

### ZX81 Adventure - Create Your Own

Suitable for : 16K RAM

This program allows you to create your own adventure-type games on the ZX81! A complete game is included for you to see how it is done, and also to give you a bit of fun into the bargain.

If you are already acquainted with the original Adventure game, then you may prefer to skip the next few paragraphs.

Adventure was originally written to run on computers somewhat larger than a ZX81 (having roughly 256K bytes of memory available - think about that!), and running faster as well. The game does not vary from one run to another, as the layout of the game remains fixed. Although this may seem a bit limited, the original version is so large that you become bored with playing before you find out all there is to know.



So how is it played? The game takes place in a network of caves, and the object of the game is to gather all the treasure that is dotted around in these caves. Some of the treasure cannot be taken straight away - for example, there is a pearl stuck inside a giant clam, and part of the fun of the game is to discover a way of getting the clam open.

At each turn, you are given a description of your current location. Then the game waits for you to type in up to two keywords. These keywords may ask for you to be moved in a certain direction (for example: GO SOUTH or WALK EAST or CLIMB UP) or to manipulate an object which is in sight - TAKE GOLD or TAKE JEWELS etc.

The objects are not necessarily treasures; they may seem useless at times, but nearly always have some use throughout the game to enable you to get past an obstacle.

In order to give the game some "spice", dwarves pop up at random and throw knives at you (if one hits you, you are killed), although you can retaliate with an axe or some other weapon.

Since the original game is too vast to run on a ZX81, I have borrowed several ideas from different sources to bring a version which can be tailored to run many different games - you will be quite capable of producing your own and swapping them with friends. The game comes in three portions:-

Adventure Master program

which controls the flow of the game and checks that all commands are suitably obeyed

Adventure Loader

which allows you to set up games of your own

Adventure Information

this makes each game unique - it contains the room descriptions, keyword actions, objects etc.

### Steps in creating your Adventure

If this is your first encounter with Adventure-type games, I would suggest that you start by entering the games in this section, playing them, then returning to this text when you have a better idea of the capabilities of the game.

The steps involved in creating a game are as follows:-

1. Enter the Master program and the Loader routine together. Save this combination on tape, as it becomes the basis for all your own games.
2. Enter the text messages and room descriptions (use either "The City of Alzan" or the mini-test game found later on).  
Type RUN 9000 to start the Loader program. Enter all the arrays and
3. variables as given in the appropriate game. The Loader program will automatically save the program for you.
4. Start the game by GOTO 10 (not normally required, since the Loader program ensures that the game starts once loaded from tape).

### How the Master program works

The following discussion may seem involved, but if you follow the Test Adventure carefully (I have annotated it fully), you should quickly grasp the way that the Master program works.

Referring to the three portions of the game mentioned above, the Information section requires several items in order to make a complete game. You may like to look at the Test Adventure while you see how the Information section is built up. The items required are:-

#### 1. Room descriptions

Each room has a unique number associated with it in the range 1 to (max. no. of rooms), and the Master program causes a subroutine call to line  $8000 + (\text{room} \times 10)$  so that the room description can be printed. This means that the description for room number 1 should start at line 8010, and a RETURN statement should be included after the description is printed, Room 2 description will be at line 8020, room 3 at 8030, and so on.

#### 2. Text messages

These are messages which are printed by the Master program on request (see later). The reason for printing these messages can be extremely varied - for example, if the keywords "LIGHT LAMP" are entered, and the lamp is already lit, then you may want to print a message saying so. As for rooms, each message has a unique number starting from 1, and message number 1 should be placed at line number 7010 (followed by a RETURN statement). Message 2 should be placed at line 7020, and so on.

#### 3. Objects

Objects can be used by the player throughout the game either carried or just manipulated in some way. Each object needs an entry in two arrays -  $O()$  which tells the Master program in which room the object is (initially) located, and  $O\$( )$  which contains the text describing the object. This description is 16 characters in length (although you could alter this if required for a particular game). The simple variable  $O$  contains the total number of objects. An object which does not initially exist must be given a room number of zero.

#### 4. Vocabulary

This table gives a list of the keywords that your game will recognise as words to be acted upon (in some way). With each word is a two-digit number that this keyword is translated into, so that two different keywords can be translated into the same action (e.g. LEAVE and EXIT would both be given the same two-digit code as they both have the same interpretation). The first twelve keywords are reserved for direction commands (NORTH, SOUTH, UP, DOWN etc), but this is not totally inflexible. These keywords are held in array V\$, and each entry is 6 characters long. The first four are the keyword (only the first four characters of the keyword are matched) and the last two are the two-digit "translated code" number. Note that the code number must have a leading zero for codes less than 10. Variable V should contain the total number of keywords in the array V\$().

#### 5. Room connection table

Array M\$() contains a table of the "tunnels" connecting each room. For every room, there is an entry showing the number (and direction) of tunnels leading away from the room. Each tunnel requires four characters - two to give the keyword code corresponding to the direction of this tunnel, and two representing the room number that the tunnel leads to (with leading zeros if necessary). A keyword code of "00" indicates the end of the list for this room. As an example, room 1 would be held in M\$(1) and might look like:-

```
M$(1) "0127061300"
```

This indicates that keyword code 01 will cause the Master program to continue at room 27, while keyword code 06 will cause the Master to continue at room 13. The final 00 signifies the end of the list.

Variable R should contain the total number of rooms. Array M\$ allows for 32 characters per room maximum - this will be adequate for almost any set-up.

#### 6. Action table

The action table gives a list of actions to be performed when certain keywords (or combinations of keywords) are entered. Note that direction movements are catered for by the Room Connection table above. Each entry in the Action table gives:-

```
Keyword 1 code (or 00 if any word is sufficient)
Keyword 2 code (or 00 if any word is sufficient)
Further conditions
```

Actions to be performed if all conditions are met. A small example at this stage might be :- "If the keywords "LIGHT LAMP" are entered and the lamp is already alight, then display message 5"

A full list of available conditions and actions can be found below, but the overall format of each entry is:

```
AABBC01C02.A01A02A03.
```

...where AA and BB represent the keyword codes for two keywords, C01 C02 represent additional conditions (you may have more than two) followed by a full-stop which indicates the end of conditions. Next follows the appropriate action codes (A01 A02 and A03 - again, you may have more if you wish) also terminated by a full-stop. A complete example is given after the tables below.

