Initialisation

Although it would be regarded as 'proper' to declare all the variables used at the start we have only initialised those necessary to begin the program correctly. Arrays are all DIMensioned to their correct sizes at this point and the vital positioning strings of cursor movements are also created.

The dummy READ routine between 300 and 320 moves the current position in DATA over the first block which will be used later for building castle-type scenes.

Monster data is loaded into three arrays; the monster name, its initial strength and its initial magical power. These starting values are modified according to the 'floor level' of the scene on which it appears. The monster details are presented in Table 1. **HB**

```
 99  REM ** DEFINE MAJOR VARIABLES
100  DIM D(3),G(73),P(8),N(8),S(4),T(2)
110  DIM M$(18),MS(18),N1(18)
120  VG$="":GC$="":F$="":DL$=""
130  TS=0:TN=0:TM=3:CF=0
140  D$="[HOM][21 CD]"
150  D1$=LEFT$(D$,17)
160  SP$="[39 SPC]"
170  R$="[30 CR]"
180  R1$=LEFT$(R$,21)
299  REM ** SKIP SCENE DATA
300  FOR I=1 TO 32
310  READ C$
320  NEXT I
329  REM ** LOAD MONSTER DATA
330  FOR I=0 TO 18
340  READ M$(I):READ MS(I):READ N1(I)
350  NEXT I
```

Character Initialisation

This block of program allows the user to set up his character with a name and one of a number of options of character type; Wizard, Cleric, Barbarian, etc. Table 2 contains information on the various character types.

Alternatively, if the game has been played before, the user may have a character stored on tape so the option exists to load this instead of starting afresh. The selection is made in 1050 after the name has been entered. The maximum length of name is 16 characters (checked in line 1040) and, because the string has to be entered as an INPUT, a simple bomb-proof trap is inserted at line 1030. This works by forcing an asterisk to appear under the cursor so if you simply press RETURN this is entered rather than nothing. This trap should only be needed on the PET, other systems may allow more sophisticated trapping techniques.

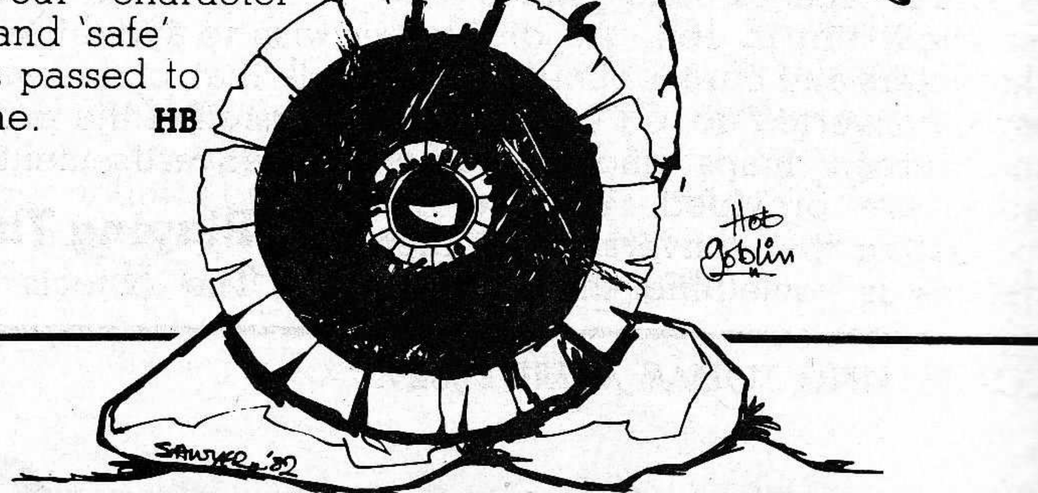
The tape input routine, from 1090 to 1210 is absolutely straightforward and can be changed to suit your system; TRS-80 would use INPUT# −1 followed by the list of variables and MZ80-K would require the file to be ROPENed first. Check your manual to find the appropriate method for your system.

If data entry is from tape, the stamina value is set to maximum and the program then jumps to 1400. If, however, this is the first time through the game, the tape loading section is skipped and the player offered the choice of five character types. According to your selection, the initial values of your character's physical and magical strengths are determined together with your two 'gain' factors. These determine how much you gain from finds and how well various sections of the combat work. The P1 factor also acts as a limit on your character's growth, see Table 2.

Once the character has been selected, the Valley is then created for the first time and the initial status information displayed. The game now starts with your character located in the left-hand 'safe' castle and control is passed to the Movement routine. **HB**

Monster	Physical	Magical	Code	Scenes
Thunder Lizard	50	0	V	V
Wolfen	9	0	A	V,W,S,C
Hob-Goblin	9	0	A	V,W,S,C
Orc	9	0	A	V,W,S,C
Ogre	23	0	A	V,W,S,C
Balrog	50	50	A	V,W,S,C
Fire Imp	7	3	E	V,W,C,L,T
Harpy	10	12	E	V,W,C,L,T
Fire Giant	26	20	E	V,W,C,L,T
Rock Troll	19	0	G	V,C
Centaur	18	14	H	V,W
Wyvern	36	12	F	W,S
Water Imp	15	15	L	W,S
Kraken	50	0	L	W,S
Minotaur	35	25	C	C,L,T
Wraith	0	30	C	C,L,T
Ring Wraith	0	45	C	C,L,T
Barrow Wight	0	25	B	L,T
Dragon	50	20	B	L,T

Table 1. The monsters, their strengths and their habitats. The code letter is extracted during the monster selection routine and matched against the current scene's 'allowable monsters' string. Note also that the two monsters coded with L can only be found in the lakes. The scenes are coded as; V-Valley, W-Woods, S-Swamps, C-Black Tower, L-Vounim's Lair and T-Temple of Y'Nagioth. If you wish to extend the range of monsters to the TSR Games series of 'Monsters Manuals' are valuable reference works.



```
 999  REM ** CHARACTER CHOICE AND LOAD
1000  PRINT "[CLS][CD]LOAD A CHARACTER FROM TAPE (Y/N) ?"
1010  VG$="YN":GOSUB 1500:REM ** UNIGET
1020  INPUT "[CD]CHARACTER'S NAME [2 CR]*[3 CL]";J$
1030  IF J$="*" THEN 1020
1040  IF LEN(J$)>16 THEN PRINT "[CD]TOO LONG":GOTO 1020
1050  IF GC$="N" THEN 1240
1060  PRINT "[CLS]PLACE DATA TAPE IN THE TAPE DECK"
1070  PRINT "[CD]IS IT REWOUND ?"
1080  GOSUB 1600:REM ** ANYKEY
1090  OPEN 1,1,0,J$
1100  INPUT#1,P$
1110  INPUT#1,TS
1120  INPUT#1,EX
1130  INPUT#1,TN
1140  INPUT#1,CS
1150  INPUT#1,PS
1160  INPUT#1,T(0)
1170  INPUT#1,T(1)
1180  INPUT#1,T(2)
1190  INPUT#1,C1
1200  INPUT#1,P1
1210  CLOSE 1
1220  C=150
1230  GOTO 1400
1240  PRINT "[CLS][2 CD]CHARACTER TYPES...CHOOSE
      CAREFULLY"
1250  PRINT
1260  PRINT "WIZARD     (1)"
1270  PRINT "THINKER    (2)"
1280  PRINT "BARBARIAN (3)","KEY 1-5"
1290  PRINT "WARRIOR    (4)"
1300  PRINT "CLERIC     (5)"
1310  GET GC$:IF GC$="" THEN 1310
1320  A=VAL(GC$)
1330  IF A=1 THEN P$="WIZARD":P1=2:C1=0.5:CS=22:PS=28
1340  IF A=2 THEN P$="THINKER":P1=1.5:C1=0.75:CS=24:PS=26
1350  IF A=3 THEN P$="BARBARIAN":P1=0.5:C1=2:CS=28:PS=22
1360  IF A=4 THEN P$="WARRIOR":P1=1:C1=1.25:CS=26:PS=24
1370  IF A=5 THEN P$="CLERIC":P1=1.25:C1=1:CS=25:PS=25
1380  IF A<1 OR A>5 THEN P$="DOLT":P1=1:C1=1:CS=20:PS=20
1390  EX=5:C=150
1400  PRINT "[2 CD]GOOD LUCK"
1410  PRINT "[CD]";J$;" THE ";P$
1420  DF=150:DL$="D":GOSUB 36000:REM ** DELAY
1430  GOSUB 10000:REM ** VALLEY DRAW
1440  DF=5:GOSUB 36000:REM ** DELAY + UPDATE
1450  GOTO 2000:REM ** MOVEMENT
```

Type	P1	C1	CS	PS	C	CS Max	PS Max
Wizard	2.00	0.50	22	28	100	66	777
Thinker	1.50	0.75	24	26	113	72	241
Barbarian	0.50	2.00	28	22	125	77	89
Warrior	1.00	1.25	26	24	113	75	117
Cleric	1.25	1.00	25	25	113	74	157
Dolt	1.00	1.00	20	20	113	75	117

Table 2. The six possible character types with their initial values and the maximums to which their physical and magical strengths can rise.

Fast Subroutines

UNIGET: This is a universal GET routine for the PET and is designed to operate in conjunction with the string VG$. It will only return to the main program if the character keyed is one of those in VG$. On other systems this may be replaced by the INKEY$ function.

ANYKEY: This routine is used in the tape save and load routines to allow the player to ensure the cassette is ready in the tape machine before proceeding, it can be removed or replaced as required.

COMBAT GET: A special timed GET routine for combat. It returns to the main program as soon as any key is pressed, assuming that this occurs within the time limit. The key pressed is held in GC$. If the time limit is exceeded the variable TV is set to 1. The routine also wipes away the text message "*** STRIKE QUICKLY ***" **HB**

```
1499  REM ** UNIGET ROUTINE
1500  GET GC$:IF GC$="" THEN 1500
1510  FOR I=1 TO LEN(VG$)
1520  IF MID$(VG$,I,1)=GC$ THEN RETURN
1530  NEXT I
1540  GOTO 1500
1599  REM ** ANYKEY ROUTINE
1600  PRINT "[CD]** PRESS ANY KEY TO CONTINUE **"
1610  GET GC$:IF GC$="" THEN 1610
1620  RETURN
1699  REM ** COMBAT GET ROUTINE
1700  FOR I=1 TO 10:GET GC$:NEXT I:REM ** EMPTIES BUFFER
1710  TV=0
1720  FOR I=1 TO 60
1730  GET GC$:IF GC$="" THEN 1750
1740  GOTO 1770
1750  NEXT I
1760  TV=1:REM ** NO KEY PRESSED
1770  PRINT D$;SP$:REM ** WIPE AWAY MESSAGE
1780  RETURN
```

Movement

In many ways this represents the core of the whole program; it is certainly the most executed loop and controls access to all other major routines.

For such an important routine it occupies surprisingly little space, lines 2000 to 2250 in fact. Line 2000 is only used as an initial starting point when you first enter the Valley, either at the start of the game or when you return to the Valley from a scenario; all other calls are made to 2010. The POKE code 81 is the symbol used to display your current position in the Valley, Table 3 gives alternatives for other systems.

The first operation is to give the character stamina a boost of 10, a dynamic refresh? Your current position is now examined to see if you are standing on a path or in the Valley and an appropriate message is printed requesting you to make a move. The player may move one square at a time in any direction, see Fig. 1. The choice of direction is made by keying one of the keys of the numeric keypad, the value of the key pressed then being inspected to find which direction it represents. In many programs of this type the checking is done by way of a look-up table which, although universal in operation and not restricted to numeric keypads is expensive in terms of memory.

