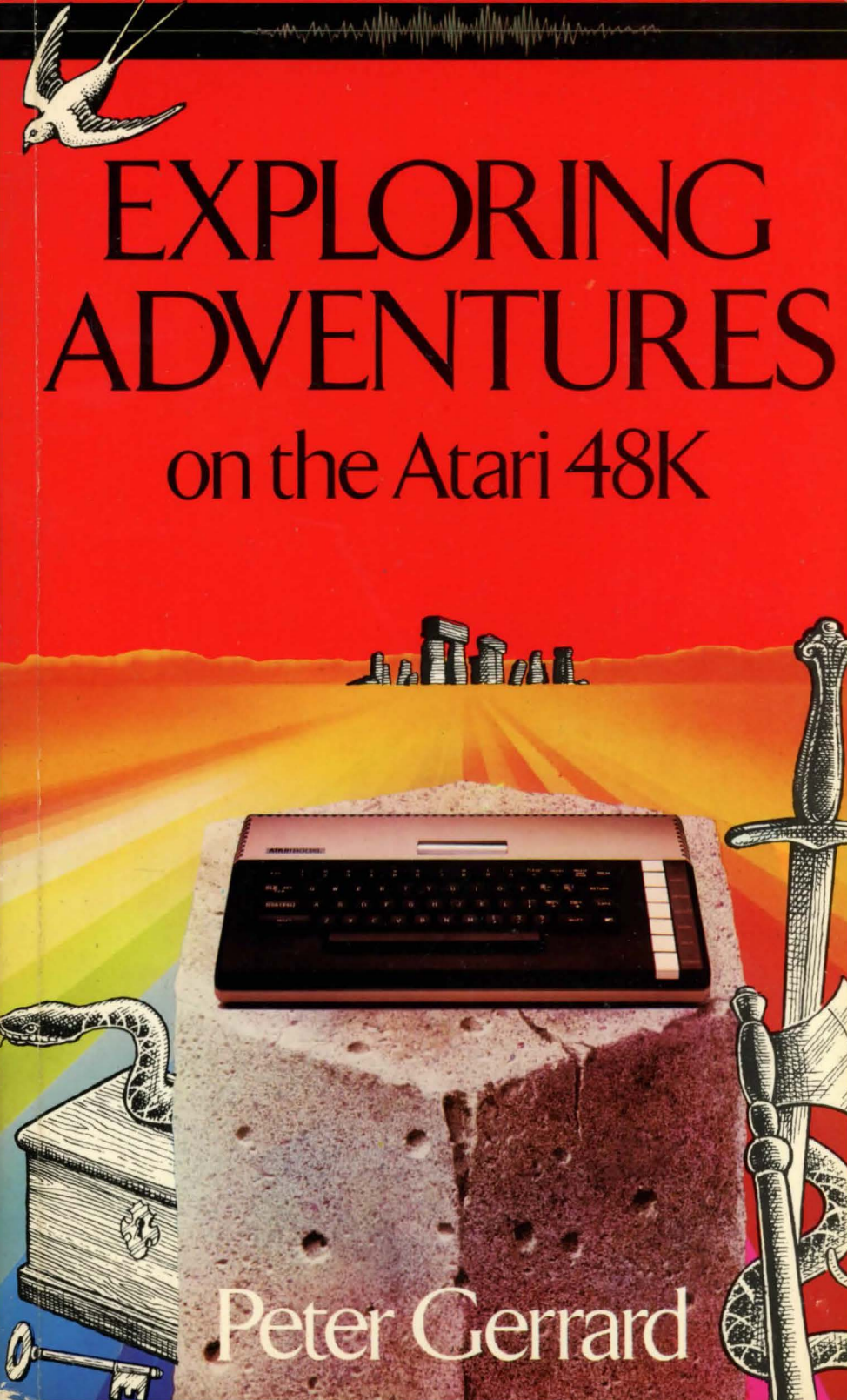
EXPLORING ADVENTURES

on the Atari 48K



Peter Gerrard

Exploring Adventures on the Atari 48K

EXPLORING ADVENTURES

on the Atari 48K

Peter Gerrard

Duckworth

First published in 1984 by
Gerald Duckworth & Co. Ltd.
The Old Piano Factory
43 Gloucester Crescent, London NW1

©1984 by Peter Gerrard

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

ISBN 0 7156 1924 1

British Library Cataloguing in Publication Data
Gerrard, Peter
Exploring adventures on the Atari. —
(Duckworth home computing)
1. Computer games 2. Atari computer
I. Title
794.8'028'5404 GV1469.2

ISBN 0-7156-1924-1

Typeset by The Electronic Village, Richmond
from text stored on a Commodore 64
Printed in Great Britain by
Redwood Burn Ltd., Trowbridge

Contents

Preface vii

1. An Introduction to Adventure Games 1
2. How to Solve Adventures 13
3. Programming Adventures in Basic 47
4. Writing Your Own Adventures 79
5. Creating Your Own Adventures 115
6. Underground Adventure 135
7. Castlemaze Adventure 215
8. Tunnel Adventure 227
9. Further Information 239

Index 245

Introduction

This book is for anyone interested in the world of adventure games on computer.