Array A\$() contains the action table, and variable A must contain the total number of items in the array.

The limit of each entry in array A\$() is 31 characters. If this should be insufficient, you can get round the problem by including one action that sets a temporary marker. Further entries can be added into the array C\$() which test for this marker and continue the required effects, then unset the temporary marker. This idea is used in "The City of Alzan"

## 7. Conditional table.

This table is almost identical to the Action table, but does not have any associated keywords. This table is scanned before each command is entered, and allows you to cater for some special circumstances, such as having a dwarf pop up in front of the player, or displaying different messages dependant on certain conditions. Array C\$() contains the conditions and variable C contains the total number of items in the array. The format of each entry is:-

C01C02.A01A02.

...where C01 C02 A01 etc are defined as for the Action table. Notice the full-stops that terminate both the conditions and actions.

The various conditions you may use consist of three-character codes. The first is a letter which signifies the desired condition, the last two are a two-digit parameter associated with the condition (shown as nn below):-

<u>Condition code</u>	<u>Test made</u>
A	nn is the current room number
B	Object nn is here (or being carried)
C	Object nn is not here (or being carried)
D	Object nn is being carried
E	Marker nn is set
F	Marker nn is not set
G	Countdown nn has reached value 1
H	Random number from 1-99 is less than nn

<u>Action code</u>	<u>Action performed</u>
A	Print list of objects carried
B	Carry object nn
C	Put down object nn
D	Display text message nn - causes GOSUB to line 7000+(nn*10)
E	Set marker nn
F	Unset marker nn
G	Set countdown nn to the value mm
H	Swap objects nn and nn+1 in the object table
I	Set object nn into current room number
J	Set object nn room number to 00
K	Set current room number to nn (i.e. forced move to room nn)
L	Print "OKAY" and await a new command
M	Await a new command
N	Await a new command, but the Conditional table is not scanned first
O	Describe current room then await new command

U	Describe current room then await new command
P	Abandon the game (player is asked "ARE YOU SURE?" and if the answer is Y then the game is abandoned)
Q	Stop the game

Here is an example of these in use:-

The keyword "TAKE" has (say) the keycode 15, and "GOLD" has the keycode 22. The object "gold" is number 3.

Suitable entries into the Action table might be:-

1522B03.B03L.

This means:-

Keywords 15 and 22 must be typed (TAKE GOLD), and the condition B03 must also be true (i.e. object 03 must be here or being carried). If this is true, then actions B03 and L are obeyed - object 03 is now carried, and then the message "OKAY" is printed and a new command is input.

There are 10 "markers" that can be set/unset and tested (see the above tables). All markers are initially unset. Markers 1 to 3 have a special significance to the Master program, but all others can be used as required. The special ones are:-

- 1 Indicates the total number of objects being carried
- 2 Tells the Master program whether the room is a "dark" room or "light" room - i.e. whether a lamp is required in order to see.
- 3 Tells the Master program whether a lamp is off or on.

If the room is a "dark" room, and the lamp is off, the Master program prints a message "IT IS DARK. BETTER GET SOME LIGHT OR YOU MAY BE IN TROUBLE".

Marker 2 should be unset when the player is above ground, and set once he goes underground.

There are also 5 countdown markers that are for general use. They can be set to any desired two-digit value by using Action code G. This code requires two parameters - the countdown number and the value it is to be given. E.g. G0105 would set countdown 1 to the value 5. Countdowns 1 to 4 are automatically reduced under certain conditions by the Master program:-

- 1 Reduced each time a command is entered.
- 2 Reduced each turn when marker 2 is set (i.e. when the player is in "dark" rooms).  
Reduced each turn when marker 2 is set but marker 3 is not - i.e. when
- 3 the player has not got a lamp on but it is dark. This lets you drop the player in a pit after (say) three moves in total darkness.
- 4 Reduced each time a command is entered (as for countdown number 1).

Condition 7 lets you test if a countdown marker has reached the value 1, so that you can perform some actions once a limit has been reached.

You should note the following points:-

1. A maximum of 5 objects can be carried at any one time (line 4100 in the Master program). Further objects can only be picked up if another object is dropped first.  
The Master program checks that you are not already carrying an object when it obeys action B, and that you are carrying the object when it
- 2.

-- obeys action C. This saves quite a considerable number of items in the Action table.

At the end of the program listing, you will find a small "test" adventure of four rooms, which will allow you to see what is going on and also see if your new program works.

You may like to send a copy of your own Adventure games to us at the address given at the front of this book - if we receive enough good ones to publish, we will pay for all those included.

1	<u>REM</u> ZX81 ADVENTURE MASTER	Tape name: "ADVENT"
2	<u>REM</u> *****	
3	<u>PRINT</u> "DO NOT USE ""RUN"". "	(see pages 95-96
10	<u>DIM</u> S(10)	(switch array
20	<u>DIM</u> C(5)	(countdown array
30	<u>LET</u> ROOM=1	(initial room no.
40	<u>DIM</u> P\$(2,2)	(keywords 1 & 2
50	<u>DIM</u> O(0)	(objects array - type the
		(letter "O" and not "0".
60	<u>FOR</u> X=1 <u>TO</u> 0	(set up objects initially
70	<u>LET</u> O(X)=Q(X)	
80	<u>NEXT</u> X	
100	<u>IF</u> <b>NOT</b> S(2) <u>THEN</u> <u>GOTO</u> 200	(test if darkness
110	<u>IF</u> C(2) <u>THEN</u> <u>LET</u> C(2)=C(2)-1	(darkness countdown
120	<u>IF</u> S(3) <u>THEN</u> <u>GOTO</u> 200	(see if lamp on
130	<u>PRINT</u> "IT IS DARK - BETTER GET SOME", "LIGHT OR YOU MAY BE IN TROUBLE."	
140	<u>IF</u> C(3) <u>THEN</u> <u>LET</u> C(3)=C(3)-1	(no lamp countdown
150	<u>GOTO</u> 1000	(wait for a command
200	<u>REM</u> DESCRIBE ROOM	
210	<u>PRINT</u>	
220	<u>GOSUB</u> 8000+ROOM*10	(print room description
300	<u>LET</u> F=0	(reset flag
310	<u>FOR</u> X=1 <u>TO</u> 0	(print any objects here
320	<u>IF</u> O(X)<>ROOM <u>THEN</u> <u>GOTO</u> 500	
330	<u>IF</u> F <u>THEN</u> <u>GOTO</u> 400	
340	<u>PRINT</u> ,,"THERE IS ALSO:"	
350	<u>LET</u> F=1	
400	<u>PRINT</u> " ";O\$(X)	
500	<u>NEXT</u> X	
1000	<u>REM</u> ACCEPT COMMAND	(await a command
1010	<u>LET</u> T=1	(first check automatics
1020	<u>GOTO</u> 2000	
1100	<u>IF</u> C(1) <u>THEN</u> <u>LET</u> C(1)=C(1)-1	(countdown every command
1110	<u>IF</u> C(4) <u>THEN</u> <u>LET</u> C(4)=C(4)-1	(countdown every command
1120	<u>PRINT</u> ,,">"	(prompt - use inverse
1130	<u>INPUT</u> Y\$	(input command
1140	<u>CLS</u>	
1150	<u>LET</u> Y=0	(command scan
1160	<u>PRINT</u> ">";Y\$	(print command at top
1170	<u>LET</u> P\$(2)="00"	