Fortunately, one of the programming team had a mathematical background and produced the code between 2050 and 2090. Because each direction of movement corresponds directly to a screen displacement value, it must be possible to establish a simple mathematical relationship between the numeric key pressed and the direction in which you wish to move. In fact, the relationship is so simple no one appears to have thought of it before; another first for CT!

The routine starts at 2050 by clearing out the keyboard buffer, an essential operation to prevent old keystrokes causing problems. A character is now read in by the GET command in 2060 and checked to see if it is an 'E'. If it is an 'E', control is passed to the rating routine which gives your current 'Ego'. As we are only looking for numeric inputs in this routine we can test for validity by finding the VAL of the character; if this is 0 it must have been non-numeric so the program loops back for another character.

Once a valid key has been pressed, its value is held in the variable A and testing starts at line 2080. The first piece of code repetitively subtracts 3 from the value of A to determine the horizontal displacement. If keys 1,4 or 7 have been pressed we must wish to move left; 3,6 and 9 indicate a move to the right and 2,5 and 8 maintain the current column position. As we are now left with a number between 1 and 3, we can determine the horizontal displacement by subtracting 2 from this remainder and this is done in 2090.

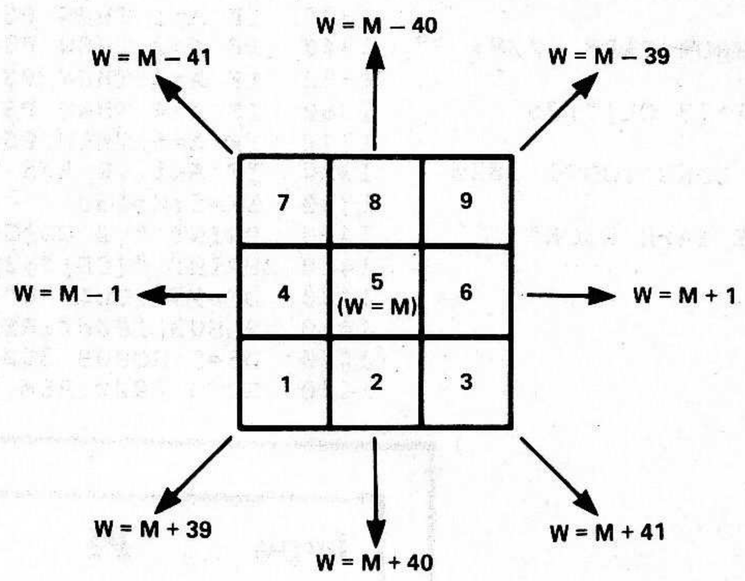


Fig. 1. The directions corresponding to the keys on a numeric pad together with their 40-column displacements.

All we have to do now is to determine the correct vertical displacement, this also being computed in line 2090. Assuming a 40 column screen we must now establish the row we wish to move to; this is best explained by referring to Fig. 2. If we subtract 1 from the key value (Fig. 2b) and then divide by 3 (Fig. 2c) the INTeger part of the remaining number is related to both the key value and its corresponding row (Fig. 2d). Subtract 1 and multiply by −40 and the resulting number is the vertical displacement.

Having computed the address of the position you wish to move to (held in variable W), we can now start to check what is in that position. These checks are preceded in line 2100 by incrementing the turns count and clearing away the movement message. The next block of lines inspects the contents of address W and is best shown as a small table:

Line	Character	Action
2110	Nothing	Jump to Movement
2120	Safe Castle	Jump to Quit routine
2130	Solid object	Try again!
2140	Scene code	Jump to Scene Control
2150	Scene exit	Jump to Scene Control
2160	Stairs	Jump to Stairs routine
2170	Lakes	Reverse the character code
2180	Special find	Jump to Special Finds

After all these checks have been performed it only remains to move your character into its selected position; line 2190 does this and, because you stepped on a blank square, the program now generates a random number to see if there is either a hidden 'find' or a waiting monster. It does this in lines 2200 to 2220 and, depending on the value of the random number, control may be passed to either the Monster Selection routine or to the Finds routine.

If nothing is found a suitable message is printed and the program loops back to the beginning of movement at 2010. The DF value of 80 in line 2230 enables you to read the message; if you have stepped on the path, checked in line 2200, the delay is only set to 5 because there is no message to read. **HB & PNG**

Scene	Object	PET	MZ80-K	TRS-80
Valley	Border	214	166	191
	Safe Castle	219	74	35
	Path 'up'	78	118	154
	Path 'down'	77	119	169
	Woods	216	80	87
	Swamps	173	42	83
	Tower	87	65	84
	Character	81	202	79
Woods	Border	96	*	128
	Trees	88	70	90
	Lake	224	163	191
	Vounim's	230	239	86
	Character	81	202	79
Swamps	Border	96	*	128
	Tufts	45	227	45
	Lake	224	163	191
	Y'Nagioth	230	239	89
	Character	81	202	79
Tower Vounim & Y'Nagioth	Border	160	67	191
	Walls	160	67	191
	Stairs	102	109	153
	Doorway	104	212	176
	Treasures	42	107	42
	Character	81	202	79

Table 3. These are the recommended POKE codes for the three systems the game has been implemented on. The borders around the Woods and Swamps for the MZ80-K (marked * in the Table) have to be PRINTed into place using the following characters: 50,51,60,61,112,113,114 and 115. This provides a neat border at the expense of extra entries in the look-up table.

Fig. 2. The mathematical sequence required to convert key value to row displacement.

7	8	9
4	5	6
1	2	3

(a) VAL (GC$)

6	7	8
3	4	5
0	1	2

(b) VAL (GC$) – 1

2	2.3	2.6
1	1.3	1.6
0	0.3	0.6

(c) (VAL (GC$) – 1) / 3

2	2	2
1	1	1
0	0	0

(d) INT((VAL (GC$) –1) / 3

1	1	1
0	0	0
–1	–1	–1

(e) INT ((VAL (GC$) – 1) / 3 –1

–40	–40	–40
0	0	0
+40	+40	+40

(f) – 40* (INT ((VAL (GC$) – 1) / 3) – 1)

THE VALLEY

```
1999  REM ** MOVEMENT ROUTINE
2000  M=W:PK=PEEK(W):POKE M,81
2010  C=C+10
2020  IF PK=77 OR PK=78 THEN 2040
2030  PRINT D$;"YOUR MOVE...WHICH DIRECTION ?":GOTO 2050
2040  PRINT D$;"SAFE ON THE PATH...WHICH WAY ?"
2050  FOR I=1 TO 10:GET GC$:NEXT I:REM ** CLEAR KBD BUFFER
2060  GET GC$:IF GC$="E" THEN 45000:REM ** EGO
2069  REM ** SPECIAL ROUTINE FOR NUMERIC KEYPADS
2070  A=VAL(GC$):IF A=0 THEN 2060
2080  IF A>3 THEN A=A-3:GOTO 2080
2090  W=M+A-2-40*(INT((VAL(GC$)-1)/3)-1)
2100  TN=TN+1:PRINT D$;SP$
2109  REM ** AM I STEPPING ON SOMETHING ?
2110  Q=81:Q1=PEEK(W):IF Q1=32 OR Q1=45 THEN 2190
2120  IF Q1=219 THEN 48000:REM ** QUIT
2130  IF Q1=214 OR Q1=160 OR Q1=88 THEN TN=TN-1:GOTO
      2030:REM ** HIT WALL OR TREE
2140  IF Q1=216 OR Q1=87 OR Q1=173 OR Q1=230 THEN 9000:
      REM ** SCENE ENTRY
2150  IF Q1=104 OR Q1=96 THEN 9090:REM ** SCENE EXIT
2160  IF Q1=102 THEN 15000:REM ** STAIRS
2170  IF Q1=224 OR (GC$="5" AND PK=224) THEN Q=209:
      C=C-20:IF C<=0 THEN 55000:REM ** WATER
2180  IF Q1=42 THEN 2800:REM ** SPECIAL FIND
2190  POKE M,PK:PK=PEEK(W):M=W:POKE M,Q
2200  IF PK=77 OR PK=78 THEN DF=5:GOTO 2250
2209  REM ** NOTHING, MONSTER OR FIND PERHAPS ?
2210  RF=RND(TI)
2220  IF RF<0.33 THEN 3000:REM ** MONSTER SELECT
2230  IF RF>0.75 THEN 2300:REM ** FIND SELECT
2240  PRINT D$;"NOTHING OF VALUE...SEARCH ON":DF=80
2250  GOSUB 36000:REM ** DELAY + UPDATE
2260  GOTO 2010
```

Finds

The 'ordinary finds' module starts lines 2300-2310 by randomly selecting one of four finds — three good, one not so good. A random integer between 1 and 6 is generated and two line numbers appear twice in the ON...GOSUB list thus giving probabilities of roughly 16%, 32%, 32% and 16% to the finds.

The first find, starting at 2340, is the bad one. Although line 2350 boosts combat strength by an amount dependent on FL, magical ability drops (FL again being a factor) and stamina is reduced by 20. Line 2360 jumps to the Death routine if C falls below zero.

At line 2380 'a hoard of gold' is found. Treasure is incremented by a value between 100 and 700, depending on FL and a random factor. The third and fourth finds are not monetary but physical and magical; although different messages are printed, lines 2140 and 2440.

Each of the four subroutines returns to line 2330 for a delay and update before control is returned to the Movement routine, line 2010.

PNG

```
2299  REM ** FINDS ROUTINE
2300  RF=INT(RND(TI)*6+1)
2310  ON RF GOSUB 2340,2380,2380,2410,2410,2440
2320  DF=80:GOSUB 36000:REM ** DELAY + UPDATE
2330  GOTO 2010
2340  PRINT D$;"A CIRCLE OF EVIL...DEPART IN HASTE !"
2350  CS=CS+INT((FL+1)/2):PS=PS-INT((FL+1)/2):C=C-20
2360  IF C<=0 THEN 55000:REM ** DEATH
2370  RETURN
2380  PRINT D$;"A HOARD OF GOLD"
2390  TS=TS+INT(FL*RND(TI)*100+100)
2400  RETURN
2410  PRINT D$;"YOU FEEL THE AURA OF THE DEEP MAGIC..."
2420  PRINT "[8 SPC]...ALL AROUND YOU..."
2430  GOTO 2450
2440  PRINT D$;"...A PLACE OF ANCIENT POWER..."
2450  PS=PS+2+INT(FL*P1):CS=CS+1+INT(FL*C1):C=C+25
2460  RETURN
```

Special Finds

If the Movement routine establishes that you have stepped onto an asterisk, a jump is made to the Special Finds module at line 2800. Here you are placed on the marker which is then erased (PK = 32); once you've picked up a special find, it's gone for good! Next, a random number is generated to decide what you've found; the 'Movement' message is wiped and a series of tests begins.

The first test (line 2820) succeeds if you're in Vounim's Lair, S = 6, have a full Amulet, T(1) = 6, a rating greater than 25, and have not already found the Helm of Evanna, T(2) = 0. You also have to be very lucky, RN > 0.95!

Line 2830 tests for the empty Amulet; this can only be found in the Temple of Y'Nagioth, S = 5, with RN > 0.85. You may only have one Amulet at a time, T(0) = 0. Note that although you need a full Amulet to obtain the Helm, losing the Amulet later (through reincarnation) means you are free to find another one; no problems arise if you already have the Helm.