Whether you like to play them, write them, or write about them, this book has been written with you in mind.

More specifically, it is aimed at the person who loves to get absorbed in a game for hours on end, has always wanted to write one of his own, but has taken one look at a listing of someone else's game and thought 'There is no way that I could write something like that!'

This book shows you how to write a fully fledged adventure game, with unique sections on room mapping, data structure, input routines, verb handling, and everything you'll need to know to write an adventure of your own.

The main game in the book, Underground Adventure, is gone through line by line, with each piece of code explained so that you know precisely what is going on.

By the end of this book, you will be in a position to produce your own game for your computer.

Thanks to Steve Darnold, for getting me started in all this (although you didn't know it at the time!).

Thanks also to Jim Butterfield, who gave me my first game of Adventure. And what a game to start with!

Finally, a couple of dedications. Thanks to my wife for doing all the illustrations. Living with her has certainly been an adventure!

And last of all, to the lad with whom I played the longest ever game of Adventure I've played in my life, which probably did more than anything to get me hooked on these games. This single game lasted for about twelve hours, after which time we were still bribing trolls, feeding bears and exploring the bedrock room as we walked to the pub for a pint. Denis Timm, have you managed to get out of the Pirate's maze yet?!

1
An Introduction to Adventure Games

General Introduction

Adventure Games have been played on computers for many years, and are one of the most popular of all types of computer games, if not *the* most popular.

It is sometimes difficult to describe exactly what an adventure game consists of. You're in a magical world of the writer's imagination, doing battle with unknown and often unseen problems, that sometimes appear to defy all logical solutions. You can be placed underground, underwater, in outer space, in colossal caves, or just about anywhere within the known universe, but the ultimate objectives of all the games are usually the same: to survive, and collect all the treasure that is rumoured to exist in these weird and wonderful games.

My own connections with adventure game playing and writing started with the very first game of all - Adventure - playing an abridged version on the Commodore PET 3032 computer, with a 3040 disk drive. One night after a party two of us sat down in front of the computer and, armed with a bottle of whisky in the real world and nothing more than a torch, a bottle of water, a key and some food in the adventure world, began playing a game that was to go on for more than twelve hours!

We simply did not notice that it was now light outside. We were deep underground, trying to cross a bridge with a bear that was too heavy for the bridge, and we didn't care about such commonplace things as sleep!

That early start has led to a lifetime of interest in that game and adventure games as a whole, and my interest in the games is shared by countless other people around the world, who have made this one of the most popular of computer games.

It is hard to explain this popularity to a non-addict. Peculiar looks and pitying stares are the usual response when it is revealed that you spend hours at the keyboard, glued to happenings in an imaginary world.

On the other hand, joining one of the many Adventure user groups will place you amongst many like-minded people who fully understand the frustration at trying to solve a particular problem. 'What *do* you do with the platinum pyramid?!', no longer evokes a 'What on earth are you on about now' attitude, instead you're more likely to get a hundred and one hints and tips on solving the problem of the platinum pyramid.

Adventure enthusiasts even have their own Agony Aunt now in Tony Bridges, who writes a regular weekly column for the microcomputer magazine *Popular Computing Weekly*. Every week he'll take a look at an aspect of adventure playing, or a particular problem in one of the more popular games, and you're welcome to contact him over any problems that you might be experiencing in your own adventure game.

The number of players of these games is legion, and this book has been written to help you write your own adventure programs, and to explain a little bit about the origins of the games, with more than a passing glance at some of the games (and the people) who helped to make this genre of game playing the success it's become today.

We're also giving you three complete adventure game listings at the back of the book, with a full explanation of the Underground Adventure game and how it was written, and brief explanations for the other two.

If the thought of typing in pages and pages of code is a daunting one, you'll be pleased to know that the publishers are also offering these programs on cassette, and that cassette will cost you £7.95, available direct from the publishers.

The listings and sections on programming are all aimed at your computer, using a cassette deck as the storage medium.

It is hoped that, by the end of this book, you'll be more than capable of writing your own games, and perhaps joining the author's Fool's

Gold and Tombs of Xeiops as top-selling adventure programs!

So without further ado, let's take a look at the history of adventure games, and we'll start with the very first one of all, called, simply, Adventure: the game from which all others have taken their generic name.

How It All Began

Although most adventure programs these days seem to be written in Basic, which is the style of writing that we'll be showing you in this book, or machine code, the very first one was written in Fortran, not a language known for its string handling capabilities. Which language you choose is very much up to you, bearing in mind the restrictions of the computer in front of you.

