was not much problem with the ‘*DAGGER*’ in your possession.
115. “*W*” —Servants’ quarters. (A ‘*CRUCIFIX*’ is here).
116. “*GET CRUCIFIX*” —O.K.

117. “S” — Chamber of fear. (Leave the ‘LASER GUN’ for the moment).

118. “S” — Room with a clue. (Some ‘GARLIC’ is here, but it is attached firmly).

119. “*CUT GARLIC*” — O.K. (Your “AXE” was small enough to be suitably handy to cut it with).

120. “*GET GARLIC*”—O.K.

121. “*DROP AXE*”—O.K. (It is no longer needed).

122. “N”—Chamber of fear. (Now “GET” the ‘LASER GUN’).

123. “*GET LASER GUN*”—O.K.

124. “N”—Servants’ quarters.

125. “E”—Room with cages.

126. “S”—Room with a sarcophagus. (A ‘SILVER COIN’ is here).

127. “*DROP LIT CANDLES*” — O.K. (These are no longer required and you needed to ‘DROP’ something so that the ‘SILVER COIN’ may be picked up).

128. “*GET SILVER COIN*”—O.K.

129. “S” — Burial Chamber. (This room had a function in the original ‘Dracula’s Castle’, but under the circumstances of this version, it only serves as a link room).

130. “E”—Chamber of suspense.

131. “*BLAST SEALED EXITS*” — O.K. (Two more exits have been revealed — you must not go North since you would be killed in the ‘Chamber of Horror’).

132. “E”—Main Hall.

133. “*DROP LASER GUN*”—O.K. (It has served its purpose).

134. “*DROP DAGGER*”—O.K.

135. “*GET MEAT CHOPPER*” — O.K. (You need this for the next section).

136. “S”—S. End of main hall.

137. “*DROP SILVER COIN*” — O.K. (This is a suitable place to leave it).

138. “*GET STAKE*”—O.K.

139. “*GET MALLET*” — O.K. You do not have much time left. (You need the ‘MALLET’ to ‘HAMMER’ the ‘STAKE’).

140. "W"—Living room.
141. "W" — Computer Room. A computer asks if you think that this program is good. (Answer Y or N).
142. "Y"—(No remarks).
143. "W"—Candlelit room.
144. "S"—Wine room.
145. "S" — Room with a pit. Your rubber boots give you a grip on the slippery surface.
146. "E"—Dracula's bedroom. (A "COFFIN" is here).
147. "*LIFT COFFIN*"—O.K. (A "V AMPIRE" has appeared).
148. "*HAMMER STAKE*" — The vampire is dead but your mallet has disappeared.
149. "*CHOP VAMPIRE*"—O.K.
150. "*DROP GARLIC*"—O.K. (This prevents you from being killed).
151. "*DROP MEA T CHOPPER*"—O.K. (You do not need it anymore).
152. "*GET STAKE*"—O.K. (You still need this).
153. "W" — Room with a pit. Your rubber boots give you a grip on the slippery surface.
154. "N"—Wine room.
155. "N"—Candlelit room.
156. "E" — Computer room. A computer asks if you think that this program is good. (Answer Y or N).
157. "Y"—(No remarks).
158. "E"—Living room.
159. "E"—S. End of main hall.
160. "*DROP RUBBER BOOTS*" — O.K. (They have served their purpose).
161. "E" — Werewolf's chamber. (You must remember that you have to leave the castle with the "JEWEL").
162. "*GET JEWEL*" — O.K. (At this stage the "JEWEL" is fairly safe without the "WALLET").
163. "W"—S. End of main hall.
164. "*GET GUN*"—O.K. (You need to pass the robot again).