The next line, 2840, checks for Amulet stones which only occur in the Black Tower, S = 4. Not only must you first have the Amulet, T(0) = 1, and space left in it, T(1) < 6, but you must be on a sufficiently high floor. The first stones can be found low down the Tower but as you find each stone you must venture higher to find the remaining ones, FL > T(1). Assuming you have found an Amulet stone, lines 2910-2920 decide whether it fits or not. Since RN must already be greater than 0.7 to get to these lines, the condition that RN > 0.85 here gives a 50-50 chance of the stone being the right one.

If the tests in lines 2820-2840 all fail then you have found either a precious stone or a worthless bauble. The random factor of 0.43 in line 2850 was chosen to get a long-term average of roughly 50% between precious stones and worthless baubles since RN may have been 'filtered' by the previous tests. For example, if all three tests failed on factors other than the value of RN, it could be anything between 0 and 1 on line 2850 and the probability is 43% that you have a worthless bauble. On the other hand, if line 2840 failed only because RN < 0.7 then the probability shifts to 0.43/0.7 = 61.4%.

As well as obtaining the objects themselves your treasure, TS, is updated in line 2930 by an amount which depends on the number of items you've already found. Baubles and wrong Amulet stones don't count and bypass this line. **PNG**

```
2799  REM ** SPECIAL FINDS ROUTINE
2800  POKE M,32:M=W:PK=32:POKE M,81
2810  RN=RND(TI):PRINT D$;SP$
2820  IF S=6 AND RN>0.95 AND T(1)=6 AND T(2)=0 AND RT>25
      THEN T(2)=1:GOTO 2870
2830  IF S=5 AND RN>0.85 AND T(0)=0 THEN T(0)=1:GOTO 2880
2840  IF S=4 AND RN>0.7 AND T(0)=1 AND T(1)<6 AND FL>T(1)
      THEN 2890
2850  IF RN>0.43 THEN PRINT D$;"A WORTHLESS BAUBLE":
      GOTO 2940
2860  PRINT D$;"A PRECIOUS STONE !":GOTO 2930
2870  PRINT D$;"YOU FIND THE HELM OF EVANNA !":GOTO 2930
2880  PRINT D$;"THE AMULET OF ALARIAN...EMPTY...":
      GOTO 2930
2890  PRINT D$;"AN AMULET STONE...":PRINT
2900  DF=60:DL$="D":GOSUB 36000:REM ** DELAY
2910  IF RN>0.85 THEN PRINT "[CD]...BUT THE WRONG ONE !":
      GOTO 2940
2920  PRINT "[CD]...THE STONE FITS !":T(1)=T(1)+1
2930  TS=TS+100*(T(0)+T(1)+T(2)+FL)
2940  DF=80:GOSUB 36000:REM ** DELAY + UPDATE
2950  GOTO 2010
```

Monster Selection

If you have the misfortune to draw a monster at the end of the Movement routine, control is passed to the segment of code which starts at 3000. A random number is generated between 1 and 16 and tested to see if it is greater than 9. This test is made to ensure that the stronger monsters cannot occur too frequently; the checks and limits for this are established in line 3020.

If the character is currently standing or swimming in a lake the choice of monsters is limited to the Water Imp and the Kraken, both prefixed L in the DATA.

The most unpleasant general monster is the Balrog and if he is drawn from the array, a further check is made in line 3040 to ensure that he appears less frequently.

Some monsters live only in the rarified heights of the Black Tower or one of the two special castle-type scenes and if these are drawn from the array, a further check is made in line 3050 to ensure that these conditions are met.

Once an acceptable monster has been selected from the array, the left-hand character of its name is stripped off to see if it can exist in the current scenario; this character is then checked against F$ in lines 3060 to 3090. If all is correct, the chosen beast is displayed on the screen and combat commences. The base strengths of each monster are held in arrays MS() and N1() and these values are further modified by the code between lines 3120 and 3170 to produce the actual strengths of the chosen monster. **HB**

```
2999  REM ** MONSTER SELECTION ROUTINE
3000  PRINT D$;"** BEWARE...THOU HAST ENCOUNTERED **"
3010  MS=0:N=0:CF=1
3020  RF=INT(RND(TI)*17):IF RF>9 AND RND(TI)>0.85
      THEN 3020
3030  IF Q1=224 OR PK=224 THEN RF=INT(RND(TI)*2+17)
3040  IF RF=16 AND RND(TI)<0.7 THEN 3020
3050  IF FL<5 AND RF=15 THEN 3020
3060  X$=LEFT$(M$(RF),1)
3070  FOR I=1 TO LEN(F$)
3080  IF MID$(F$,I,1)=X$ THEN 3110
3090  NEXT I
3100  GOTO 3020
3110  M$=RIGHT$(M$(RF),LEN(M$(RF))-1)
3120  IF MS(RF)=0 THEN 3150
3130  MS=INT((CS*0.3)+MS(RF)*FL^0.2/(RND(TI)+1))
3140  IF N1(RF)=0 THEN 3160
3150  N=INT(N1(RF)*FL^0.2/(RND(TI)+1))
3160  U=INT((RF+1)*(FL^1.5))
3170  IF RF>23 THEN U=INT((RF-22)*FL^1.5)
3180  PRINT "[CD]";LEFT$(R$,12-(LEN(M$))/2);"AN EVIL ";M$
3190  DF=40:GOSUB 36000:REM ** DELAY + UPDATE
```

Character's Combat

The action of fighting a monster can be broken down into three main sections; you hitting it, it hitting you and you casting a spell. The Spells are controlled by their own section of code that will be described later and can be simply treated as a jump out of the physical combat routines.

The Character's Combat section is located from 3570 to 3910 but before this can be executed we must determine whether you have surprised the beast or not. This is tested for in line 3500 where a random number is generated giving a 60/40 chance of you surprising the monster. If you do have surprise you are then offered the option of retreating from combat. The "R" key must be pressed within the time limit of the Combat Get routine or control passes directly to the Monster's Combat.

If you do choose to retreat a suitable message is displayed and the program goes back to the Movement routine. Choosing to attack, the message "*** Strike Quickly ***" is displayed and you have the choice of attacking its Head, Body or Limbs with the further option of trying to cast a Spell. If no key was pressed or the wrong one was chosen, control passes to the Monster's Combat via a suitable message. The control for this section of the combat is handled by lines 3570 to 3600.

Assuming that you have pressed a valid key the program checks in line 3630 to see if you wished to cast a spell and if so passes control to the Spell Control section. Before determining how much damage, if any, you have done to the beast, the program computes your current experience factor in line 3620 and deducts one stamina point. If you have exhausted yourself attempting to fight, the program detects this in 3660 and passes control to the Death routine.

Because each of the three target areas of the monster have different levels of vulnerability the code between 3670 and 3710 determines whether you hit the beast or not and, if you did, sets the damage factor variable, Z, to the appropriate level. As it is possible to strike a heavy blow which will leave the monster helpless we must first inspect the corresponding flag, HF, which tells us if the beast is certain to die on this attempt. If this flag has not been set in line 3730, we calculate the damage done to the monster and display it — it is possible to hit the monster yet do no damage! There are now several options available to us and these are sorted out by the rest of the routine from 3800 to 3910. The first alternative is that we have killed it, in which case we collect experience, reset the combat flags and go back to the movement routine (lines 3860 to 3890). Our second option is that we have done so much damage to the beast that it is unable to have another go at us. In this case we set the flag, HF, and go back to the "*** Strike Quickly ***" message. The remaining alternative is that we either did no damage or insufficient to cripple the monster and in both cases, control now passes to the Monster's Combat routine. **HB**


```
3499  REM ** CHARACTER'S COMBAT ROUTINE
3500  IF RND(TI)<0.6 THEN 4000:REM ** MONSTER'S COMBAT
3510  PRINT D$;"YOU HAVE SUPRISE...ATTACK OR RETREAT"
3520  GOSUB 1700:REM ** COMBAT GET
3530  IF GC$="R" THEN 3900
3540  IF TV=1 THEN 3600
3550  IF GC$<>"A" THEN 4000
3560  DF=30:DL$="D":GOSUB 36000:REM ** DELAY
3570  PRINT D$;"*** STRIKE QUICKLY ***"
3580  GOSUB 1700:REM ** COMBAT GET
3590  IF TV=0 THEN 3620
3600  PRINT D$;"* TOO SLOW...TOO SLOW *"
3610  HF=0:GOTO 3830
3620  E=39*LOG(EX)/3.14
3630  IF GC$="S" THEN 4500:REM ** SPELL CONTROL
3640  IF MS=0 THEN PRINT D$;"YOUR SWORD AVAILS YOU NOUGHT
      HERE":GOTO 3830
3650  C=C-1
3660  IF C<=0 THEN PRINT D$;"YOU FATALLY EXHAUST YOURSELF"
      GOTO 55000:REM ** DEATH
3670  RF=RND(TI)*10
3680  IF GC$="H" AND (RF<5 OR CS>MS*4) THEN Z=2:GOTO 3730
3690  IF GC$="B" AND (RF<7 OR CS>MS*4) THEN Z=1:GOTO 3730
3700  IF GC$="L" AND (RF<9 OR CS>MS*4) THEN Z=0.3:
      GOTO 3730
3710  PRINT D$;"YOU MISSED IT !"
```

```
3720  HF=0:GOTO 3830
3730  IF HF=1 THEN D=MS+INT(RND(TI)*9):HF=0:GOTO 3760
3740  D=INT((((CS*50*RND(TI))-(10*MS)+E)/100)*Z):IF D<0
      THEN D=0
3750  IF CS>(MS-D)*4 THEN HF=1
3760  MS=MS-D
3770  PRINT D$;"A HIT..."
3780  DF=60:DL$="D":GOSUB 36000:REM ** DELAY
3790  IF D=0 THEN PRINT D$;"[8 CR]BUT...NO DAMAGE":HF=0:
      GOTO 3830
3800  PRINT D$;"[8 CR]";D;" DAMAGE...":IF MS<=0 THEN
      3860:REM ** IT'S DEAD
3810  IF HF=1 THEN DF=30:DL$="D":GOSUB 36000:REM ** DELAY
3820  IF HF=1 THEN PRINT "[CD]THE ";M$;" STAGGERS
      DEFEATED"
3830  DF=110:GOSUB 36000:REM ** DELAY + UPDATE
3840  IF HF=1 THEN 3570
3850  GOTO 4000:REM ** MONSTER'S COMBAT
3860  PRINT D$;"[2 CD]...KILLING THE MONSTER..."
3870  EX=EX+U:HF=0:CF=0
3880  DF=80:GOSUB 36000:REM ** DELAY + UPDATE
3890  GOTO 2010:REM ** MOVEMENT
3900  PRINT D$;"KNAVISH COWARD !":CF=0
3910  GOTO 3880
```

Monster's Combat

Unlike the character, the monster has only two possible methods of attack. Its normal approach is to hit you but as magical monsters, ones with no physical strength, are unable to wield swords and the like, there is the option to attack you with a lightning bolt. To make life even more unpleasant this option is extended to a physical monster whose strength has fallen below its psi power!

The options for the monster are checked at the beginning of its routine, lines 4000 to 4040, before a random number is generated line 4050 determining the outcome. The random number can be between 1 and 10 and there are eight possible messages, two messages appearing twice. Depending on the value of the random number the appropriate message is selected by line 4060 and control is passed to the appropriate section of the routine.