Basic is usually chosen because it's easier than anything else, most Basics have a good set of commands for manipulating strings, and there is no great requirement for speed in this type of game. The essence of these games should always be that you have to think, not act in the frantic fashion of a good arcade game, and because of that we don't have to program everything to happen at lightning speed.

Some adventure games are written in machine code - Zork is a classic example - but the writing of a game like that is beyond the scope of this introductory tome. It is a vast program, usually supplied on three different disks, such is its size.

In Zork, speed is required because of the many and varied ways it can accept the inputting of information from you, the player. Most adventures are restricted to the TAKE STAFF style of commands: one verb and one object, but Zork goes beyond that to the level where you can say something like BURN ALL THE BOOKS EXCEPT THE BLACK ONE, and other complicated instructions.

The first Adventure game used the simple GO NORTH style of instructions: for that game, and for just about everything that's appeared since, credit has to go to Willie Crowther and Don Woods, who wrote the program on a DEC (Digital Equipment Corporation) PDP-10, in , as we've seen, Fortran.

That program required about 300K of computer memory to play it: a great deal more than you get on your computer!

Abridged versions have appeared since then for most of the popular home computers, and it was the work of Jim Butterfield that led to the version now available for all the Commodore range of computers.

Since then, a version has appeared for the IBM Personal Computer, but for some reason it is being marketed commercially. Odd, since it is available free of charge from most user groups!

If you want a copy of that game for your computer, I would suggest getting in touch with one of your local user groups: several names and addresses are given at the back of this book.

If you have never played this, the first ever Adventure program, I would strongly suggest that you do so. Not only is it one of the best adventure games ever written, it is also the origin of every other adventure game. Without it, people like Scott Adams and Greg Hassell would probably have never written their own series of (very good) adventure games.

We'll look at some of theirs later, but for now let's stick to the original.

It is sometimes called the Colossal Cave Adventure, for the opening scenario goes like this: =

'Somewhere nearby is colossal cave, where others have found fortunes in treasure, though it is rumoured that some who enter are never seen again. Magic is said to work in the cave. I will be your eyes and hands. Direct me with commands of 1 or 2 words. I should warn you that I only look at the first five letters of each word, so you'll have to enter "northeast" as "ne" to distinguish it from north. This program was developed by Willie Crowther and Don Woods. This version is abridged for PET disk by Jim Butterfield.'

We'll go into more detail on Adventure (with a capital A to distinguish it from the games as a whole) in chapter 2.

All of this was developed on a mainframe computer with 300K of memory. So how did they get to appear on the microcomputers that we know today ?

The Transition to Microcomputers

The first person to think about putting an adventure onto a small microcomputer was Scott Adams, an American who is commonly ackowledged to be the father of adventure games on small computers.

His story makes interesting reading, and you can find it in the December 1980 edition of the American magazine BYTE, in which there was a special feature on adventure games, and Scott Adams related the story of how it all began.

For the benefit of those who haven't got access to the magazine, here's a brief synopsis:

Scott Adams' first game was written on a Radio Shack TRS-80 level II computer, and came about after he'd already written a few other, non-adventure, games for it.

At the time he was working as a systems programmer for Stromberg Carlson, and he'd been introduced to the original Adventure by a friend. After apparently playing the game for ten days he managed to solve the whole thing, having been totally addicted from that opening scenario given earlier.

However, he realised that not everyone could afford a DEC PDP-10! So, the quest was on to produce a reasonable adventure on a much smaller computer: in his case the TRS-80.

The idea came to him of producing an adventure interpreter. This would allow him to write many different adventures, but at the same time cram an awful lot of information into a very small area of memory.

The programs at the back of this book work along similar lines, in that routines exist within them to move from room to room, store the room descriptions, handle the input of data, and so on, and these routines are common to every listing given. This makes it possible to create adventures with a minimum amount of work from the writer, but at the same time they can be different enough to keep people occupied trying to solve them for many, many hours.

Possibly the most difficult part of writing an adventure, once the actual program structure has been grasped and understood, is getting the original idea in the first place, and working it through as a strong idea that doesn't rely on the impossible happening before the adventure can be solved.

The idea for Scott Adams' first adventure, generally reckoned to be his best, was not particularly brilliant, in that one was doing the usual treasure seeking and problem solving. Nevertheless, it did fit into 16K as opposed to 300K!

After six months of testing his adventure, and of course the interpreter that was driving it all, this first program (called Adventureland) was released through The Software Exchange of Milford, New Hampshire, and Creative Computing Software.

Thus, as he says in his own article, the Scott Adams series of adventures was born.

Apparently it almost died there and then, since his wife was taking great exception to him spending six months locked in a room writing programs! However, all was solved when she decided to write an adventure, and came up with the idea for Pirate Adventure, the second best adventure program he's ever marketed.

In this one the idea is different, in that you have to do slightly more than merely collect treasures and solve problems. You have to build a pirate's ship, and not many people start off with the knowledge to do that!.