1170 <u>END</u> IF (Z) = 0	
1200 <u>FOR</u> W=1 <u>TO</u> 2	(get (up to) two keywords
1210 <u>GOSUB</u> 6000	
1220 <u>IF</u> Y>= <b>LEN</b> Y\$ <u>THEN</u> <u>GOTO</u> 1300	(check if all scanned
1230 <u>IF</u> P\$(W)="00" <u>THEN</u> <u>GOTO</u> 1210	(was the keyword found?
1240 <u>NEXT</u> W	(next keyword
1300 <u>IF</u> P\$(1)<>"00" <u>THEN</u> <u>GOTO</u> 1600	(was at least one word?
1310 <u>PRINT</u> " PARDON?"	(printed if nothing found
1320 <u>GOTO</u> 100	(try again
1600 <u>REM</u> CHECK FOR MOVEMENT	
1610 <u>LET</u> Z=1	(now scan movement table
1620 <u>LET</u> T\$=M\$(ROOM) (Z <u>TO</u> Z+1)	(get matching keyword
1630 <u>IF</u> T\$="00" <u>THEN</u> <u>GOTO</u> 1900	(check if end of entry
1640 <u>IF</u> T\$<>P\$(1) <u>THEN</u> <u>GOTO</u> 1700	(see if it matches word 1
1650 <u>LET</u> ROOM= <b>VAL</b> (M\$(ROOM) (Z+2 <u>TO</u> Z+3))	
1660 <u>GOTO</u> 100	(continue in new room
1700 <u>LET</u> Z=Z+4	(try next match
1710 <u>GOTO</u> 1620	
1900 <u>LET</u> T=0	(set "Action table" flag
1910 <u>LET</u> MATCH=0	(no match found yet
2000 <u>REM</u> CHECK FOR CONDITIONALS	
2010 <u>LET</u> CP=0	(table subscript number
2100 <u>LET</u> CP=CP+1	
2110 <u>IF</u> <b>NOT</b> T <u>THEN</u> <u>GOTO</u> 2300	(see if scanning Action
2120 <u>LET</u> E\$=C\$(CP)	(get from Conditionals
2130 <u>GOTO</u> 2600	
2300 <u>IF</u> CP<=A <u>THEN</u> <u>GOTO</u> 2400	(have all been scanned?
2310 <u>IF</u> MATCH <u>THEN</u> <u>GOTO</u> 1000	(has a match been found?
2320 <u>PRINT</u> "YOU CANT";	(print message
2330 <u>IF</u> <b>VAL</b> (P\$(1))<13 <u>THEN</u> <u>PRINT</u> " GO THAT WAY";	
2340 <u>PRINT</u> "."	
2350 <u>GOTO</u> 100	(try again
2400 <u>IF</u> A\$(CP) (1 <u>TO</u> 2)<>P\$(1) <u>THEN</u> <u>GOTO</u> 2100	
	(check if matches key 1
2410 <u>LET</u> Y\$=A\$(CP) (3 <u>TO</u> 4)	(get keycode 2
2420 <u>IF</u> Y\$<>"00" <u>AND</u> Y\$<>P\$(2) <u>THEN</u> <u>GOTO</u> 2100	
2430 <u>LET</u> E\$=A\$(CP) (5 <u>TO</u> )	(get conditions/actions
2600 <u>REM</u> CONDITIONS	
2610 <u>LET</u> E=1	(now scan further conds.
2700 <u>IF</u> E\$(E)="." <u>THEN</u> <u>GOTO</u> 3000	(full-stop ends conds.
2710 <u>LET</u> TYPE= <b>CODE</b> (E\$(E))-38	(get condition code
2720 <u>LET</u> N= <b>VAL</b> (E\$(E+1 <u>TO</u> E+2))	(get parameter
2800 <u>GOSUB</u> 2900+TYPE*10	(evaluate if true/false
2810 <u>IF</u> <b>NOT</b> OK <u>THEN</u> <u>GOTO</u> 2100	
2820 <u>LET</u> E=E+3	(try next condition
2830 <u>GOTO</u> 2700	
2900 <u>LET</u> OK=(N=ROOM)	(condition A - see
2905 <u>RETURN</u>	( text
2910 <u>LET</u> OK=(O(N)=ROOM <u>OR</u> O(N)<0)	(condition B