If the monster has missed you or used up all its stamina in the attempt, the section between 4240 and 4290 takes the program back to the Character's Combat or the Movement routine respectively.

Just as your character can hit a selected area of the beast, the reverse is now true but the area is selected randomly by line 4060. Again, as in the Character's Combat, a damage factor Z is set to an appropriate value and the potential damage done to you 'G' is calculated by line 4160. This amount is deducted from your character's stamina by line 4180 and your health is then examined by line 4220; if you are now an ex-character, control is passed to the Death routine.

As mentioned earlier in this section the monster can throw a lightning bolt at you and if this option is selected by line 4030, control is passed to the section of program from 4300 to 4410. This computes the possibility of the lightning bolt hitting or missing and, if it hits, how much damage will result.

The outcome of the Monster's Combat routine is again a three-way option; it has killed you so the game jumps to Death; it has wounded but not killed (or missed completely) so control passes back to Character's Combat; or it has killed itself in which case control passes back to the Movement section. **HB**

```
3999  REM ** MONSTER'S COMBAT ROUTINE
4000  PRINT D$;"THE CREATURE ATTACKS..."
4010  DF=50:DL$="W":GOSUB 36000:REM ** DELAY + WIPE
4020  IF MS=0 THEN 4300:REM ** PSIONIC ATTACK
4030  IF MS<N AND N>6 AND RND(TI)<0.5 THEN 4300
4040  MS=MS-1:IF MS<=0 THEN 4240
4050  RF=INT(RND(TI)*10+1)
4060  ON RF GOTO 4070,4080,4090,4100,4110,4110,4120,4120,
      4130,4140
4070  PRINT D$;"IT SWINGS AT YOU...AND MISSES":GOTO 4280
4080  PRINT D$;"YOUR BLADE DEFLECTS THE BLOW":GOTO 4280
4090  PRINT D$;"...BUT HESITATES, UNSURE...":GOTO 4280
4100  Z=3:PRINT D$;"IT STRIKES YOUR HEAD !":GOTO 4150
4110  Z=1.5:PRINT D$;"YOUR CHEST IS STRUCK !":GOTO 4150
4120  Z=1:PRINT D$;"A STRIKE TO YOUR SWORDARM !":
      GOTO 4150
4130  Z=1.3:PRINT D$;"A BLOW TO YOUR BODY !":GOTO 4150
4140  Z=0.5:PRINT D$;"IT CATCHES YOUR LEGS !"
4150  DF=60:DL$="D":GOSUB 36000:REM ** DELAY
4160  G=INT((((MS*75*RND(TI))-(10*CS)-E)/100)*Z)
4170  IF G<0 THEN G=0:PRINT D$;"...SAVED BY YOUR
      ARMOUR ![2 SPC]":GOTO 4280
4180  C=C-G
4190  IF G>9 THEN CS=INT(CS-G/6)
4200  IF G=0 THEN PRINT D$;"SHAKEN......BUT NO DAMAGE
      DONE":GOTO 4280
4210  PRINT D$;"YOU TAKE...[6 SPC][6 CL]";G;" DAMAGE...
      [6 SPC]"
4220  IF CS<=0 OR C<=0 THEN 55000:REM ** DEATH
4230  GOTO 4280
4240  PRINT D$;"...USING ITS LAST ENERGY IN THE ATTEMPT"
4250  EX=INT(EX+U/2):CF=0
4260  DF=100:GOSUB 36000:REM ** DELAY + UPDATE
4270  GOTO 2010:REM ** MOVEMENT
4280  DF=100:GOSUB 36000:REM ** DELAY + UPDATE
4290  GOTO 3570:REM ** CHARACTER'S COMBAT
4299  REM ** MONSTER'S PSIONIC ATTACK
4300  PRINT D$;"...HURLING A LIGHTNING BOLT AT YOU !"
4310  G=INT(((180*N*RND(TI))-(PS+E))/100):N=N-5:IF G>9
      THEN N=N-INT(G/5)
4320  DF=80:DL$="W":GOSUB 36000:REM ** DELAY + WIPE
4330  IF N<=0 THEN N=0:GOTO 4240
4340  IF RND(TI)<0.25 THEN 4410
4350  IF G<=0 THEN G=0:GOTO 4400
4360  PRINT D$;"IT STRIKES HOME !"
4370  DF=110:GOSUB 36000:REM ** DELAY + UPDATE
4380  C=C-G:IF G>9 THEN PS=INT(PS-G/4)
4390  GOTO 4210
4400  PRINT D$;"YOUR PSI SHIELD PROTECTS YOU":GOTO 4280
4410  PRINT D$;"...MISSED YOU !":GOTO 4280
```

Spell Control

If the option of using a spell is chosen during the Combat routine, control jumps to the routine located from 4500. An initial message is displayed asking which spell you wish to cast and the player's reply is collected by the Combat Get routine.

If you don't press a key within the allotted time the program jumps back to the "Too Slow" message and you have missed your chance; this test is performed in line 4510.

As there are only three spells available in the 16K version (we've left plenty of room for expansion) a check is made at line 4520 to ensure that the key you pressed is valid and if not, a suitable message is printed and control passes back to the Combat routine through line 4640.

Given that you have pressed a valid key, line 4540 now checks to see if you are strong enough to use the chosen spell; if not, control jumps to line 4590 with a suitable message and, once again, the program goes back to the Combat routine via line 4640.

If you meet the requirements to use the spell the appropriate subroutine is selected by line 4550 and control passes to the chosen spell subroutine.

On RETURNing from the spell subroutine a flag variable, SC, will have been set to a value betwen 1 and 7 and this value represents the outcome of casting the spell. Line 4560 causes the program to jump to the correct message and result. **PF**


```
4499  REM ** SPELL CONTROL ROUTINE
4500  PRINT D$;"WHICH SPELL SEEK YE ? ":GOSUB 1700:REM **
      COMBAT GET
4510  IF TV=1 THEN 3600:REM ** TOO SLOW
4520  IF VAL(GC$)>0 AND VAL(GC$)=<3 THEN 4540
4530  PRINT D$;"NO SUCH SPELL...[5 SPC]":GOTO 4640
4540  IF 4*PS*RND(TI)<=N THEN 4590
4550  ON VAL(GC$) GOSUB 5000,5200,5400
4559  REM ** SC CONTAINS OUTCOME FLAG
4560  ON SC GOTO 4620,4640,4660,4570,4600,4580,4590
4570  PRINT D$;"IT IS BEYOND YOU[5 SPC]":GOTO 4640
4580  PRINT "BUT THE SPELL FAILS...!":GOTO 4640
4590  PRINT D$;"NO USE, THE BEAST'S PSI SHIELDS IT":
      GOTO 4640
4600  PRINT D$;"THE SPELL SAPS ALL YOU STRENGTH"
4610  GOTO 55000:REM ** DEATH
4620  DF=100:GOSUB 36000:REM ** DELAY + UPDATE
4630  GOTO 2010:REM ** MOVEMENT
4640  DF=60:GOSUB 36000:REM ** DELAY + UPDATE
4650  GOTO 4000:REM ** MONSTER'S COMBAT
4660  DF=60:GOSUB 36000:REM ** DELAY + UPDATE
4670  GOTO 3570:REM ** CHARACTER'S COMBAT
```

The Spells

SLEEPIT: The first checks made in this subroutine are in line 5000 where stamina is deducted and, if your total has dropped below 0, the program is RETURNed via line 4560 to the Death routine. If your stamina is still healthy the message for the spell is printed by lines 5010 to 5050 moving on to line 5060 to see if the spell actually worked.

The possibility of the spell working is 50% and if successful the program prints the welcome message and updates your experience before going back to the Movement routine via line 4560. If it fails the program still RETURNs via 4560 but the value of the SC flag variable passes control to the Monster's Combat.

PSI-LANCE: An initial check is made in line 5200 to see if the character has met the requirements for the spell to function, if not the program RETURNs to 4560 and then to the Monster's Combat after displaying a suitable message.

Once again the cost to the character in stamina is deducted, this time in line 5210, and the check made to see if he has exhausted himself is done. If you are still alive and fighting, the check is made in line 5220 to see if you are really attacking a monster with some psi power, if not then you are wasting your time so control passes back via 4560 to the Monster's Combat.

If all is well at this stage the text for the spell is printed by lines 5230 and 5240 and the test to see if the spell was successful is then made at line 5250. If the spell failed the program RETURNs to line 4560 and the Monster's Combat after printing a suitable message.

The amount of damage your spell did to the monster is calculated in 5260 and if no damage was inflicted the program goes back to 4560 and the Monster's Combat. If you did damage the creature the amount inflicted is displayed and deducted from monster's strength and psi power and these values are then checked to see if the monster has expired by line 5310. If the monster is still alive the program returns to Monster's Combat via 4560 otherwise line 5320 tells you that you have killed the beast and your experience is subsequently updated by line 5330 and the program RETURNs to 4560 and then back to the Movement routine.

CRISPIT: Once again an initial test is made to see if your character meets the requirements to use the spell, if not, it's back to line 4560 and then to the Monster's Combat. If you can use the spell line 5410 tests to see if the stamina you use to cast the spell has killed you off; if it has it passes control to Death via 4560.

The spell message is printed by lines 5420 to 5470 before line 4580 computes the outcome of the spell. Once again, if the spell failed it RETURNs to 4560 and then to the Monster's Combat.