And so the transition to microcomputers was complete. It was possible to write an adventure with only a minimal amount of memory, and the market suddenly begain to explode.

The Market Blossoms

Scott Adams has written a large number of adventures now, well into double figures, and we'll be taking a look at some of the better ones later.

But while Scott was doing all his work, there was another, younger, adventure devotee called Greg Hassett, who is now I believe 16 years old, but already the author of at least 8 adventure programs.

Some of his programs, natural enough because of his age, are not worthy competitors to the earlier Adams work, but nevertheless there are some gems to be found from this young schoolboy.

In particular, Enchanted Island Plus is well worth seeking out. Written entirely in machine code (as opposed to his earlier Basic Enchanted Island), solving this one will keep you occupied for a long time to come.

Some of his plots are also refreshingly different. Journeys to the centre of the earth and visiting Atlantis may be fairly run of the mill, but situations where you have to save an almost totally polluted earth from

extinction are much better. World's Edge is possibly the best one that Hassett has written.

Companies producing adventures in these early days tended to be mainly American, and it took a long while before the rest of the globe started producing comparable games, although Britain is now catching up fast.

In those days, Radio Shack themselves started bringing out a couple of adventures, The Programmers Guild took a few pages out of Tolkein's *Lord of the Rings* and had you fighting orcs and spiders, while Mad Hatter Adventures, who started off just handling the Hassett programs, also produced a couple of their own, although these were generally considered to be rather poor when compared to the wonderful program that started off the whole craze.

Since then, of course, many companies have started marketing adventure programs, and now many exist for just about every make of home microcomputer.

Why They Are So Successful

It is true that adventure games generally have captured the computer market to a vast extent. They are one of the most popular types of all computer games, and are now enjoying something of a renaissance, with many new games currently becoming available for all types of computers.

Whilst relatively few will have the long-standing success of the original game, most will probably be worthy of playing, and many will no doubt tie their buyers to the computer for many weeks to come. Tony Bridges is going to be very busy in the months ahead.

But why is there this phenomenal success, and why do so many people spend so long typing in commands on a computer keyboard just to see what appears next upon the screen ?

It is easy to analyse the success of, say, arcade games. The sound effects, the stunning graphics, are obviously pleasing to the human ear and eye, and our society seems to be depressingly heading into a more violent era. Thus the chance to annihilate a few more aliens for a mere 20 pence is not one to be missed.

But adventure games have none of this. There is usually no graphical

display, although we'll see later that games are available that use graphics to one extent or another. Generally, there is no sound being generated by the computer either, although again there are exceptions.

Finally, there is no 'shoot-em-up and zap-em-down' approach to adventures. They are games for the thinker, rather than the person of action.

And perhaps this is part of the secret of their great success. To solve a good adventure like the original Crowther and Woods game requires a lot of logical throught, to say nothing of a lot of time. The first people to start playing the game were computer programmers themselves, and one survey in the States showed that, when an implementation of Adventure appeared on the work's computer, they would lose an estimated two weeks work due to staff playing the game in their free, and not so free, time.

Obviously people tried to put a stop to this, and started restricting access to the game, but it was generally reckoned that whatever a company tried to do, nothing would stop its employees from playing the game. Better to let them have their way for a couple of weeks, and see them emerge contented at having attained the goal of master adventurer.

The same is true for most people who start playing the game. Once you've started, it is virtually impossible to rest until you've completely solved the puzzle.

How can I get past the troll without losing treasure? How do I open the clam? How do I open the treasure chest in the pirate's maze? All these questions have to be solved before attaining the magical status of master adventurer. Sometimes you're setting yourself an impossible task, but that won't prevent people from taking hours trying to solve it, until they give up in disgust.

It becomes a question of pride: 'I am not going to be beaten by a stupid computer!' is the usual response.

Also, pride comes in when you hear of someone else talking about a room, or a particular problem, that you haven't encountered. The desire to find that room, or solve that problem, drives many people back to the keyboard again.

And, strangely enough, you will very rarely get a direct answer when you ask someone how to solve a certain problem. You'll usually get

a cryptic hint, but nothing more. So, you're back to your own logic again, and few people will admit to not being able to solve something.

Finally, adventure games usually have a sense of fun. Take the classic Adventure. The version by Jim Butterfield produces some lovely responses at times. Like this:

FEED BIRD

THE BIRD IS NOT HUNGRY, HE IS MERELY PINING FOR THE FJORDS!

Shades of John Cleese and ex-parrots. If you try typing in the inevitable rude statements, requesting the snake to do the anatomically impossible, again a variety of replies can be generated.

So a combination of problem solving, pride and fun have all contributed to making adventure games required playing for most people.

But what will happen to them in the future, as computers become more and more sophisticated?

A Glimpse into the Future

There will always be a limit to the amount of technology that can be squeezed into a home computer, just because of the sheer size of the thing.