2915	<u>RETURN</u>	
2920	<u>LET</u> OK= (O(N) <> ROOM <u>AND</u> O(N) >=0)	(condition C
2925	<u>RETURN</u>	
2930	<u>LET</u> OK= (O(N) <0)	(condition D
2935	<u>RETURN</u>	
2940	<u>LET</u> OK=S(N)	(condition E
2945	<u>RETURN</u>	
2950	<u>LET</u> OK= ( <b>NOT</b> S(N))	(condition F
2955	<u>RETURN</u>	
2960	<u>LET</u> OK= (C(N)=1)	(condition G
2965	<u>RETURN</u>	
2970	<u>LET</u> OK= (( <b>INT</b> (RND*100)+1) <=N)	(condition H
2975	<u>RETURN</u>	
3000	<u>REM</u> ACTIONS	
3010	<u>LET</u> MATCH=1	(now perform actions
3020	<u>LET</u> E=E+1	
3100	<u>IF</u> E\$(E)=". " <u>THEN</u> <u>GOTO</u> 2100	(all done?
3110	<u>LET</u> TYPE= <b>CODE</b> (E\$(E))-38	(get action code
3120	<u>IF</u> E\$(E+1) <> ". " <u>THEN</u> <u>LET</u> N= <b>VAL</b> (E\$(E+1 <u>TO</u> E+2))	(get any parameter
3200	<u>LET</u> BREAK=0	(return line number
3210	<u>GOSUB</u> 4000+TYPE*100	(perform action
3220	<u>IF</u> BREAK <u>THEN</u> <u>GOTO</u> BREAK	(goto relevant line
3230	<u>LET</u> E=E+3	(next action
3240	<u>GOTO</u> 3100	
4000	<u>PRINT</u>	(action A - see table
4010	<u>PRINT</u> "YOU ARE HOLDING:"	
4020	<u>LET</u> F=1	
4030	<u>FOR</u> X=1 <u>TO</u> 0	(the letter "O" not "0"
4040	<u>IF</u> O(X) >=3 <u>THEN</u> <u>GOTO</u> 4070	
4050	<u>PRINT</u> " ";O\$(X)	
4060	<u>LET</u> F=0	
4070	<u>NEXT</u> X	
4080	<u>IF</u> F <u>THEN</u> <u>PRINT</u> " NOTHING."	
4090	<u>LET</u> BREAK=100	
4095	<u>RETURN</u>	
4100	<u>IF</u> S(1) <5 <u>THEN</u> <u>GOTO</u> 4140	(action B
4110	<u>PRINT</u> "YOU CANNOT CARRY MORE."	
4120	<u>LET</u> BREAK=100	
4130	<u>RETURN</u>	
4140	<u>IF</u> O(N) ==-1 <u>THEN</u> <u>GOTO</u> 4180	
4150	<u>LET</u> O(N) ==-1	
4160	<u>LET</u> S(1)=S(1)+1	
4170	<u>RETURN</u>	
4180	<u>PRINT</u> "YOU ALREADY HAVE IT."	
4190	<u>GOTO</u> 4120	
4200	<u>IF</u> O(N) ==-1 <u>THEN</u> <u>GOTO</u> 4240	(action C
4210	<u>PRINT</u> "YOU DONT HAVE ";O\$(N)	

4220 <u>LET</u> BREAK=100	
4230 <u>RETURN</u>	
4240 <u>LET</u> O(N)=ROOM	
4250 <u>LET</u> S(1)=S(1)-1	
4260 <u>RETURN</u>	
4300 <u>PRINT</u>	(action D
4310 <u>GOSUB</u> 7000+N*10	
4320 <u>RETURN</u>	
4400 <u>LET</u> S(N)=1	(action E
4410 <u>RETURN</u>	
4500 <u>LET</u> S(N)=0	(action F
4510 <u>RETURN</u>	
4600 <u>LET</u> C(N)= <b>VAL</b> (E\$(E+3 <u>TO</u> E+4))	(action G
4610 <u>LET</u> E=E+2	
4620 <u>RETURN</u>	
4700 <u>LET</u> X=O(N)	(action H
4710 <u>LET</u> O(N)=O(N+1)	
4720 <u>LET</u> O(N+1)=X	
4730 <u>RETURN</u>	
4800 <u>LET</u> O(N)=ROOM	(action I
4810 <u>RETURN</u>	
4900 <u>IF</u> O(N)<0 <u>THEN</u> <u>LET</u> S(1)=S(1)-1	(action J
4910 <u>LET</u> O(N)=0	
4920 <u>RETURN</u>	
5000 <u>LET</u> ROOM=N	(action K
5010 <u>RETURN</u>	
5100 <u>PRINT</u> " OKAY."	(action L
5200 <u>LET</u> BREAK=1000	(action M
5210 <u>RETURN</u>	
5300 <u>LET</u> BREAK=1100	(action N
5310 <u>RETURN</u>	
5400 <u>LET</u> BREAK=100	(action O
5410 <u>RETURN</u>	
5500 <u>PRINT</u> " ARE YOU SURE? ";	(action P
5510 <u>INPUT</u> W\$	
5520 <u>PRINT</u> W\$	
5525 <u>LET</u> BREAK=1100	
5530 <u>IF</u> <b>CHR\$ CODE</b> W\$<>"Y" <u>THEN</u> <u>GOTO</u> 5400	
5600 <u>GOTO</u> 9999	(action Q
6000 <u>REM</u> REMOVE WORD	
6010 <u>DIM</u> W\$(4)	(first four letters
6015 <u>LET</u> P\$(W)="00"	(set "not found" reply
6020 <u>GOSUB</u> 6600	(find first character
6025 <u>IF</u> END <u>THEN</u> <u>RETURN</u>	(test if end of command
6030 <u>FOR</u> Q=1 <u>TO</u> 4	(get four letters
6040 <u>LET</u> W\$(Q)=Y\$(Y)	
6050 <u>GOSUB</u> 6500	(check if word end
6060 <u>IF</u> END <u>THEN</u> <u>GOTO</u> 6100	
6070 <u>NEXT</u> Q	

```

6070 NEXT Q
6080 GOSUB 6500 (look for end of word
6090 IF NOT END THEN COTO 6080
6100 IF W$=" " THEN RETURN (no word entered
6110 FOR Q=1 TO V (scan vocabulary table
6120 IF W$=V$(Q) (3 TO ) THEN GOTO 6200
6130 NEXT Q
6140 RETURN (not found in table
6200 LET P$(W)=V$(Q) ( TO 2) (get keyword code number
6210 RETURN
6500 LET Y=Y+1 (check for end of word
6510 LET END=(Y>LEN Y$)
6520 IF END THEN RETURN
6530 LET END=(Y$(Y)=" ") (don't forget the space!
6540 RETURN
6600 LET Y=Y+1 (look for end of word
6610 LET END=(Y>LEN Y$)
6620 IF END THEN RETURN
6630 IF Y$(Y)=" " THEN GOTO 6600 (don't forget the space!
6640 RETURN
7000 REM ACTION MESSAGES
7001 REM MESSAGE NO. 1 CAUSES
7002 REM COSUB TO LINE 7010
7999 RETURN
8000 REM ROOM DESCRIPTIONS
8001 REM ROOM 1 CAUSES A
8002 REM GOSUB TO LINE 8010
9999 STOP

```

Now that you have the "central manager" portion, you need two further items before you can start running a game. First is a "game loader" routine that initialises all the essential arrays (like the object table, interconnecting room table, vocabulary table, and the automatic and conditional event tables) plus a few other variables. The second item is the game itself - all you have here is something that allows you to quickly create your own adventures.