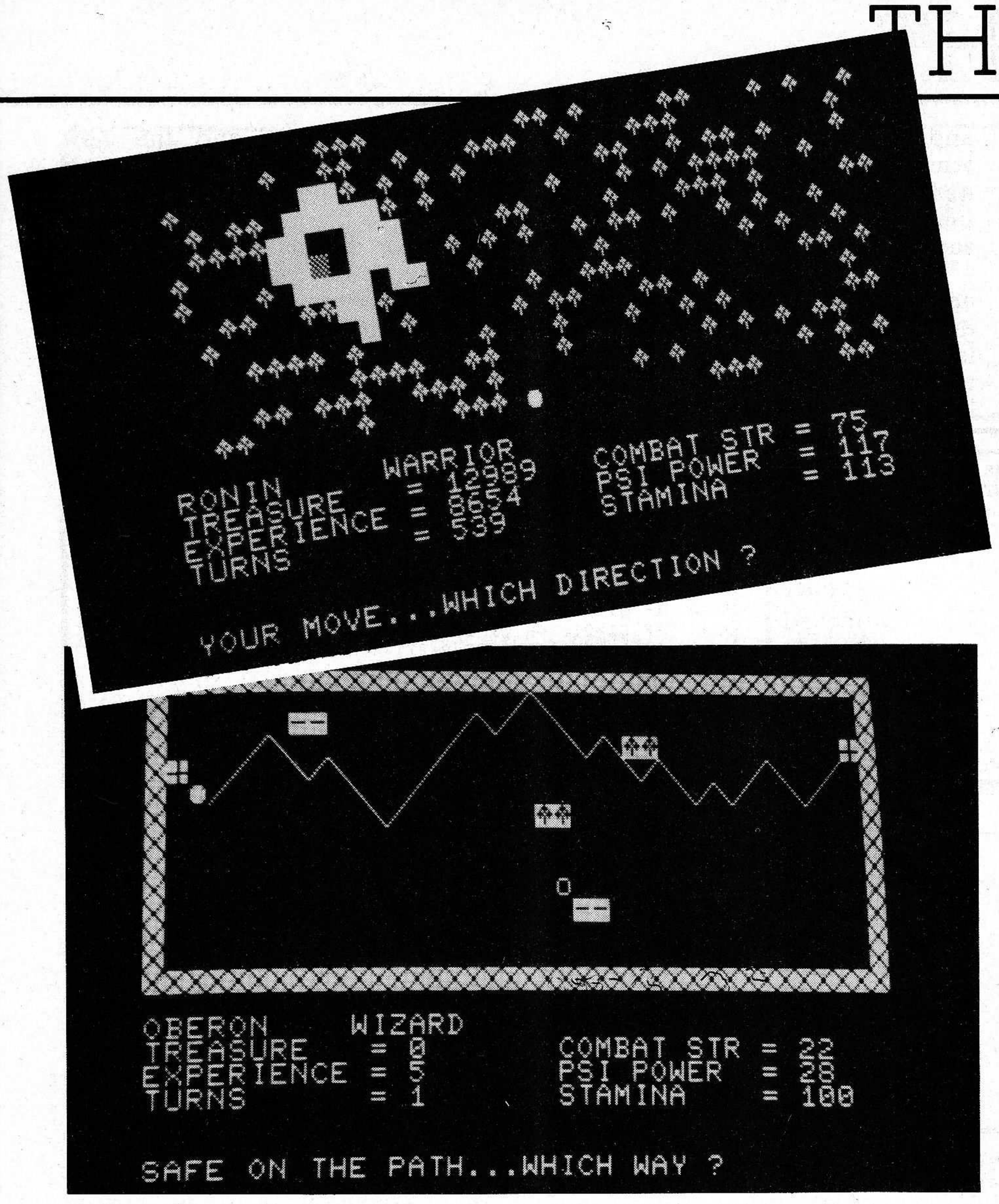
If the spell worked, line 5490 calculates the damage done; if none, then it's back to 4560 and the Monster's Combat again! Given that the spell has worked, the damage is deducted from the monster's physical strength and if it hasn't got any it is deducted from its psi power; all this is handled by lines 5510 to 5530. In fact, if you do more than a certain amount of damage, line 5520 takes some extra points off the psi power as well.

The rest of the routine is concerned with printing out the amount of damage done and checking to see if the monster is now dead. Depending on the result of these tests the program can jump to either the Movement routine or the Monster's Combat via the ever-faithful 4560. **PF**

```
4999  REM ** SPELL 1 (SLEEPIT)
5000  C=C-5:IF C<=0 THEN SC=5:RETURN
5010  PRINT D$;"SLEEP YOU FOUL FIEND THAT I MAY ESCAPE"
5020  PRINT "AND PRESERVE MY MISERABLE SKIN"
5030  DF=180:GOSUB 36000:REM ** DELAY + UPDATE
5040  PRINT D$;"THE CREATURE STAGGERS..."
5050  DF=40:DL$="D":GOSUB 36000:REM ** DELAY
5060  IF RND(TI)<0.5 THEN 5090
5070  PRINT "AND COLLAPSES...STUNNED"
5080  EX=INT(EX+U/2):CF=0:SC=1:RETURN
5090  PRINT "BUT RECOVERS WITH A SNARL !"
5100  SC=2:RETURN


5199  REM ** SPELL 2 (PSI-LANCE)
5200  IF MS>C OR PS<49 OR EX<1000 THEN SC=4:RETURN
5210  C=C-10:IF C<=0 THEN SC=5:RETURN
5220  IF N=0 THEN PRINT D$;"THIS BEAST HAS NO PSI TO
      ATTACK":SC=2:RETURN
5230  PRINT D$;"WITH MY MIND I BATTLE THEE FOR MY LIFE"
5240  DF=120:GOSUB 36000:REM ** DELAY + UPDATE
5250  RF=RND(TI):IF RF<0.4 AND N>10 THEN SC=6:RETURN
5260  D=INT((((CS*50*RF)-5*(MS+N)+E)/50)/4)
5270  IF D<=0 THEN D=0:SC=7:RETURN
5280  PRINT D$;"THE PSI-LANCE CAUSES ";D*2;" DAMAGE"
5290  N=N-3*D:IF N<=0 THEN N=0
5300  MS=MS-D:IF MS<=0 THEN MS=0
5310  IF (MS+N)>0 THEN SC=2:RETURN
5320  PRINT "[CD]...KILLING THE CREATURE"
5330  EX=EX+U:CF=0:SC=1:RETURN


5399  REM ** SPELL 3 (CRISPIT)
5400  IF PS<77 OR EX<5000 THEN SC=4:RETURN
5410  C=C-20:IF C<=0 THEN SC=5:RETURN
5420  PRINT D$;"WITH THE MIGHT OF MY SWORD I SMITE THEE"
5430  PRINT "WITH THE POWER OF MY SPELL I CURSE THEE"
5440  PRINT "BURN YE SPAWN OF HELL AND SUFFER..."
5450  DF=240:GOSUB 36000:REM ** DELAY + UPDATE
5460  PRINT D$;"A BOLT OF ENERGY LASHES AT THE BEAST..."
5470  DF=80:DL$="W":GOSUB 36000:REM ** DELAY + WIPE
5480  IF RND(TI)>(PS/780)*(5-P1) THEN PRINT D$;"MISSED
      IT !":SC=2:RETURN
5490  D=INT((CS+PS*RND(TI))-(10*N*RND(TI)))
5500  IF D<=0 THEN D=0:SC=7:RETURN
5510  IF MS=0 THEN N=N-D:GOTO 5530
5520  MS=MS-D:IF D>10 THEN N=INT(N-(D/3))
5530  PRINT D$;"IT STRIKES HOME CAUSING ";D;" DAMAGE
      [2 SPC]!"
5540  IF (MS+N)<=0 THEN 5570
5550  DF=80:DL$="D":GOSUB 36000:REM ** DELAY
5560  SC=2:RETURN
5570  PRINT "[CD]THE BEAST DIES SCREAMING !"
5580  EX=EX+U:CF=0:SC=1:RETURN
```



Scene Control

Having drawn the Valley, the Path and the scenario positions, our hero is now free to wander where he chooses. A monster may kill him but with a little luck and fast reflexes, he will sooner or later enter one of the scenarios, or having entered may wish to escape!

To do this he simply moves onto the scenario or exit symbol. The part of the program concerned with movement, lines 2000-2240, detects the symbol and the program is directed to Scene Control at line 9000 or Scene Exit at line 9090.

A check is made at line 9000 to see if you are attempting to enter one of the secondary scenarios direct from the water, POKE code 224. If you are, the program is directed back to Movement via line 9110 which inhibits the turn count and the stamina refresh.

The two scene building arrays, P() and N(), are dealt with in lines 9010-9050. Array P() is used to fix the pattern of rooms on different levels of a scenario and if the scenario has more than one level, each level retains the same room pattern while you remain in that scenario. Array P() has all previous values zeroed.

Array N() has a series of random numbers, integers between 4 and 8, assigned to it (5 is not permitted as it can produce an unacceptable pattern). This array determines the depth of the rooms in the primary and secondary castle-type scenarios. Vounim's Lair and the Temple of Y'Nagioth (pronounced Ee-nag-ee-oth).

Entering a scenario, tested for in line 9060, from the Valley, MP is assigned your last position in the Valley, M, so that when you leave the scenario you will return to the same position.

A random integer between 1 and 30 is generated in line 9070 and is assigned to P(2). This is used in all scenarios to determine the position of lakes or the patterns of rooms. Line 9080 sets a temporary variable TF to the number of turns you have had so far, TN. This is then used when you try to exit from a scenario testing whether you have been 'in' for a certain number of turns. The line then directs the program past the initial portion of the 'scene exit' to line 9130.

Moving on to line 9090, a test is made to see if you have remained within a scenario for a minimum amount turns. Thus if the turns count is equal to or larger than the generated random number, the program jumps to line 9130. If not, you are barred from exiting the scenario and line 9110 inhibits the turns count and the stamina refresh. The program is then directed back to the Movement routine at line 9120.

With the Valley still displayed on the screen, line 9130 POKEs a space into your present position and POKEs your character symbol to the position of the scenario symbol, W is the position you are about to go to . . .

The lines 9140-9220 check to see what scenario or exit symbol has been stepped on thus determining the scene change required; Q1 represents the POKE code of position W.

If the symbol which frames the Swamps or Woods is found, for the PET we use a shifted space . . .POKE code 96, as specified in line 9140, then you must be leaving the scene and entering the Valley, S = 1:F1 = 1. Line 9150 detects gateways, Q1 = 104, in the Black Tower, S = 4, denoting that you are leaving the Tower and entering the Valley.

If a gateway is detected in the Temple, S = 5, or the Lair, S = 6, line 9160 directs the program to allow you to enter the Swamps, S = 2, or the Woods, S = 3, respectively. These secondary scenarios are only found on a lake in these scenarios but in both cases the scene number of the Swamps or Woods is 3 below their secondary scenarios, S = S − 3. FL is determined in a similar manner, FL = FL − 4, and your position in the Swamps or Woods is reset to your original position outside the secondary scenario, M = MW.

A check is made at line 9170 to see if you have entered the Swamps, setting the scene number, S = 2, and the level , FL = 2, if you have.

Another check is made at line 9190 for either Swamps or Woods to assign two string variables, D2$ and R2$, if either of these scenarios are entered. Both D2$ and R2$ are cursor control movements: D2$

being a random number of Cursor Downs between 0 and 9 and R2$ being a random number of Cursor Rights between 1 and 30; this is the first use of P(2).

In line 9210, there is a check for one of the secondary scenarios in the Swamps or the Woods. If the Lair or Temple symbol, POKE code 230 is recognised, the appropriate scene number is assigned S=S+3 and level number FL=FL+4. The temporary position variable, MW, is assigned your position, M, immediately prior to entering the secondary scenario.

The program is directed to the appropriate subroutine is line 9220 and then in line 9230, the program is directed to the delay and status update subroutine. On RETURNing, line 9240 takes you back to the Movement routine and the new scenario is displayed awaiting further exploration.

To give the maximum variety of scenes for the minimum amount of memory space used we chose, for the 16K game, two primary scenarios in addition to the Valley itself. We then used variations of these basic scenarios to provide additional areas to explore. **PF**

```
8999  REM ** SCENARIO CONTROL ROUTINE
9000  IF Q1=230 AND PK=224 THEN PRINT D$;"YOU CANNOT
      ENTER THIS WAY...":GOTO 9110
9010  FOR I=2 TO 7
9020  P(I)=0
9030  N(I)=INT(RND(TI)*5+4)
9040  IF N(I)=5 THEN 9030
9050  NEXT I
9060  IF S=1 THEN MP=M
9070  P(2)=INT(RND(TI)*30+1)
9080  TF=TN:GOTO 9130
9089  REM ** EXIT FROM SCENARIO
9090  IF TN>TF+INT(RND(TI)*6+1) THEN 9130
9100  PRINT D$;"THE WAY IS BARRED"
9110  TN=TN-1:C=C-10:DF=100:DL$="W":GOSUB 36000:
      REM ** DELAY + WIPE
9120  GOTO 2010
9130  C=C-10:POKE M,32:POKE W,Q
9140  IF Q1=96 THEN S=1:FL=1
9150  IF Q1=104 AND S=4 THEN S=1:FL=1
9160  IF Q1=104 AND S=5 OR S=6 THEN S=S-3:FL=FL-4:M=MW
9170  IF Q1=173 THEN S=2:FL=2
9180  IF Q1=216 THEN S=3:FL=3
9190  IF Q1=216 OR Q1=173 THEN D2$=LEFT$(D$,INT(RND(TI)*
      10)):R2$=LEFT$(R$,P(2))
9200  IF Q1=87 THEN S=4:FL=2
9210  IF Q1=230 THEN S=S+3:FL=FL+4:MW=M
9220  ON S GOSUB 10000,12000,12010,14000,14010,14010
9230  DF=5:GOSUB 36000:REM ** DELAY + UPDATE
9240  GOTO 2000:REM ** MOVEMENT
```

The Valley

Let us start where our 'alter ego', whatever his character type, will step out into his adventure... the Valley. Line 10000 clears the screen, sets F$ to 'VAEGH' which determines what monsters may be found in the Valley, ie monsters from groups V,A,E,G, and H (see Table 1). The difficulty level is set to 1 (FL has a bearing on how strong the monsters are) and finally sets the scene number also to 1 ...the Valley.