However, there appears to be no limit to the amount of programming talent that can be squeezed into them, and it is this growth of talent that will dictate the course of adventures over the next few years.

We can already see the results of one extremely intelligent set of people in the adventure game Zork, which is in many people's minds a great step forward from the original game.

Again this was developed on a PDP-10, and has now appeared as a three part adventure for a number of home computers. Like a lot of adventures nowadays it is supplied on disk, and thus not everyone will get the chance to play it.

Still, there's always the local user club, and most user clubs have people who are perfectly capable of copying the protected disks on which Zork is supplied! Saying a disk is protected is like waving a red flag

at a bull: sooner rather than later a dedicated programmer is going to crack any form of protection you care to name. As someone once observed: 'You have to have a disk drive to make the protected disk, and you've got precisely the same disk drive as they have. Therefore, you've got the equipment to unprotect the disk.'

I'm not advocating software piracy, by the way, but when you see things like the original Adventure, a public domain program, being sold for anything up to £30, it just invites copying!

So when we get to the stage where all home computers come supplied with built-in disk drives, you can guarantee that there'll be some very sophisticated adventures coming out.

Just as the Crowther and Woods game requires a disk drive, so will many future adventures. Why a disk drive ?

Well, there is a limit to how much memory a computer has, and a disk drive will always have more. Therefore it makes sense to store the core of a program in the computer, and call up the relevant descriptions from the drive.

It will also pave the way for many more graphical adventures. If a computer has got sophisticated graphical capabilities, like many of the current home computers, it makes sense to use them.

However, to utilise all the graphical features on many computers even now can take up to 8K of memory per screen. That's a lot of memory to take up in the computer at one time, and adventures with four rooms in them tend to be solved fairly quickly.

However, hitch up a half a megabyte disk drive and you've got the capacity to handle over sixty rooms. Much more difficult to solve, and as disk drives speed up in terms of access time we can't be too far away from a true animated adventure.

Whether people want animated adventures or not is another question. They say that a picture paints a thousand words, but fifty words can paint a much more graphical image on the mind than an 8K screen display.

That is why *Lord of the Rings*, and other books of that genre, will never make a successful transition to the cinema screen or the home computer. The mind is always capable of imagining far more from a few simple words than can ever be depicted on a screen.

Perhaps that's why the more successful games are always purely textual in their display. Leave it up to the player to imagine it all, and let the computer take care of everything else.

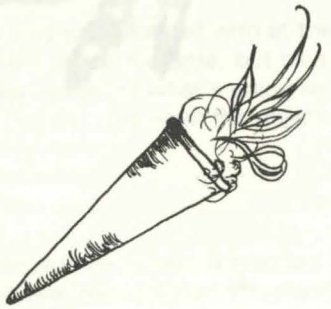
Adventure games that use half-hearted graphics, like the much praised Temple Of Apshai series, from Epyx, tend to be a great disappointment, certainly to this writer anyway.

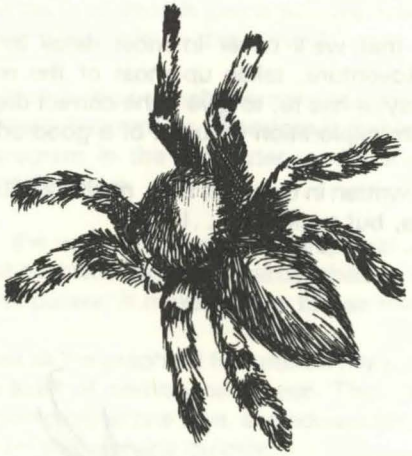
Dungeons and Dragons games in real life are all very well, but the implementation on the home computer hasn't yet arrived.

So in this book we'll stick to textual games with no graphics, on the basis that a) not everyone will have a disk drive, and b) not everyone wants graphics anyway.

The adventure that we'll cover in most detail in this book, the Underground Adventure, takes up most of the memory of your computer anyway: it has to, to give it the correct degree of problem solving and room exploration required of a good adventure.

If this book is re-written in ten years time, maybe we'll be talking about graphical games, but until then...!





2

How to Solve Adventures

Adventure Scenarios

Whatever the adventure game that you're playing, you will obviously want to solve all the puzzles presented to you, usually in the minimum amount of time, but if you're a newcomer to the game you'll probably think that the adventure is too difficult and you'll just give up, probably never to play an adventure again.

This chapter is aimed at solving adventures, and as well as some general notes we'll be taking a detailed look at the original Adventure (whilst trying not to give too much away), and also the adventure that forms a large part of this book, the Underground Adventure.

The type of scenario you're presented with at the start of the game will obviously vary from game to game, but as a general rule you'll usually be given a description of what's going on, how you happen to be there in the first place, and what the object of your mission is.

