Adventure

A swift glance down most software catalogues will reveal a veritable hoard of Adventure games.

They compete with Space Invader and Asteroid type games for the top-of-the-micro-pops. Which you prefer depends on whether you like to test your reactions and control, or are the mystery-loving puzzle-solver type (these are by no means mutually exclusive).

If you are new to Adventure and wondering which to try first, or if you are just wondering which to buy next, I will be helping to guide you through the Adventure jungle each issue. A brief rundown of what to expect from each Adventure, will help you select tapes and discs to suit your taste a little less randomly.

SCOTT Adams is the name behind a whole library of Adventures and his games have introduced many enthusiasts to micro adventuring. My own introduction was his Adventureland on a friend's machine. It is set in a swampy forest near a sunny meadow and progresses underground to a maze of pits.

Various things lying about the place seem rather innocuous, but as the game develops a feeling grows that most objects have a purpose, and that some very devious thinking will be needed to find uses for, among other things, an empty wine bladder and patches of oily slime! The large dragon, peacefully sleeping in the meadow, begs a good hard kick to wake him up — is he really as impervious to attack as he seems? And the "No Swimming" sign by the lake is positively urging you to take a dip — but with what consequences? The object of the game is to collect and store 13 treasures. To say more would be to give away too much.

So I'll merely say "Bunyon" and vibrate on to another Scott Adams' game — Pirate Adventure. A strong theme runs through this game which is littered with bottles of rum, treasure chests, anchors and a parrot, which is not only excessively greedy but very loquacious.

The story begins in the player's London flat, from where, after some chilling discoveries, it moves to Pirate Island. There it soon becomes apparent that you are being urged to do something without being told quite what.

Pirate is not as deep as Adventureland and the machine's memory is not so fully packed. But this is more than offset by a delightful sense of humour running through the game, climaxed by a cruel and dramatic hoax.

Both these games are available on the 16K TRS-80 and Video Genie; the 16K Exidy Sorcerer; 24K Apple 2 and Apple 2 Plus.

The Adventure series by Scott Adams, currently comprises 10 games. All games are written in machine code and have a "save game" feature enabling the current state of the game to be recorded and reloaded later.

Each month I shall be bringing you tips on how to write an Adventure program in Basic. In order to do this you will require a machine with at least 8K RAM and capable of holding many string variables, arrays (single dimension will do). It will also need string manipulation statements like: MID$, LEFT$, RIGHT$, LEN, plus the ability to concatenate.

Having devised your plot and drawn a map the next step is to number the locations from zero, and draw up a table. For simplicity I have shown a five location map in Figure 1 and the corresponding table in Figure 2. All exits in the example are compass bearings, hence the exit column entry for location 0 shows "ES"—E(ast S(outh) leading to destinations in the corresponding positions of the destination column entry, of locations 1 and 2 respectively. If more than 10 locations were to be used, double figures would be needed in the destination column.

Type the contents of the table, omitting the number column, row by row into data statements. Part of your program will now look like this:

```
DIM L$(4), E$(4), D$(4)
For I = 0 TO 4: READ
L$(I), E$(I), D$(I): NEXT
DATA COTTAGE, ES, 12, LANE,
  WS, 03, FOREST . . .
```

and your logical network is formed!

Putting aside sophisticated word decoding routines for the moment, we can test this network out with the following simple routine:

```
150 (clear screen): LN = 1: REM
  CURRENT LOCATION IS 1
160 PRINT "I AM IN A "; L$(LN)
170 INPUT "WHERE TO NOW";
  R$ (clear screen): REM ANS-
  WER N, S, E or W
180 R$ = LEFT$(R$, 1)
190 FOR I = 1 TO LEN(E$(LN) )
200 IF MID$ (E$ = VAL (MID$
  (D$(LN), I, 1) ): GOTO 160
210 NEXT
220 PRINT "I CAN'T GO THERE" :
  GOTO 160
```

To place objects in these locations is now quite easy. Make an object table as in Figure 3 using array P to hold the current location of each object. Read this in from data statements as with the locations. Now add these lines to those above, and Hey Presto!

```
165 OS$ = "I CAN SEE ": FOR I = 0
  to 3
166 IF P(I) = LN THEN OS$ =
  OS$ + O$(I)
167 NEXT: PRINT OS$
```

We can't manipulate the objects yet — that will come after we've had a look at word decoding next month.

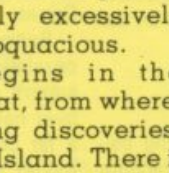

Fig. 1: Simplified network of locations showing initial positions of objects in brackets. Note: objects and locations independently numbered.

No.	Location (Array L$ (4))	Exits ES (4)	Destination DS (4)
0	Cottage	ES	12
1	Lane	WS	03
2	Forest	NE	03
3	Meadow	NWS	124
4	Lake	N	3

Fig. 2: Location table derived from map in Fig. 1. The number column is merely the subscript used to access the information on a given line. Note all variables are character (string).

No.	Object Array D$ (3)	Place P(3)
0	Knife	0
1	Fish	4
2	Cow	3
3	Axe	2

Fig. 3: Object table derived from Fig. 1. Again the number column is the array subscript. Note that since array P will only hold the number of the current location of an object it may be defined as integer numeric.

BY KEITH CAMPBELL