First let us draw the bounds of the Valley, this consists of a rectangle 39 characters wide by 14 characters high. Lines 10010 and 10050 draw in the top and bottom frames of the Valley, between them a FOR...NEXT loop draws the vertical frames consisting of the appropriate characters separated by 37 spaces. These are drawn 12 times giving an internal playing area of 37 x 12.

Line 10070 determines the position, M, of the left-hand safe castle, 32809 is the screen map position of the top left-hand corner of our frame.

Line 10080 assigns the position and character code for the left-hand safe castle to array elements G(0) and G(1) respectively. L and MP are temporary position variables. M is used throughout as your position ***now*** and W is your position when you next move (the look-ahead variable). All are set to position of the safe castle.

We now have to work out the course of the path; for the PET we use the two diagonal lines, POKE codes 77 and 78 (suggested symbols for other micros are given in Table 3). Using only these two symbols for the path enables us to make a fairly simple decision on how the path may be drawn:

1) If the path is already slanting up to the right then only two possibilities are acceptable; upwards to the North East or a downwards to the East (remember the path must be continuous).
2) If the path is already slanting down to the right then only downwards to the South East or upwards to the East are permissible.

The choice of an upwards or downwards diagonal is made randomly in line 10100. Lines 10110 and 10120 initially set the POKE code, PC, for a downwards diagonal to be to the South East of the present path (L1=L+41) and for an upwards diagonal to be to the North East (L1=L−39). L1 is a temporary position variable.

In line 10130 we check to see if the path is trying to go through the top or bottom frames of the Valley. If it is, the program is directed back to line 10100 to choose again. If it is within the Valley, we allocate in line 10140 an element of array G() to the POKE code for that section of the path. Looking back to 1) and 2) above, you will notice that if the next path element is different to the previous element it ***must*** always be to the East. This condition is checked in line 10150 and the temporary variable, L1, is altered if necessary.

Line 10160 stores the position of the path in array G(), assigns the starting position for the next section of the path to variable L and draws the section just computed on the screen.

This selection of path direction and position is repeated 36 times within a FOR...NEXT loop. Positions 1 and 37 are safe castles from which you may leave the Valley if you so wish. Even elements of G() are screen positions and odd elements are POKE codes for the safe castles or the path. Line 10180 completes the 'path draw' by placing the right-hand safe castle on the last path position.

Going back to line 10060 we see that this tests to see if the path has already been computed. If it has not then G(0) will be zero. If it has, the program skips the path computation and jumps to line 10190 which initiates a FOR...NEXT loop that draws the path using the information stored in array G(). In fact if the path is being drawn for the first time (on entering the Valley) it is also redrawn in lines 10190 to 10210.

Having drawn the Valley path we now have to work out where the different scenarios are to be placed within the Valley. These positions are stored in array S(). Line 10220 checks if they have already been assigned and if they have, the program jumps to line 10280 where the scenarios are drawn on the map of the Valley.

Line 10240 generates two random numbers that represent row and column positions within the

Valley. Line 10250 assigns this position to an element of array S(). Line 10260 checks to see that the chosen position and the position immediately to its right are not the path or safe castle (ie an empty space POKE code 32). If the positions chosen are free then the selection is repeated for the next element of array S().

Lines 10280-10300 POKE the scenario symbols on the screen. There is a chance that some symbols may be overwritten by other scenarios but then this is the luck of the draw! Woods are drawn first (code 216) then Swamps (code 173) and finally the Black Tower of Zaexon (pronounced Zeeks-on), code 87. Woods and Swamps are represented by two symbols side by side and both scenarios are repeated. The Black Tower is a single symbol; only one such Tower is found in the Valley.

Line 10310 assigns your present position to that of temporary variable, MP. This is used to remember your last position in the Valley when you return to the Valley from one of the other scenarios. **PF**

```
 9999  REM ** SCENARIO 1 (THE VALLEY)
10000  PRINT "[CLS]":F$="VAEGH":FL=1:S=1
10009  REM ** DRAW THE VALLEY FRAME
10010  PRINT "[HOM][REV][39^V][OFF]"
10020  FOR I=1 TO 12
10030  PRINT "[REV][^V][OFF][37 SPC][REV][^V][OFF]"
10040  NEXT I
10050  PRINT "[REV][39^V][OFF]"
10059  REM ** IF PATH ALREADY DRAWN SKIP
10060  IF G(0)<>0 THEN 10190
10069  REM ** COMPUTE THE PATH
10070  M=32809+(INT(RND(TI)*11+1)*40)
10080  L=M:MP=M:W=M:G(0)=M:G(1)=219
10090  FOR I=2 TO 72 STEP 2
10100  IF RND(TI)>0.5 THEN 10120
10110  PC=77:L1=L+41:GOTO 10130
10120  PC=78:L1=L-39
10130  IF L1>=33286 OR L1<=32806 THEN 10100
10140  G(I+1)=PC
10150  IF I>2 AND G(I+1)<>G(I-1) THEN L1=L+1
10160  G(I)=L1:L=L1:POKE G(I),G(I+1)
10170  NEXT I
10180  G(73)=219
10189  REM ** PLOT IN PATH
10190  FOR I=0 TO 72 STEP 2
10200  POKE G(I),G(I+1)
10210  NEXT I
10220  IF S(0)<>0 THEN 10280
10229  REM ** COMPUTE SCENARIO POSITIONS
10230  FOR I=0 TO 4
10240  N1=INT(RND(TI)*11)+1:N2=INT(RND(TI)*34)+1
10250  S(I)=32809+(40*N1)+N2
10260  IF PEEK(S(I))<>32 OR PEEK(S(I)+1)<>32 THEN 10240
10270  NEXT I
10279  REM ** PLOT IN SCENARIOS
10280  POKE S(0),216:POKE S(0)+1,216:POKE S(1),216:POKE
       S(1)+1,216
10290  POKE S(2),173:POKE S(2)+1,173:POKE S(3),173:POKE
       S(3)+1,173
10300  POKE S(4),87
10310  M=MP:W=M
10320  RETURN
```

Woods And Swamps

These two scenarios are fundamentally the same, only the characters representing their contents and the special scene in the middle of the Lake differ. The starting point selected by the Scene Control routine is 12000 for the Swamp and 12010 for the Woods. The function of each of these entry points is to establish the valid monster string, F$, and to assign the POKE code, PC, to the correct value for the scene. The routine proper begins by re-assigning the POKE value of the square under your feet to 32, a space, so that when you move away you don't leave the character you were standing on in the Valley behind you.

The screen is now cleared in preparation for the construction of the scene. A pointer variable, L, is set to the value of the top left-hand corner of the scene and a random FOR...NEXT loop inserts 200 appropriate scene characters into the screen area. This is all handled by the block code from 12040 to 12070.

Having constructed the basis of the scenario we now need to plot in the Lake containing the secondary scenario and this is performed by the block code from 12080 to 12140. The values of D2$ and R2$ were previously determined in the Scenario Control section and serve to position the Lake within the scene area. The castle-type scene at the centre of the Lake is dependent on whether you are currently in the Woods, Vounims' Lair, or the Swamps, The Temple of Y'Nagioth.

Having now constructed the inside of the scene, we have to print a border around it in order to detect an attempt at movement outside the scene area. The border is printed by the section of program between 12150 and 12190 and is made up of non-32 type spaces on the PET, other suggestions are to be found in Table 3. Because the border is printed in after the scene has been constructed, it will overwrite any stray scene characters in the 1st and 40th column; this simplifies the random printing routine. The semicolons at the end of of lines 12150 and 12170 ensures that the frame is continuous; printing into the 40th column would otherwise force a Carriage Return.

The next operation is to POKE a space character to the position in which the character will appear and assign W to the address of that position. This ensures that when you enter the Woods you are not completely hemmed in by trees; it doesn't matter in the case of the Swamps.

Because you could re-enter the Woods and Swamps from one of the secondary scenes, a check is made in line 12210 to see if the POKE code of the square you were standing on was a doorway. If it was, your current position is reset to the position you were in when you entered that secondary scene. The value of that position is then held in the variable MW and is assigned when you enter a secondary scene. **PF**

```
11999  REM ** SCENARIO 2 (WOODS AND SWAMPS)
12000  F$="AFL":PC=45:GOTO 12020
12010  F$="FAEHL":PC=88
12020  PK=32
12030  PRINT "[CLS]"
12039  REM ** DRAW RANDOM WOODS OR SWAMPS
12040  L=32810
12050  FOR I=1 TO 200
12060  POKE L+INT(RND(TI)*515),PC
12070  NEXT I
12079  REM ** PRINT IN LAKE
12080  PRINT "[HOM]";D2$;R2$;"[2 CR][REV][2 SPC][OFF]"
12090  PRINT R2$;"[CR][REV][5 SPC][OFF]
12100  PRINT R2$;"[REV][2 SPC][OFF][2 SPC][REV][2 SPC]
       [OFF]"
12110  PRINT R2$;"[REV][2 SPC][^&][OFF][SPC][REV][3 SPC]
       [OFF]"
12120  PRINT R2$;"[CR][REV][4 SPC][OFF][CR][REV][2 SPC]
       [OFF]"
12130  PRINT R2$;"[3 CR][REV][2 SPC][OFF]"
12140  PRINT R2$;"[4 CR][REV][SPC][OFF]"
12149  REM ** DRAW IN THE FRAME
12150  PRINT "[HOM][40^SPC]";
12160  FOR I=1 TO 13
12170  PRINT "[^SPC][38 CR][^SPC]";
12180  NEXT I
12190  PRINT "[40^SPC]"
12200  POKE 33306,32:W=33306
12210  IF Q1=104 THEN M=MW:W=M
12220  RETURN
```

The Black Tower

Our other primary scenario is the Black Tower of Zaexon. This is a six floor castle-type scene and its construction is also used to produce the secondary single floor scenes found in the Woods and Swamps. The Tower has a stable floor pattern; once a floor has been entered it will remain the same as long as you are in the Tower.

Scenario Control directs the program to the routine at 14000 assigning the monster string, FS; zeroing the floor pattern variable, P; and setting the room depth variable, H, to the current FLth element of array N(). The current position character is set to a space and the program jumps to 14020. The variables for the secondary scenes are initialised in 14010. One slight change is that the array P() is set to the value of P(2). This is done because the secondary scenes have initial FL values of 6 or 7 depending on type and these elements of P() are 0 which would cause the room pattern to be the same each time (see line 14070).