Pirate Adventures

For instance, in Scott Adams' Pirate Adventure you start off in an apartment listening to the roar of the traffic, and only after getting the non-slip sneakers and entering the secret corridor behind the bookcase will you be able to start the adventure properly by saying the magic word and being whisked off to a pirate's island, where you have to build a pirate ship and make your escape.

On the island you'll encounter a wonderfully dotty series of characters, including a drunken pirate and a mongoose, that all add to the charm of this game.

Some of the problems presented to you in various games can at first appear insurmountable. There's one game called Castle Adventure, the object of which is to explore a castle, and make your way safely back with all the treasures.

However, getting into the castle appears to be impossible at first, since it is surrounded by a moat, the drawbridge isn't down, and the moat is full of piranha fish! How do you swim through a shoal of piranhas ?

Sleepy Piranha

The answer is that you don't. You have to roam around outside the castle first of all, finding what you can, and on your travels you eventually discover a set of sleeping pills. Provided that you don't take these yourself you can drop them in the moat, whereupon the piranha obligingly swallow them and go to sleep, thus allowing you to swim across in safety. Of course, you might get your matches wet and soggy in the process, but you had thought of that hadn't you?

Another popular conundrum is the gap in the rocks that is too narrow to squeeze through with whatever you happen to be carrying at the time. The original Adventure has a feature like this, and we've taken that idea and adapted it in the Underground Adventure listing here.

The problem is usually that you can slip through the gap, but nothing that you're carrying can go through with you. As most of these adventures take place underground you require a lit torch to be with you at all times, and if the torch goes out you can't see anything, which means that you just have to blunder around until you fall into a pit and die.

If your torch can't get through the gap, how can you see anything when you're on the other side? The answer usually lies elsewhere in the game, and there will be something that will fit through with you, that begins to glow when you've got through to the other side, thus letting you see whatever happens to be there.

As a final example, Philosopher's Quest for the BBC micro has a delightful problem when it tells you that you no longer have any existence! In other words, you can no longer do anything: if you don't exist, how can you do anything? The answer is one of those horribly obvious ones when you think about it, and that in itself is the answer: if you think, you must exist, as Descartes once said.

Thus by thinking the computer acknowledges your existence, and you can carry on with the game again.

So most adventures follow a fairly standard pattern, although there have been a number of extremely silly adventures that have appeared in recent times, and two of them have both been based on popular television programmes.

There is one adventure based (loosely) on Monty Python's Flying Circus, which has you travelling around on buses, mugging old ladies, and doing all kinds of things in the worst possible taste. Rather like Python itself, really.

Hitch Hiking Around

A second game has now unfortunately been taken off the market, because it was infringing someone's copyright laws. It used to be called Hitch Hiker's Guide to the Galaxy, better known as a radio, television and book series, which found its way into an adventure game by Bob Chappell. All the favourite characters where there, and the plot for this particular game was about as sensible as the series.

However, it did have to be withdrawn, although it has since re-appeared under another name, as a thinly disguised version of its former self.

More usually though, you're exploring caves, or weird haunted castles and houses, and are presented with a reasonably logical set of problems to solve.

Often, these problems will have to be tackled in a specific order, as the solving of one inevitably leads you onto another one that will again have to be solved before you can progress further.

Underground Adventure features this, in that you have to solve some 16 problems before you can complete the entire game, and those problems have to be solved in a set order. In fact, it is usually impossible to progress further if you don't solve them in the right order.

For instance, you can't get past the giant deadly fly until you've found the giant deadly fly-spray, which is itself hidden away behind something else. And so on: solve one problem and you can progress to another.

Some adventures do present almost life-like situations, and your

behaviour has to be judged truly in the light of what you would do if you were actually in that same situation in real life.

Building Ladders

If there's a gap above your head that you can't get to, how would you reach it in real life? Most people would probably go and borrow a ladder, but as adventure games don't usually feature conveniently handy neighbours you're going to have to build one for yourself.

What do you need in order to build a ladder? Nails, wood and some kind of saw are the usual ingredients, so off you go to try and find them all.

Another popular feature is that of having some kind of animal about the place. Bears, snakes and revolting insects are the usual order of the day, and most of them will have two purposes. Bears might eat you alive at first, but tend to calm down when they're fed and perform a number of useful functions.

So the number of possible scenarios is legion, and we can expect just about anything to turn up at one point or another. However, whatever the scenario happens to be you're going to have to solve everything that's thrown at you sooner or later, so let's go about solving an adventure.

Solving Adventures

There are a number of golden rules to be observed when setting out upon a new adventure, and the principal one amongst them must be:

NEVER IGNORE ANYTHING!!

Everything you see will have been put there for a purpose, because writing adventure games on a home computer does restrict the amount of data that can be packed in, and therefore you can't really afford to put in things that will not have a purpose.

Most objects that you enounter will probably only have one role in the game, although this is by no means a hard and fast rule.

In the classic Adventure, you will repeatedly need to keep the axe with you, as little dwarves have a habit of racing out from behind rocks

and engaging you in mortal combat from time to time, and they can only be seen off by throwing the axe at them.