3. “*GET MATCHES*” —O.K.
4. “*E*” —Cloakroom. (Leave the ‘BOOTS’ till later).
5. “*E*” — W. End of corridor. (Searching for a suitable implement with which a ‘GUARD’ can be passed).
6. “*E*” — Corridor. (The ‘GUARDS QUARTERS’ lie to the ‘SOUTH’).
7. “*E*” —E. End of corridor.
8. “*N*” —Food room. (Leave the ‘FOOD’ till later).
9. “*W*” — Store room. (A ‘BOXING GLOVE’ is here — this should be useful against the ‘GUARD’).
10. “*GET BOXING GLOVE*” —O.K. (It must be worn).
11. “*WEAR BOXING GLOVE*” —O.K. (Now to attack the ‘GUARD’).
12. “*E*” —Food room.
13. “*S*” —E. End of corridor.
14. “*W*” —Corridor
15. “*S*” —Guard’s quarters. (The ‘GUARD’ is here).
16. “*PUNCH GUARD*” — Fuel room. You have been thrown by the guard into an adjacent room. (That was not too great, but anyway you are in the next room —some cans are here).
17. “*GET CANS*” —O.K.
18. “*S*” —Money room. (A ‘CHEST’ is here).
19. “*GET CHEST*” —O.K.
20. “*N*” —Fuel room.
21. “*W*” — Guard’s quarters. (It might be a good idea to ‘GET’ the ‘TAPER’).
22. “*GET TAPER*” —O.K.
23. “*N*” —Corridor.
24. “*LIGHT TAPER*” —O.K. (You no longer need the ‘MATCHES’).
25. “*DROP MATCHES*” — O.K. (The ‘BOXING GLOVE’ has also served its purpose).
26. “*DROP BOXING GLOVE*” —O.K.
27. “*E*” —E. End of corridor.
28. “*N*” —Food room. (You can take the ‘FOOD’ now).

29. “*GET FOOD*”—O.K.
30. “*W*” — Store room. (You need to move the “*DUSTER*” to a more accessible place).
31. “*GET DUSTER*”—O.K.
32. “*W*”—Supply room. (You can store the “*OIL*” in your “*CANS*”).
33. “*GET OIL*”—O.K.
34. “*E*”—Store room. Guards have discovered Dracula’s death.
35. “*E*”—Food room.
36. “*S*”—E. End of corridor.
37. “*W*”—Corridor.
38. “*W*”—W. End of corridor.
39. “*W*” — Cloakroom. (You are carrying too many objects, but the “*BOOTS*” are required in preference to the “*DUSTER*”).
40. “*DROP DUSTER*”—O.K.
41. “*GET BOOTS*”—O.K. (They must be worn).
42. “*WEAR BOOTS*”—O.K.
43. “*W*”—Entrance to outhouse.
44. “*N*”—Part of path.
45. “*W*” — Woodcutter’s hut. (The “*CHEST*” must be dropped so that the “*CHAIN SAW*” may be picked up).
46. “*DROP CHEST*”—O.K.
47. “*GET CHAIN SAW*” — O.K. (The “*OIL*” has to be transferred into the “*CHAIN SAW*” so that it will work; if this is done the “*CANS*” can be dropped and the “*CHEST*” can therefore be picked up again).
48. “*FILL CHAIN SAW*”—O.K.
49. “*DROP CANS*”—O.K.
50. “*GET CHEST*”—O.K.
51. “*S*” — Path beside a river. (To pass the “*DOG*” you must show some kindness towards it).
52. “*PAT DOG*”—Glade. The dog lets you past it.
53. “*S*”—Bank of a river.
54. “*THROW CHEST*” — Swamp in forest. The chest acts as a platform across the river. (Your “*BOOTS*” protect you in the “*SWAMP*”).