The frame of the Tower is printed first by lines 14020 to 14060 using a reversed space character (Table 3 holds alternatives for other systems). The vertical walls are drawn next by the somewhat complex routine found between 14070 and 14250. In order to ensure that the pattern of rooms varies on each floor and on each visit to the scene, we use the 31 element data statement at line 60000. These are READ sequentially for each new floor and represent the width of each room. To give variety to the pattern of rooms the starting point of the READ is determined randomly in the Scenario Control section and stored in P(2). To start the drawing sequence the DATA pointer is RESTOREd and then P(2) dummy READs are made; V is used only as a temporary store. Once again we use the pointer variable, L, to hold the address of the top left-hand corner of the scene. We now read the next three DATA items from the list and store them in array D(); the number of sets of 3 is stored in the temporary variable, P.

The actual drawing of the vertical walls is done by lines 14150 to 14240 and their length is dependent on the value of H, the room depth variable. The wall characters are POKEd into position as are the doors which occur a predetermined distance along them. Having drawn the first set of vertical walls the starting point is re-assigned in line 14250 and the next set is drawn in — this process is repeated until the walls have reached the bottom of the frame.

The horizontal walls can now be drawn in and their spacing is dependent on the value of the current element of array N(). The routine is located between lines 14270 and 14340.

As only the Black Tower has stairs, line 14350 causes the secondary scenes to skip over this section of the program. The Black Tower has stairs located in opposite corners for each floor and these are POKEd into position on lines 14360 and 14370. If you are on the ground floor of the Tower or in one of the secondary scenes, a doorway is POKEd into position by line 14380.

If you are stepping into the Tower or either of the secondary scenes for the first time, your character will be placed just inside the doorway; the check for this is made in 14390 as P(3) will only be 0 if you haven't gone up any stairs yet.

The appropriate name for the castle-type scene is PRINTed into position by lines 14400 to 14480 and, in the case of the Tower, the floor number is also displayed.

Treasure can be found in the upper floors of the Tower, and either of the two secondary scenes, provided the value of FL is equal to or greater than 4 ***and*** a random factor is greater than 0.3. If these conditions are not met, control returns to the Scenario Control section and then back to the Movement routine. If both conditions are met, a random number of special treasure symbols are displayed; between 2 and 6 can appear and are shown as asterisks. They are positioned by the two temporary variables N1 and N2 which act as row and column co-ordinates. Provided the position selected is vacant an asterisk is POKEd into place. **PF**

```
13999  REM ** SCENARIO 3 (CASTLE-TYPES)
14000  F$="CAGE":P=0:H=N(FL):PK=32:GOTO 14020
14010  F$="CBE":P=0:H=N(FL):PK=32:P(FL)=P(2)
14019  REM ** DRAW FRAME
14020  PRINT "[CLS][REV][2 CR][21 SPC][OFF]"
14030  FOR I=1 TO 13
14040  PRINT "[REV][2 CR][SPC][OFF][19 SPC][REV][SPC]
       [OFF]"
14050  NEXT I
14060  PRINT "[REV][2 CR][21 SPC][OFF]"
14069  REM ** DRAW VERTICAL WALLS
14070  RESTORE:FOR I=1 TO P(FL)
14080  READ V:IF V=100 THEN RESTORE
14090  NEXT I
14100  L1=32810
14110  FOR J=1 TO 3
14120  READ D(J):P=P+1
14130  IF D(J)=100 THEN RESTORE:D(J)=3:P=P+1
14140  NEXT J
14150  FOR I=0 TO H:PC=160
14160  L=L1+(40*I):IF L>33290 THEN 14260
14170  IF I=1 THEN PC=32
14180  IF D(1)=0 THEN PC=160:GOTO 14200
14190  POKE L+D(1),PC:PC=160
14200  IF I=3 THEN PC=32
14210  POKE L+D(1)+D(2),PC:PC=160
14220  IF I=4 THEN PC=32
14230  POKE L+D(1)+D(2)+D(3),PC:PC=160
14240  NEXT I
14250  L1=L1+(40*H)+40:GOTO 14110
14259  REM ** DRAW HORIZONTAL WALLS
14260  L1=32810
14270  FOR J=1 TO 4
14280  L=L1+(40*J*(H+1))
14290  FOR K=1 TO 19
14300  IF L>33250 THEN 14350
14310  POKE L+K,PC
14320  IF K=2 OR K=3*H OR K=17 THEN POKE L+K,32:
       POKE L+K-40,32:POKE L+K+40,32
14330  NEXT K
14340  NEXT J
14349  REM ** DRAW IN THE STAIRS
14350  IF S=5 OR S=6 THEN 14380
14360  IF FL/2=INT(FL/2) THEN POKE 33291,102:GOTO 14380
14370  POKE 32829,102
14379  REM ** DOORWAY NEEDED ?
14380  IF FL=2 OR S=5 OR S=6 THEN POKE 33336,104:
       POKE 33296,32
14390  IF P(3)=0 THEN W=33296
14399  REM ** WRITE APPROPRIATE NAME
14400  IF S=5 THEN 14470
14410  IF S=6 THEN 14450
14420  PRINT "[HOM]";R1$;"[4 CD][3 CR]THE BLACK TOWER"
14430  PRINT R1$;"[3 CR][3 SPC]OF ZAEXON"
14440  PRINT R1$;"[3 CD][3 CR][3 SPC]FLOOR ";FL-1:
       GOTO 14490
14450  PRINT "[HOM]";R1$;"[2 CD][5 CR][REV][SPC]
       VOUNIM'S[SPC][OFF]"
14460  PRINT R1$;"[5 CR][REV][3 SPC]LAIR[3 SPC][OFF]":
       GOTO 14500
14470  PRINT "[HOM]";R1$;"[2 CD][4 CR][REV]THE TEMPLE OF
       [OFF]"
14480  PRINT R1$;"[4 CR][REV][2 SPC]Y'NAGIOTH[2 SPC]
       [OFF]"
14490  P(FL+1)=P(FL)+P
14499  REM ** SCATTER SPECIAL FINDS
14500  IF FL<4 OR RND(TI)<0.3 THEN RETURN
14510  FOR I=1 TO INT(RND(TI)*5)+2
14520  N1=INT(RND(TI)*19)
14530  N2=INT(RND(TI)*12)
14540  IF PEEK(32811+40*N2+N1)<>32 THEN 14520
14550  POKE (32811+40*N2+N1),42
14560  NEXT I
14570  RETURN
```

Stairs

In the Black Tower each floor is connected to the next by a set of stairs. These are set at diagonally opposite corners of each floor and each stair operates only in one direction. This means that if you walk up one flight you have to cross the entire floor to reach the next set; you can't simply go down the flight you came up!

The routine is located from 15000 to 15110 and starts by offering you a choice of going either up or down. The character pressed is checked at 15030 to 15050 to see if it is valid and if it is, the FL variable is incremented. This value represents the floor to which you wish to move and checked by line 15060 to ensure that it is within limits. If the value of FL is outside the limits, a suitable message is printed and the current floor level reset into FL from the temporary variable TV. **HB**

```
14999  REM ** STAIRS ROUTINE
15000  POKE W,81:POKE M,32
15010  PRINT D$;"A STAIRWAY...
       UP OR DOWN ?":TV=FL
15020  VG$="UD":GOSUB 1500:
       REM ** UNIGET
15030  IF GC$="U" THEN FL=FL+1:
       GOTO 15050
15040  FL=FL-1
15050  IF FL>7 OR FL<2 THEN 15080
15060  DF=110:DL$="D":GOSUB 36000
15070  GOTO 9220
15080  PRINT D$;"THESE STAIRS
       ARE BLOCKED[SPC]"
15090  DF=60:DL$="D":GOSUB 36000:
15100  FL=TV:GOTO 15010
```

Delays

The routine from 36000 can be broken down into three functional blocks; delay, wipe and update. All calls to the routine are first set up by defining the contents of the variable DF which controls the length of the delay. If only the delay section of the routine is required then a flag variable, DL$, is set to "D" to indicate this; the test in line 36020 causes an early RETURN.

In cases where a message wipe is needed after the delay but no update is required, the flag is set to "W" which forces a RETURN at line 36060. The wipe is simply performed by overwriting the text area with spaces.

The rest of the routine is concerned with updating the adventurer's status on the screen. Before the data is printed it is checked to see if it has reached or exceeded the maximum for the current character type, see Table 2. The code that performs these checks can be found in lines 36070 to 36100. The variables for experience, treasure and turns, can only increase so these are simply overprinted in line 36120 to 36140. The value of combat strength, psi power and stamina can decrease as well as increase so these are first erased and then reprinted; lines 36150 to 36170 perform this task.

If a combat is in progress the flag variable, CF, is set to 1 and this is tested for in 36180. If it is set, the monster's current status is also updated at line 36210 and 36220. If, however, the flag is cleared to show that no combat is taking place, the line of the screen where this information would normally occur is wiped clean. **HB**

```
35999  REM ** DELAY, WIPE & UPDATE ROUTINE
36000  FOR DL=1 TO (DF*TM)
36010  NEXT DL
36020  IF DL$="D" THEN DL$="":RETURN
36030  PRINT D$;SP$
36040  PRINT SP$
36050  PRINT SP$
36060  IF DL$="W" THEN DL$="":RETURN
36070  IF CS>77-INT(2*P1^2.5) THEN CS=77-INT(2*
       P1^2.5)
36080  IF PS<7 THEN PS=7
36090  IF PS>INT(42*(P1+1)^LOG(P1^3.7))+75 THEN
       PS=INT(42*(P1+1)^LOG(P1^3.7))+75
36100  IF C>125-(INT(P1)*12.5) THEN C=125-INT(INT(P1)*
       12.5)
36110  PRINT D1$;"[CU]";J$,P$
36120  PRINT "TREASURE   =";TS
36130  PRINT "EXPERIENCE =";EX
36140  PRINT "TURNS      =";TN
36150  PRINT D1$;R1$;"COMBAT STR =[4 SPC][4 CL]";CS
36160  PRINT R1$;"PSI POWER  =[4 SPC][4 CL]";PS
36170  PRINT R1$;"STAMINA    =[4 SPC][4 CL]";C
36179  REM ** IF FIGHTING UPDATE MONSTER
36180  IF CF=1 THEN 36210
36190  PRINT SP$
36200  RETURN
36210  PRINT D$;[2 CU][REV]";M$;"[OFF]";
36220  PRINT D$;R1$;"[2 CU]M STR =[12 SPC][12 CL]";MS;N;
       "[4 SPC]"
36230  RETURN
```

Ratings

The ratings system used in the Valley program is based on a character achieving the maximum rating of 28, Master of Destiny, only after amassing 200,000 experience points.

Assigning a rating of 7 to an experience of 10,000 and a rating of 20 to 50,000 experience, the plotted curve began to show definite parabolic tendencies. After experimenting with the general equation of a parabola, $y^2 = 4ax$ or $y = c\sqrt{x}$ (where c is a constant), no simple values were found to fit. So...we compromised! Using the formula, $y = 0.067\sqrt{x}$, we managed to get y values of 6.7 for an x value of 10,000, 15 at 50,000 and 28 at 200,000.