The torch also has to be carried with you most of the time, and in the classic Adventure you have to get a new set of batteries for it after a while, but more of that later.

Although we've said that everything has a purpose, that purpose may only be to annoy and delay you in solving the puzzle.

Life in a Dead End

This is particularly true of some roads and corridors. In Underground Adventure, for instance, there are a number of dead ends. Some of these are purely dead ends and go no further, but others are there to test you, and can be got past.

A giant boulder gets in your way at one point, but can be got round by finding some dynamite, which exists elsewhere in the puzzle, and blowing it up.

Just make sure that you're not carrying the dynamite yourself when you decide to light it, as the only thing you'll blow up then will be yourself.

Vast chasms are another popular feature, and Underground Adventure has two of them, which need to be solved in different ways. The Crowther and Woods game also employs a chasm, and if you're carrying the black rod with you when you encounter it you should be all right, provided you can work out the correct verbal syntax.

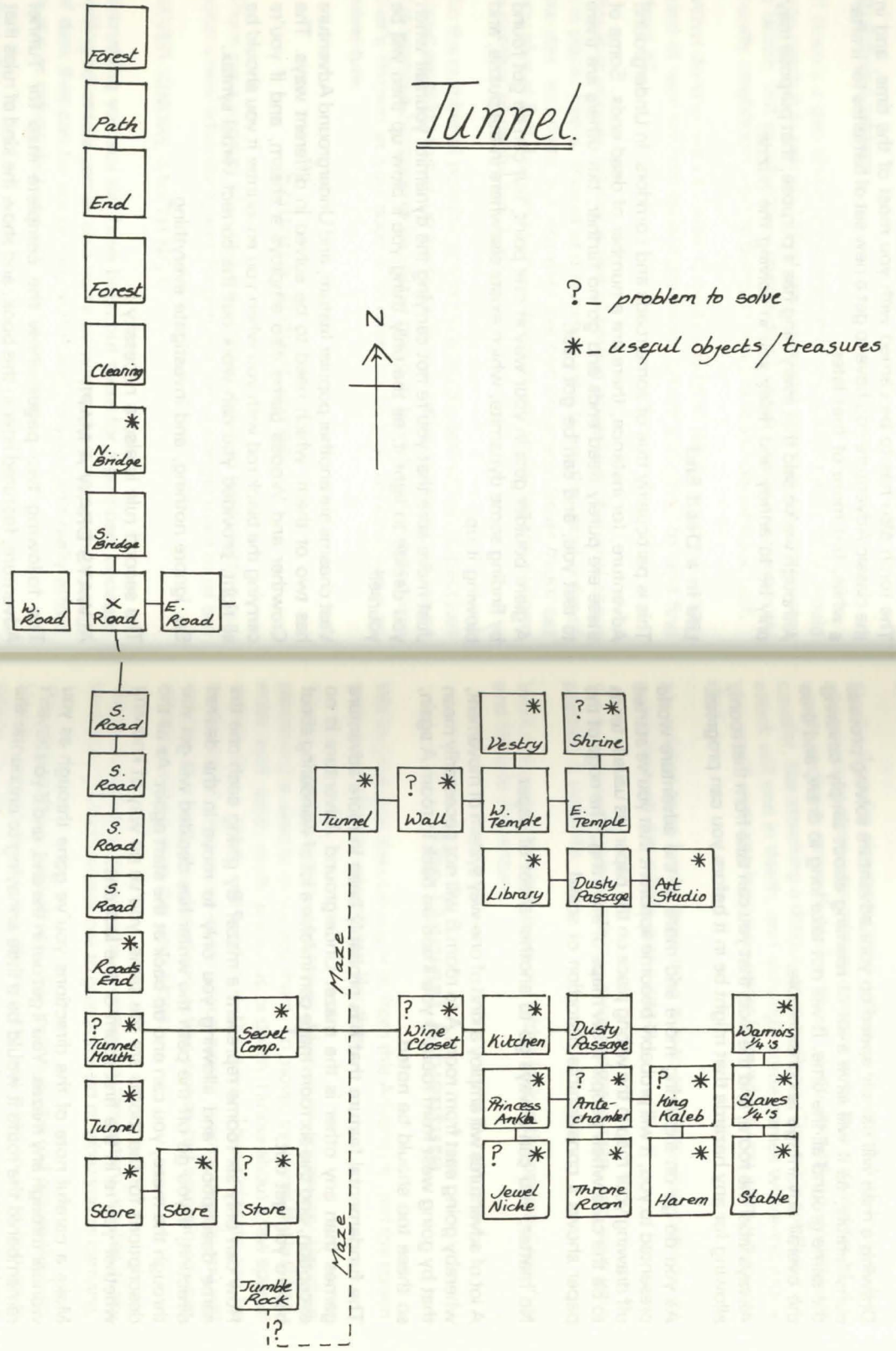
So, ignore nothing, and investigate everything.