55. “*N*” —Dense part of forest.
56. “*CUT TREES*” —See what has been revealed.
57. “*DROP CHAIN SAW*” —O.K. (It is no longer required).
58. “*GET SPADE*” —O.K.
59. “*S*” —Swamp in forest.
60. “*E*” —Bank of river.
61. “*N*” — Glade. (You have to go back and pick up the ‘*DUSTER*’ how that you are able to carry it).
62. “*N*” —Path beside a river.
63. “*N*” —Woodcutter’ hut.
64. “*E*” —Part of a path.
65. “*S*” —Entrance to outhouse.
66. “*E*” —Cloakroom. (The ‘*DUSTER*’ is here).
67. “*GET DUSTER*” —O.K.
68. “*W*” — Entrance to outhouse. (Heading back to pursue your progress across the river).
69. “*N*” —Part of a path. They have now been alerted to find you.
70. “*W*” —Woodcutter’s hut.
71. “*S*” —Path beside a river.
72. “*PAT DOG*” —Glade. The dog lets you past it.
73. “*S*” —Bank of river.
74. “*GET CHEST*” — O.K. (You need to pick it up and ‘*THROW*’ it again).
75. “*THROW CHEST*” — Swamp in forest. The chest acts as a platform across the river.
76. “*N*” —Dense part of the forest.
77. “*N*” —Cleared area of forest.
78. “*W*” —Large tree in forest.
79. “*S*” — Thin forestry. (I’d advise you to feed the ‘*LION*’ with the ‘*FOOD*’ that you have brought along for that purpose).
80. “*FEED LION*” —The lion disappeared after eating your food.
81. “*N*” —Large tree in forest.

82. “E” —Cleared area of forest.
83. “W” — Lion pits. (Now that the ‘LION’ has gone away you can pass here in relative safety).
84. “W” —Cairn in forest. (A ‘RUSTY KEY’ is here).
85. “GET RUSTY KEY” —O.K. (How about dusting it).
86. “DUST RUSTY KEY” —The rust has been dusted off your key.
87. “E” —Lion pits.
88. “S” —Cleared area of forest.
89. “W” —Large tree in forest.
90. “S” —Thin forestry.
91. “S” — Clearing. (Your ‘KEY’ is now able to fit the lock of the ‘DOOR’ and ‘OPEN’ it without much trouble).
92. “OPEN DOOR” — Mouth of a cave. You have passed through the door.
93. “E” —Dimly lit part of cave. (An ‘AIR CYLINDER’ is here).
95. “GET AIR CYLINDER” —O.K. (I suggest wearing it).
96. “WEAR AIR CYLINDER” —O.K.
97. “W” —Bear’s cave.
98. “N” —Mouth of cave.
99. “N” —Clearing.
100. “DROP KEY” — O.K. You no longer need it and you have to ‘DROP’ it to be able to pick up the ‘CORD’).
101. “GET CORD” —O.K.
102. “N” —Thin forestry.
103. “N” —Large tree in forest.
104. “E” —Cleared area of forest. Your presence has been detected.
105. “S” —Dense part of forest.
106. “S” —Swamp in forest.
107. “E” —Bank of river.
108. “N” —Glade.
109. “N” —Path beside a river.
110. “N” —Woodcutter’s hut.

111. “*E*” —Part of a path.
112. “*S*” — Entrance to outhouse. (The wall must be climbed one way or other but first it is necessary for some object which cannot be used further to be dropped).
113. “*DROP LIT TAPER*” —O.K.
114. “*DROP BOOTS*” —O.K.
115. “*THROW CORD*” —It has caught onto something.
116. “*CLIMB CORD*” — Rubbish dump. (The “*CORD*” still remains in your possession since it may be used again later).
117. “*GET DIAMOND*” —O.K. Try dusting it.
118. “*DUST DIAMOND*” —Your diamond is now a lethal weapon.
119. “*DROP DUSTER*” —O.K. (This is of no further use).
120. “*E*” —Thick bushes in maze.
121. “*EXAMINE BUSHES*” —See what has been revealed.
122. “*GET GARLIC*” —O.K.
123. “*E*” —Burnt area. (A “*SPEAR*” is here).
124. “*GET SPEAR*” —O.K.
125. “*S*” —Exit from maze.
126. “*E*” — Steep sides of a mountain. (Two of your objects are to be dropped here — they would clutter up your “*Inventory*” and are not required till later).
127. “*DROP SPEAR*” —O.K.
128. “*DROP CORD*” —O.K.
129. “*GET SNOW*” —O.K.
130. “*W*” —Exit from maze.
131. “*GET SHOTGUN*” —O.K. (A useful weapon).
132. “*W*” — Area guarded by a cobra. The cobra keeps away from your garlic.
133. “*W*” —Thick undergrowth. (Some “*UNDERGROWTH*” is here).
134. “*EXAMINE UNDERGROWTH*” — See what has been revealed. (A “*GRAVE*” is here — your curiosity will make you want to know what is inside it).
135. “*DIG GRAVE*” — Dracula is here in an advanced state of reincarnation. See what has been revealed. (Dracula never seems to give up,

but on the other hand you should not as well).