Two complete mini-adventures are included here for you to load and run yourself, so that you can see how it all fits together (and have a laugh, I hope!), but first, here's the loader routine.

### Adventure Loader

This routine should be included with the manager program above, but it can be removed once the various arrays have been properly installed. I would advise you to keep the routine in your game until you have tested it properly - if you miss a few words out of the vocabulary table (or any other table) and you want to re-enter it, then you'll feel mad if you've just deleted the loader routine!

When you run it, it asks how many items are required in each array (like the vocabulary table), then dimensions the array, and inputs the elements one-by-one. It stops after each array has been created, thus giving you an opportunity to check what you've done. If you are re-entering an array that was incorrect, it also gives you a chance to re-input only one array and not all of them.

```

9000 REM GAME ARRAY LOADER

```

```

9010 CLS
9020 PRINT "NO. OF OBJECTS?"
9030 INPUT O
9040 DIM Q(O)
9050 DIM O$ (O,16)
9080 FOR X=1 TO O (type the letter "O" not "0".
9090 SCROLL
9100 PRINT "NO. ";X;" ROOM?",
9110 INPUT Q(X)
9120 PRINT Q(X)
9130 SCROLL
9140 PRINT "DESCRIPTION?",
9150 INPUT O$ (X)
9160 PRINT O$(X)
9170 NEXT X
9199 STOP (use CONT to continue
9200 CLS
9210 PRINT "NO. OF WORDS?"
9220 INPUT V
9230 DIM V$(V,6)
9240 FOR X=1 TO V
9250 SCROLL
9260 INPUT V$(X)
9270 PRINT V$(X)
9280 NEXT X
9299 STOP (use CONT to continue
9300 CLS
9310 PRINT "NO. OF ROOMS?"
9320 INPUT R
9330 DIM M$(R,32)
9340 FOR X=1 TO R
9350 SCROLL
9360 INPUT M$(X)
9370 PRINT M$(X)
9380 NEXT X
9399 STOP (use CONT to continue
9400 CLS
9410 PRINT "NO. OF CONDITIONALS?"
9420 INPUT C
9425 LET C=C+1
9430 DIM C$(C,21)
9440 FOR X=1 TO C-1
9450 SCROLL
9460 INPUT C$(X)
9470 PRINT C$(X)
9480 NEXT X
9490 LET C$ (C)=".N."
9499 STOP (use CONT to continue

```

```

9500 CLS
9510 PRINT "NO. OF ACTIONS?"
9520 INPUT A
9530 DIM A$ (A,31)
9540 FOR X=1 TO A
9550 SCROLL
9560 INPUT A$ (X)
9570 PRINT A$ (X)
9580 NEXT X
9599 STOP                                (use CONT to continue
9600 CLS
9610 PRINT "ENTER THE ADVENTURE NAME"
9620 INPUT N$
9630 PRINT ,, "START THE TAPE..."
9640 PAUSE 150
9645 POKE 16437,255
9650 CLS
9660 SAVE N$
9670 GOTO 10
9999 STOP

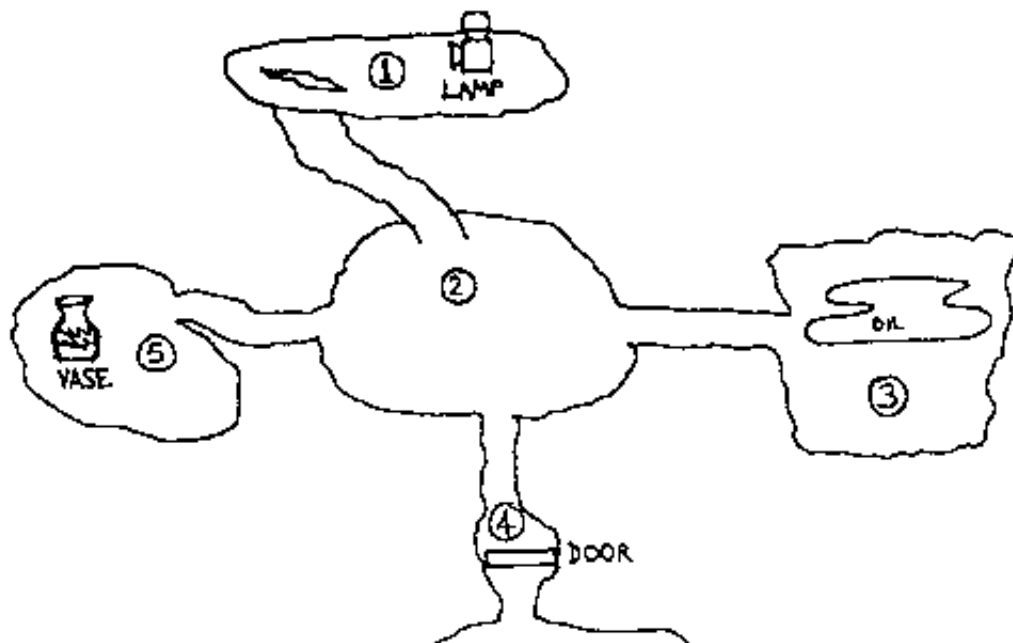
```

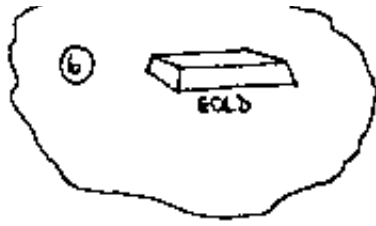
Before you get a real-live game, here follows a "test" Adventure for you to enter. It should give you a good idea as to how the program works, how you can create your own Adventures, and whether all your typing has been accurate.

### **Test Adventure**

This mini-test adventure uses 6 rooms. Room 1 is above ground, and a lamp can be found there. The objective is to get the bar of gold out of the caves back above ground. The gold is hidden in cave 6 behind a rusty door (cave 4), which will not open. Cave 5 contains a vase and cave 6 contains a pool of oil. Obviously, you must fill the vase with oil and then oil the door! Once this has been done, you can open the door and reach the gold.