Realising that the rating should be based on experience and treasure, the x factor was defined as $x = EX + TS/3$. Then, in an attempt to penalise cowardice and rewarding those taking risks, a second factor, log (EX/(TN + 1) ↑1.5)), was added taking the number of turns to acquire your experience into the final equation. **HB**

	RATING CLASSIFICATION		
1	Monster Food	14	Champion
2	Peasant	15	Necromancer
3	Cadet	16	Loremaster
4	Cannon Fodder	17	Paladin
5	Path Walker	18	Superhero
6	Novice Adventurer	19	Dragon Slayer
7	Survivor	20	Knight of the Valley
8	Adventurer	21	Master of Combat
9	Assassin	22	Dominator
10	Apprentice Hero	23	Prince of the Valley
11	Giant Killer	24	Guardian
12	Hero	25	War Lord
13	Master of the Sword	26	Demon Killer
		27	Lord of the Valley
		28	Master of Destiny

Left: The rating numbers and their corresponding classifications.

```
44999  REM ** RATING ROUTINE
45000  DF=5:DL$="W":GOSUB 36000:REM ** DELAY + WIPE
45010  RT=INT(0.067*(EX+TS/3)^0.5+LOG(EX/((TN+1)^1.5))):
       IF RT>28 THEN RT=28
45020  IF RT<0 THEN RT=0
45030  PRINT D$;"YOUR RATING NOW BE";RT
45040  IF T(2)=1 THEN PRINT "YOU HAVE THE HELM OF EVANNA"
45050  IF T(0)=1 THEN PRINT "AMULET STONES...[SPC]";T(1)
45060  DF=250:DL$="W":GOSUB 36000:REM ** DELAY + WIPE
45070  IF GC$="E" THEN C=C-10:GC$="":GOTO 2010:
       REM ** MOVEMENT
45080  RETURN
```

THE VALLEY

Quit

If the adventurer steps on either of the two safe castles, one at each end of the path, he is offered the option of leaving the Valley. Regardless of his selection, his current rating is also computed and displayed at this point. If the player chooses to leave the Valley by keying "Y", control is passed to the Save routine at line 50000.

Because the castle is safe the player's character is 'healed' of his wounds and readied for the Valley once more. This healing consists of resetting the stamina to its maximum value and ensuring a minimum combat strength of 20. **HB**

```
47999  REM ** QUIT VALLEY ROUTINE
48000  PRINT D$;"THOU ART SAFE IN A CASTLE":IF CS<20 THEN
       CS=20
48010  POKE M,PK:PK=PEEK(W):M=W:POKE M,Q
48020  PRINT "WILT THOU LEAVE THE VALLEY (Y/N) ?"
48030  VG$="YN":GOSUB 1500:REM ** UNIGET
48040  DF=5:DL$="W":GOSUB 36000:REM ** DELAY + WIPE
48049  REM ** GENERATE RATING IN CASE OF SAVE
48050  GOSUB 45000:REM ** RATING
48060  DF=110:DL$="W":GOSUB 36000:REM ** DELAY + WIPE
48070  IF GC$="Y" THEN 50000:REM ** SAVE ROUTINE
48080  C=150:PRINT D$;"THY WOUNDS HEALED...THY SWORD
       SHARP"
48090  PRINT "GO AS THE GODS DEMAND..TRUST NONE OTHER"
48100  DF=120:GOSUB 36000:REM ** DELAY + UPDATE
48110  GOTO 2010:REM ** MOVEMENT


49999  REM ** SAVE CHARACTER ROUTINE
50000  PRINT "[CLS]DO YOU WISH TO SAVE ";J$;" ?"
50010  PRINT:PRINT "PLEASE KEY Y OR N"
50020  VG$="YN":GOSUB 1500:REM ** UNIGET
50030  IF GC$="N" THEN 50210
50040  PRINT "[CLS]PLACE YOUR CASSETTE IN THE TAPE DECK"
50050  PRINT "IS IT REWOUND ?"
50060  GOSUB 1600:REM ** ANYKEY
50069  REM ** THIS IS FOR PET ONLY
50070  OPEN 1,1,1,J$
50080  PRINT#1,P$
50090  PRINT#1,TS
50100  PRINT#1,EX
50110  PRINT#1,TN
50120  PRINT#1,CS
50130  PRINT#1,PS
50140  PRINT#1,T(0)
50150  PRINT#1,T(1)
50160  PRINT#1,T(2)
50170  PRINT#1,C1
50180  PRINT#1,P1
50190  CLOSE 1
50200  PRINT "[CLS][3 CD]"," *** DONE ***"
50210  PRINT D$;"[6 SPC]TYPE RUN TO START AGAIN"
50220  CLR
50230  END


54999  REM ** DEATH ROUTINE
55000  C=0:CS=0:PS=0:CF=0
55010  DF=110:GOSUB 36000:REM ** DELAY + UPDATE
55020  IF T(1)=6 THEN 55070
55030  PRINT D$,"[CR]OH WHAT A FRAIL SHELL"
55040  PRINT,"[2 CR]IS THIS THAT WE CALL MAN"
55050  DF=300:DL$="W":GOSUB 36000:REM ** DELAY + WIPE
55060  PRINT "[CLS]":GOTO 50210
55069  REM ** RESTORE CHARACTER TO LIFE
55070  T(0)=0:T(1)=0:TS=0:CS=30:C=150:PS=30
55080  PRINT D$;"ALARIAN'S AMULET PROTECTS THY SOUL"
55090  PRINT "[CD][REV][2 SPC]LIVE AGAIN[2 SPC][OFF]"
55100  DF=150:GOSUB 36000:REM ** DELAY + UPDATE
55110  L=G(0):MP=L:M=W:S=1:GOTO 9220:REM ** SCENE CONTROL


59999  REM ** DATA FOR CASTLE TYPE SCENARIOS
60000  DATA 4,7,3,6,4,4,6,5,3,6,0,3,8,4,3,5,5,3,
       8,3,4,5,0,6,3,6,4,6,4,7,4,100
60009  REM ** DATA FOR MONSTERS
60010  DATA AWOLFEN,9,0,AHOB-GOBLIN,9,0,AORC,9,0,
       EFIRE-IMP,7,3,GROCK-TROLL,19,0
60020  DATA EHARPY,10,12,AOGRE,23,0,BBARROW-WIGHT,0,25,
       HCENTAUR,18,14
60030  DATA EFIRE-GIANT,26,20,VTHUNDER-LIZARD,50,0,
       CMINOTAUR,35,25,CWRAITH,0,30
60040  DATA FWYVERN,36,12,BDRAGON,50,20,
       CRING-WRAITH,0,45,ABALROG,50,50
60049  REM ** SPECIAL MONSTERS FOR WATER ONLY
60050  DATA LWATER-IMP,15,15,LKRAKEN,50,0
```

Save

Stepping on one of the two safe castles is the only way to leave the Valley in an upright position as the option to save your character on tape is then offered. Taking this option out of the Quit routine passes control to the Save routine at 50000. The lines of code between 50070 and 50190 are specific to the PET and should be replaced with the corresponding code for whatever system you are implementing the game on.

At the end of this routine, whether you reach it by saving the data on tape or by choosing not to save in the Quit routine and dropping through, all the current variables are cleared and a farewell message displayed. **HB**

Death

This routine is the one part of the program the player would rather not have executed! Many and varied are the ways in which one can arrive at line 55000 and on all but one occasion the outcome is inevitable. The one exception is when you have been fortunate enough to collect the Amulet of Alarian and filled it with the six missing stones because this gives you a second life.

The test to see if you have the Amulet and its stones is made at 55020 and if successful you are restored to life. The price is, however, high as you lose all your treasure together with the Amulet and its stones. Your combat strength and your psi power are both set to 30; the only value that remains the same after death is your experience. The Valley is now redrawn and the character starts from the initial position once again.

If, as is most likely the case, you don't have the protection of the Amulet and its six stones then the game ends with all the variables being zeroed in line 50220. **HB**

Data

Rather than placing each data block with its relevant routine we have chosen to lump it all together at the end of the program. The first block contains all the information needed to build the three castle-type scenarios (see the relevant sections for more details on this). The second block of data holds the monster information which is READ into the three arrays M$(), MS() and N1() at the start of the progam. **HB**

CODE	SYM-BOL	CODE	SYM-BOL	CODE	SYM-BOL	CODE	SYM-BOL	CODE	SYM-BOL	CODE	SYM-BOL	CODE	SYM-BOL	CODE	SYM-BOL
0	@	32	SP	64	▬	96	SP	128	@	160	SP	192	▬	224	SP
1	A	33	!	65	♠	97	▌	129	A	161	!	193	♠	225	▌
2	B	34	"	66	▐	98	▄	130	B	162	"	194	▐	226	▄
3	C	35	#	67	▔	99	▔	131	C	163	#	195	▔	227	▔
4	D	36	$	68	▁	100	▁	132	D	164	$	196	▁	228	▁
5	E	37	%	69	▏	101	▏	133	E	165	%	197	▏	229	▏
6	F	38	&	70	▒	102	▒	134	F	166	&	198	▒	230	▒
7	G	39	'	71	▕	103	▕	135	G	167	'	199	▕	231	▕
8	H	40	(	72	▕	104	▕	136	H	168	(	200	▕	232	▕
9	I	41	)	73	◿	105	◿	137	I	169	)	201	◿	233	◿
10	J	42	*	74	◿	106	▗	138	J	170	*	202	◿	234	▗
11	K	43	+	75	◿	107	▗	139	K	171	+	203	◿	235	▗
12	L	44	,	76	◻	108	▪	140	L	172	,	204	◻	236	▪
13	M	45	-	77	◿	109	▗	141	M	173	-	205	◿	237	▗
14	N	46	.	78	◿	110	▗	142	N	174	.	206	◿	238	▗
15	O	47	/	79	◻	111	▟	143	O	175	/	207	◻	239	▟
16	P	48	0	80	◻	112	▚	144	P	176	0	208	◻	240	▚
17	Q	49	1	81	◘	113	▚	145	Q	177	1	209	◘	241	▚
18	R	50	2	82	▌	114	▚	146	R	178	2	210	▌	242	▚
19	S	51	3	83	♥	115	▚	147	S	179	3	211	♥	243	▚
20	T	52	4	84	▯	116	▟	148	T	180	4	212	▯	244	▟
21	U	53	5	85	◿	117	▟	149	U	181	5	213	◿	245	▟
22	V	54	6	86	╳	118	▟	150	V	182	6	214	╳	246	▟
23	W	55	7	87	◙	119	▟	151	W	183	7	215	◙	247	▟
24	X	56	8	88	♣	120	▟	152	X	184	8	216	♣	248	▟
25	Y	57	9	89	▯	121	▟	153	Y	185	9	217	▯	249	▟
26	Z	58	:	90	◆	122	▟	154	Z	186	:	218	◆	250	▟
27	[	59	;	91	⊞	123	▟	155	[	187	;	219	⊞	251	▟
28	\	60	<	92	▦	124	▟	156	\	188	<	220	▦	252	▟
29	]	61	=	93	◫	125	▟	157	]	189	=	221	◫	253	▟
30	↑	62	>	94	π	126	▟	158	↑	190	>	222	π	254	▟
31	←	63	?	95	◢	127	▟	159	←	191	?	223	◢	255	▟

Screen memory:- 32768-33767
8000H-83E7H

Format:- 25 lines of 40 characters

Notes:- Graphics characters may be converted to lower case alphabetics with POKE 59468,14 and back with POKE 59468,12. CHR$ (147) clears the screen. Note that when outputting screen based information the PET uses an absolute TAB rather than spaces which can disrupt apparently neat formats. For full and well explained details on the PET see the 'PET Revealed' from Computabits, price £10.

Commodore PET