136. “*DROP SPADE*” — O.K. (You need to clear room under ‘Inventory’ and you do not need this any more).

137. “*S*”—Centre of maze.

138. “*EXAMINE FOUNTAIN*” — See what has been revealed. (The ‘BULLET’ goes with the ‘SHOTGUN’).

139. “*GET BULLET*”—O.K. You have been noticed.

140. “*N*” — Thick undergrowth. (You now have the equipment with which you can once and for all kill ‘DRACULA’).

141. “*SHOOT DRACULA*”—Dracula is really dead now.

142. “*GET CROSS*” — O.K. (You would have been killed if you had tried to pick this up while ‘DRACULA’ was still there).

143. “*S*”—Centre of maze. (You are looking for another weapon).

144. “*S*” — Minotaur’s section of maze. (Your special ‘DIAMOND’ will allow you to ‘KILL’ the ‘MINOTAUR’).

145. “*KILL MINOTAUR*” — Part of a maze with a lever. The force resulting from the Minotaur’s destruction has project you into the next room.

146. “*N*” — Entrance to a maze. (A ‘CRUCIBLE’ is here — things can be melted in this).

147. “*MELT SNOW*”—The crucible is now clean.

148. “*MELT CROSS*” — A flick knife has been formed. (You have now that other weapon you were needing).

149. “*S*”—Part of a maze with a lever.

150. “*E*”—Minotaur’s section of maze.

151. “*N*”—Centre of maze.

152. “*N*”—Thick undergrowth.

153. “*E*” — Area guarded by a cobra. The cobra keeps away from your garlic.

154. “*E*”—Exit from maze.

155. “*E*” — Steep sides of a mountain. (The two objects that you dropped are here).

156. “*DROP DIAMOND*” — O.K. (You no longer need this object and the ‘GARLIC’).

157. “*DROP GARLIC*”—O.K.

158. “*GET CORD*” —O.K.

159. “*GET SPEAR*” —O.K.

160. “*S*” — Pass in a mountainous region. (You are lucky that you are wearing the “*AIR CYLINDER*”).

161. “*W*” — Large Valley. (You have to “*THROW*” the “*AIR CYLINDER*” away immediately since there has been a sharp increase in external air pressure on the descent causing it to become unsafe).

162. “*THROW AIR CYLINDER*” —It has exploded in mid-air.

163. “*S*” —Sides of a lake.

164. “*E*” —Rocky part of a route. (A “*WEREWOLF*” is here).

165. “*SPEAR WEREWOLF*” —It has run away wounded.

166. “*E*” — Old windswept road. (The “*WEREWOLF*” is in this location now).

167 “*STAB WEREWOLF*” — It has shrivelled up disappearing into the ground.

168. “*N*” — Outside of city gates. (To finish the adventure you need to “*CLIMB*” over them with the help of your “*CORD*”).

169. “*THROW CORD*” —It has caught onto something.

170. “*CLIMB CORD*” —Home city.

Score = 75 Best Score = 75

Press space to start again.

From the above possible solutions you may also learn something about the actual playing of an adventure. Since there is a limit to the number of objects that you can carry at any one time you will need to choose which you need and which you do not. It will therefore be necessary to “*DROP*” objects which you will need, in places where you will be able to retrieve them later. Some objects like the “*CORD*” in “*Journey*” can be used more than once, but you will find out things like this in an adventure after playing it several times. A rule about playing adventures is that if one thing does not work then try another.