A map of the cave looks like this:-





Markers 2 and 3 are used, as usual, to represent "dark" rooms and lamp off/on respectively. Notice that when the "lit" lamp (object number 2) is dropped, the lamp is marked as off, which prevents you from lighting the lamp then leaving it somewhere while you wander off.

Marker S is used to indicate when the door has been oiled, and marker 6 is set when the door is open.

Enter the text messages and room descriptions, then RUN 9000 to start the Loader routine. The objects, vocabulary, room connections, conditionals and keyword actions are all entered at this stage.

Once you have completed this, SAVE THE PROGRAM!!!

Start the program by GOTO 10 (otherwise you'll destroy the variables). Check that it works according to the rules above and you can be fairly sure that you have entered your Master program without any serious defects.

You should notice the way this is created in order to assist you with producing your own games.

One item of importance is shown between rooms 1 & 2 and also 4 & 6. In the first case, there is no tunnel indicated in the room connection table between rooms 1 and 2. Instead, an entry has been included in the Action table under the appropriate keyword (06). This is because I want to make sure that the "darkness" marker is set on whenever the keyword "DOWN" is given from room number 1.

Similarly, there would be no point in entering a connection between rooms 4 and 6, since the door is supposed to block the path. Consequently, an entry is found in the Action table (03 00 A04 F06....) which checks marker 6 whenever "SOUTH" is entered at room 4.

The rule is:- If you want to place some conditions on the player when he travels from one particular room to another, don't put an entry in the room connection table - use the Action table instead.

Text Messages:-

```
7010 PRINT "THE DOOR IS SHUT FAST"
7015 RETURN
7020 PRINT "THE DOOR IS OPEN"
7025 RETURN
7030 PRINT "IT IS ALREADY ALIGHT"
7035 RETURN
7040 PRINT "WITH A GRUNT YOU MANAGE TO",
      "OPEN THE DOOR."
7045 RETURN
7050 PRINT "IT IS TOO STIFF FOR YOU",
      "TO OPEN."
7055 RETURN
7060 PRINT "YOU DID IT. WELL DONE."
7065 RETURN
```

7070 PRINT "YOU CANNOT GET PAST THE DOOR."

7075 RETURN

Room Descriptions:-

8010 PRINT "YOU ARE STANDING BY A POTHOLE."

8015 RETURN

8020 PRINT "THIS IS A VAST CAVERN WITH",

"PASSAGES LEADING EAST,SOUTH,"

"AND WEST. A DIM PASSAGE SLOPES"

"UPWARDS BEHIND YOU."

8025 RETURN

8030 PRINT "THIS CAVE CONTAINS ONLY A POOL",

"OF OIL."

8035 RETURN

8040 PRINT "HERE IS A GIANT RUSTY DOOR."

8045 RETURN

8050 PRINT "YOU ARE IN THE WESTERN ALCOVE."

8055 RETURN

8060 PRINT "YOU ARE IN THE TREASURE CAVE."

8065 RETURN

Objects:-

Number of objects:- 5

<u>No.</u>	<u>Room number</u>	<u>Description</u>
1	1	A LAMP
2	0	A LIGHTED LAMP
3	5	A MING VASE
4	0	A VASE OF OIL
5	6	A BAR OF GOLD

Vocabulary:-

Number of words:- 25

Each entry below requires a maximum of six characters, the first two being the word number.

01N 14DROP

01NORT 15VASE

02E 16GOLD

02EAST 17DOOR

03S 18OPEN

03SOUT 19LAMP

04W 20LIGH

04WEST 21FILL

05U 22OIL

05UP 23INVE

06D 24QUIT

06DOWN 25LOOK

13TAKE

Number of rooms:- 6

Room connection table (the numbers in brackets are for reference only - do not enter them) :-

- (1) 00
- (2) 02030304040500
- (3) 040200
- (4) 010200
- (5) 020200
- (6) 010400

Number of conditionals:- 3

Conditionals (do not enter the spaces!) :-

A04 E06. D02 N.	(room 4 and marker 6 is set - (i.e. the door is open.
A04 F06. D01 N.	(room 4 and M6 is not set ie (the door is shut
A01 D05. D06 Q.	(room 1 carrying object 5 (got out with the gold - win!

Number of keyword actions:- 21

Keyword actions:-

13 19 B01. B01 L.	(take lamp - object 01
14 19 B01. C01 L.	(drop lamp
13 19 B02. B02 E03 L.	(take (lit) lamp - object 02 ( - also sets lamp marker 3
14 19 B02. C02 F03 L.	(drop lit lamp - unsets lamp ( marker 3
20 00 D01. H01 E03 L.	(light lamp - swaps objects 1 (and 2, also sets lamp mark
20 00 B02. D03 M.	(light lamp and object 2 is ( already here - display 03
06 00 A01. E02 K02 O.	(DOWN when at room 1, so set ( "dark" marker 2, continue at ( room number 2
05 00 A02. F02 K01 O.	(UP when at room 2, so unset ( "dark" marker and continue ( at room number 1
13 15 B03. B03 L.	(take vase
14 15 B03. C03 L.	(drop vase
13 16 B05. B05 L.	(take gold
14 16 B05. C05 L.	(drop gold
21 00 B03 A03. H03 L.	(fill vase - must have ( object 3 and be in room 3 ( swaps objects 3 & 4
22 00 A04 B04. H03 E05 L.	(oil door - must have object 4 ( a full vase and be at room 4 ( "empties" bottle 6, set mark5 (open door - must be oiled i o

18 00 A04 E05. D04 E06 M.	(open door - must be oiled i.e. ( marker 5 must be set, and ( must be at room 4. Sets M6.
18 00 A04 F05. D05 M.	(open door when not oiled. ( - displays message 5.
03 00 A04 F06. D07 M.	(SOUTH when marker 6 not set - ( i.e. door not open.
03 00 A04 E06. K06 O.	(SOUTH at door when open [mark ( 6 is set]. Continues at room ( number 6.
23 00 .A.	(give inventory
24 00 .P.	(quit
25 00 .O.	(look to see where we are

### **Testing Your Adventures**

What do you do if your new Adventure does not work? Here are a few guidelines to help you track down any errors.

From my own experience, the most common problem occurs in the Conditional and Action tables - either specifying incorrect actions, or not entering appropriate items.

You can run the program in either slow or fast mode, but I must point out that it can take quite a few seconds to scan the Action table to match your keywords and so I would recommend fast mode.

If nothing happens for one or two minutes, then suspect a fault in the Conditional table. Press the BREAK key, and inspect any of the following variables by using direct PRINT commands:-

CP	Contains the current Conditional/Action table entry number being processed.
T	indicates whether the Conditional or Action table is being scanned. Zero means Action table, 1 means Conditional table.
E\$	contains a copy of the Conditional/Action table entry.
P\$(1)	contains the keyword number of the first keyword found in any input command.
P\$(2)	contains the keyword number of the second keyword found, or "00" if no second keyword was entered.
ROOM	the current room number.
S(n)	switch n - 0=off, 1=on.
C(n)	countdown n - 0=countdown not in use.
O(n)	room number containing object n. If the object is being carried, this will have the value -1. If the object does not exist, the value will be zero.

The following arrays/variables are set up by the loader program:-

M\$()	room connection table
R	number of rooms
O	number of objects
Q()	object location table, copied into O()
O\$()	object descriptions
V	number of words
V\$()	vocabulary table
C	number of conditionals

C\$( )	conditional table
A	number of actions
A\$( )	action table

A common mistake is to terminate a Conditional table entry with action code M instead of N. Action code M causes the conditional table to be scanned again, and since the same conditionals (probably) still exist, the same program will simply loop indefinitely. Check that all conditionals finish with action code N unless you have good reason to use another (look at the tables in the "Test" Adventure).

For further information on Adventure, read the article by Ken Reed, Practical Computing Vol. 3, Issue 8 (August 1980).

### City Of Alzan

Suitable for : 16K RAM

Now for a complete Adventure, based on the "Do-it-yourself" Adventure Master program.

This takes place in a fictitious city named Alzan, which is built on top of the sea cliffs and is inhabited by thieves and cut-throats. Your quest is to find a way out of the city before they grab you, or before the plague takes hold of you. Unfortunately, the city is surrounded by extremely high walls and so you must find a way to scale them.

When you enter the game, it is possible for you to work out how the game evolves and how to win, but this would defeat the pleasure of playing, so try to "switch off" while you are typing.

When this program is fully running, you will have roughly 4150 bytes of memory free. This should give you a guide to the size of Adventures you will be able to write. I would advise the use of the "Memory Left" routine (see "Using Machine Code") while developing your own games.

Tape name : "ALZAN"

Enter the text messages:-

```

7010 PRINT "OH DEAR. YOU MUST HAVE CAUGHT",
      "THE PLAGUE IN THE TOMB. IT",
      "SEEMS THAT YOU HAVE DIED."
7015 RETURN
7020 PRINT TAB 12;"---WHOOSH---"
7022 PRINT "EL GRABBO, THE LOCAL THIEF,",
      "SNATCHES YOUR MONEY AND DIS-",
      "APPEARS INTO THE SEA MIST."
7025 RETURN
7030 PRINT "" "STOP THIEF" "" SHOUTS THE USHER,",
      "BUT YOU MANAGE TO ESCAPE."
7035 RETURN
7040 PRINT "THE COVER IS ALREADY OPEN."
7045 RETURN
7050 PRINT "IT COSTS MORE THAN YOU CAN AFFORD."

```



7055 RETURN  
7060 PRINT "THATLL DO NICELY, SIR"  
7065 RETURN  
7070 PRINT "THE MANHOLE COVER IS OPEN."  
7075 RETURN  
7080 PRINT "THE MANHOLE COVER IS SHUT."  
7085 RETURN  
7090 PRINT "THE SHOPKEEPER IS BIGGER THAN",  
      "YOU..."  
7095 RETURN  
7100 PRINT "YOU WILL NEED A LADDER TO GET",  
      "OVER THESE WALLS."  
7105 RETURN  
7110 PRINT "IT IS ALREADY ON."  
7115 RETURN  
7120 PRINT "WHAT A STROKE OF GENIUS"  
7125 RETURN  
7130 PRINT "YOU CATCH THE GUARDS UNAWARE AND";  
      "MANAGE TO SNATCH A WAD OF NOTES.";  
      "NO-ONE HAS NOTICED (FUNNY LOT, ",  
      "THESE ALZANS) "  
7135 RETURN  
7140 PRINT "YOU HAVE TAKEN ALL THERE IS."  
7145 RETURN  
7150 PRINT "I DONT SEE A TORCH?"  
7155 RETURN  
7160 PRINT "THE CINEMA IS BOOKED FOR A",  
      "PRIVATE FUNCTION."  
7165 RETURN  
8010 PRINT TAB 8;"WELCOME TO ALZAN",,,  
      "YOU MUST SCALE THE WALLS IF",  
      "YOU WISH TO ESCAPE FROM THIS",  
      "CITY OF THIEVES AND CUT-THROATS."  
8015 RETURN  
8020 PRINT "YOU ARE IN THE MAIN STREET OUT-"  
      "SIDE A HARDWARE SHOP. THE STREET";  
      "STRETCHES EAST/WEST AND A SMALL",  
      "ALLEY LEADS NORTH BESIDE THE",  
      "SHOP."  
8025 RETURN  
8030 PRINT "YOU ARE INSIDE THE SHOP. THE",  
      "SHOPKEEPER LOOKS SHIFTY, BUT HE",  
      "HAS MANY FINE GOODS ON DISPLAY."  
8035 RETURN  
8040 PRINT "YOU ARE IN AN ALLEY BEHIND THE",  
      "TALL BUILDINGS. THERE ARE MANY",  
      "FULL DUSTBINS UNDER THE FIRE",  
      "ESCAPE."

8045 RETURN

8050 PRINT "YOU ARE ON THE FIRE ESCAPE,"  
"WHICH LEADS PAST A DOOR IN THE",  
"BUILDINGS."

8055 RETURN

8060 PRINT "YOU HAVE COME DOWN A SECRET",  
"STAIRCASE INTO THE SHOP."

8065 RETURN

8070 PRINT "YOU ARE ON SOME CATWALKS BETWEEN";  
"THE BUILDINGS."

8075 RETURN

8080 PRINT "THIS IS PART OF THE CITY WALLS.",  
"THERE IS AN UNUSED DOOR IN THE",  
"WALL HERE."

8085 RETURN

8090 PRINT "YOU ARE AT A CROSSROADS."

8095 RETURN

8100 PRINT "HERE IS PART OF THE CITY WALLS.",  
"THE SEA MIST IS OUIE THICK,"  
"MAKING IT HARD TO SEE FAR."

8105 RETURN

8110 PRINT "YOO PLUNGE FROM THE WALL - RIGHT";  
"DOWN ONTO THE ROCKS BY THE SEA",  
"500FT BELOW. WELL, NEVER MIND,"  
"BETTER LUCK NEXT TIME."

8115 RETURN

8120 PRINT "YOU ARE OUTSIDE THE TOWN BANK."

8125 RETURN

8130 PRINT "INSIDE THE BANK THERE ARE MANY",  
"GUARDS WHO SEEM RATHER BORED."

8135 RETURN

8140 PRINT "YOU HAVE ARRIVED AT A DEAD END,"  
"BUT THERE IS A MANHOLE IN THE",  
"ROAD..."

8145 RETURN

8150 PRINT "YOU ARE IN A SMALL ALCOVE UNDER-";  
"NEATH THE MANHOLE, A PASSAGE",  
"LEADS SOUTH."

8155 RETURN

8160 PRINT "THE PASSAGE LEADS TO AN ANCIENT",  
"TOMB, WHERE MANY SARCOPHAGI LIE",  
"SCATTERED ABOUT. "

8165 RETURN

8170 PRINT "THE USHER WILL NOT LET YOU IN AS";  
"THE PROGRAMME HAS STARTED. HE",  
"BLOCKS YOUR PATH WITH HIS TORCH."

8175 RETURN

8180 PRINT "YOU ARE OUTSIDE THE CINEMA.",  
"SOUNDS OF CINEMA COME FROM"

"SOUNDS OF GUNFIRE COME FROM",  
"WITHIN."

8185 RETURN

8190 PRINT TAB 8; "\*\*\*CONGRATULATIONS\*\*\*" ,  
"YOU MADE IT OUTSIDE THE CITY",  
"WALLS. THIS IS INDEED A RARE",  
"OCCASION. WELL DONE."

8195 RETURN

Now type RUN 9000 to start the initialisation routine. The arrays should be set as follows:-

Number of objects : 11

Object room Description

0		A LIGHTED TORCH
0		A TORCH
3		A LADDER
3		A HAMMER
0		A HAMMER
0		A WAD OF NOTES
0		MANHOLE COVER
15		A BAG OF NAILS
16		A BARCLAYCARD
0		A ROUGH LADDER
4		SOME WOOD

Number of words : 43

01N	19HAMM
01NORT	20WAD
02E	20NOTE
02EAST	22BAG
03S	22NAIL
03SOUT	23BARC
04W	05SCAL
04WEST	05CLIM
05U	29OPEN
05UP	29LIFT
06D	30MAKE
06DOWN	30BUIL
13TAKE	31SWIT
14PUT	31LIGH
14DROP	32BUY
15ENTE	33WOOD
15IN	34ROB
16OUT	34STEA
16EXIT	35INVE
16LEAV	36QUIT
17TORC	37LOOK
18LADD	

Number of rooms : 19

<u>Room no</u>	<u>Connections</u>
1	00
2	01 04 02 09 04 18 00
3	00
4	02 02 05 05 00
5	06 04 04 07 00
6	00
7	01 08 03 05 00
8	03 07 00
9	01 12 02 10 03 14 04 02 00
10	04 09 00
11	00
12	02 09 04 18 00
13	00
14	01 09 00
15	03 16 00
16	01 15 00
17	00
18	01 12 02 02 00
19	00

Number of conditionals : 9

Conditional table (the spaces are only to make it easier to read - do not enter them) :-

A01. K02 O.  
A16 H30. G0121.  
G01. D01 Q.  
B06 H10. D02 J06.  
A14 E07. D07 N.  
A14 F07. D08 N.  
A11. Q.  
A19. Q.  
A06. K03 O.

Number of actions : 47

Action table (the spaces are to make it easier to read - do not enter them) :-

13 17 B01. B01 E03 L.  
13 17 A17 C01 C02. I02 B02 D03 K18 E10 O.  
32 18 B03. D05 N.  
13 19 B05. B05 L.  
13 20 B06. B06 L.  
29 00 A14 E07. D04 N.  
29 00 A14. E07 M.  
13 22 B08. B08 L.  
13 23 B09. B09 L.  
14 17 B01. C01 E03 L.

14 17 B01. C01 F03 L.  
14 17 B02. C02 L.  
14 19 B05. C05 L.  
14 20 B06. C06 L.  
14 22 B08. C08 L.  
14 23 B09. C09 L.  
05 00 A10 C10. D10 M.  
05 00 A08 C10. D10 M.  
05 00 A10. K11 O.  
05 00 A08. K19 O.  
05 00 A15. F02 K14 O.  
06 00 A14. E02 K15 O.  
31 00 D02. H01 E03 L.  
31 00 B01. D11 N.  
32 19 B04 B06. H04 J06 B05 L.  
32 19 B04 B09. H04 D06 B05 M.  
30 00 B05 B11 B08. D12 I10 J08 J11 M.  
13 33 B11. B11 L.  
14 33 B11. C11 L.  
15 00 A02. K03 O.  
15 00 A12. K13 O.  
15 00 A18 F10. K17 O.  
16 00 A03. K02 O.  
16 00 A13. K22 O.  
16 00 A17. K18 O.  
15 00 A05. K06 O.  
34 00 A03. D09 M.  
34 00 A13 E08. D14 M.  
34 00 A13. E08 D13 I06 B06 M.  
15 00 A18 E10. D16 M.  
13 18 B10. B10 L.  
14 18 B10. C10 L.  
13 18 B03. D09 M.  
13 17 B02. B02 L.  
35 00 .A.  
36 00 .P.  
37 00 .O.  
50 00 .N.

Now use the "save" routine giving the name ALZAN to this adventure. When you next load the program, it will automatically start running, but if you wish to begin again for any reason, use GOTO 1, as the RUN command will clear all variables thus destroying the game.

Have fun!