The second rule is also a necessity :

ALWAYS DRAW A MAP!!

The following two pages show the complete map for Tunnel Adventure, featured later in this book, and show the kind of rules that should be obeyed when drawing a map of your own.

Tunnel


Forest
Path
End
Forest
Clearing
N. Bridge
S. Bridge
W. Road
X Road
E. Road
S. Road
S. Road
S. Road
S. Road
Roads End
Tunnel Mouth
Tunnel
Store
Store
Store
Jumble Rock
Secret Comp.
Maze
Maze
Tunnel
Wall
Vestry
Shrine
W. Temple
E. Temple
Library
Dusty Passage
Art Studio
Wine Closet
Kitchen
Dusty Passage
Warriors 1/4 '3
Princess Ankha
Ante-chamber
King Kaleb
Slaves 1/4 '3
Jewel Niche
Throne Room
Harem
Stable
N
? - problem to solve
* - useful objects/treasures

Drawing a map

Drawing a map will usually speed up your adventure solving process considerably, as it will save a lot of roaming about simply covering the same ground all the time. It will not take long to draw, and thus the overall advantage is considerable.

Always label the room, and the exits that you can take from that room, allowing for any hazards that might be in it before you can progress.

As you do go on exploring more and more of the adventure world presented to you, it will probably become apparent that you've started off drawing your map in the wrong place on the paper, as usually tends to be the case when people draw maps of anything. The edge of the paper shows a considerable distortion of scale!

No matter, you can always go to another sheet of paper.

A lot of adventures will employ a kind of one-way system of movement, whereby going east from room A to room B will not necessarily mean that by going west from room B you'll end up back in room A again, so these too should be noted.

The fundamental feature that has probably been in more adventure games than any other is the maze. Underground Adventure is no exception, and this six room maze can involve a lot of wandering about before you get out.

How can only six rooms represent a maze? By giving each one the same description, and allowing you only to move in the desired direction, if you go off the path the writer has decided will get you through the maze, you can end up back at the start again. As all the descriptions for the rooms are the same, you've no way of knowing whether you're in the first room or the last one.

Make a careful note of the directions you've gone through as you wonder through any mazes. You'll get out in the end, and if you haven't remembered the route it would be a little annoying to encounter the maze again in a later game.

Drawing maps can be fun, and it's useful to note down the initial positions of any objects that you find during play. Some adventures do have a random distribution of objects, but more of them do not,

as the solving of many puzzles depends on finding the correct object in the correct room, and if it isn't there then the game becomes unsolvable.

Finally, if you're playing a game with a LOAD and SAVE feature that allows you to store your current position onto tape for later recall, it's worth saving a game if you're about to do something particularly cavalier, like attacking a dragon or something. The odds are that your attack will end in death, and although some games will re-incarnate you, you'll end up a long way away from where you were when you died.

It's quicker to re-load a tape than it is to re-create your position by going through the whole game again.

So to sum up, ignore nothing, always draw a map, and save your position if possible.

We'll now put all this into practice, with a look at the original Crowther and Woods Adventure.

The Original Adventure

We've given you the opening lines from this Adventure, and the screen goes on to display something like this:

'I know of places, actions and things. Most of my vocabulary describes places and is used to move you there. To move, try words like building, enter, east, west, north, south, up or down. I know about a few special objects like a black rod hidden in the cave. These objects can be manipulated using some of the action words I know. Usually you will need to give both the object and action words, but sometimes I can infer the object from the verb alone.

'Some objects also imply verbs. In particular, "Inventory" implies "Take Inventory", which causes me to give you a list of what you're carrying.

'The objects have side effects. For instance, the rod scares the bird. Usually people having trouble moving just need to try a few more words. Usually people trying unsuccesfully to manipulate an object are trying something beyond their (or my!) capabilities and should try a completely different tack.

'To speed the game you can sometimes move long distances with a

single word. For example, "Building" usually gets you to the building from anywhere above ground, except when lost in the forest. Also, note that cave passages turn a lot, and that leaving a room to the north does not guarantee entering the next from the south. Good luck!'

And finally, you get one last piece of help before being thrown into the game proper:

'Maximum points are earned by leaving treasure in the building. It also helps to get back out in one piece.

'If you think you have found all the treasure, keep moving around until something happens.'

And So We Begin

And with that, the game will begin, and you find yourself in a building known as the well house (since it contains a well!), which houses a number of useful objects like a torch, a bottle, some food and a key.

From the building it is but a short walk to the forest, which is very easy to get lost in, and then the real route into the heart of the game takes you south down a narrow ravine until:

'You are in a 20 foot depression floored with bare dirt. Set into the dirt is a strong steel grate mounted in concrete. A dry stream bed leads into the depression.'

Opening the gate with the key provided in the building lets you into an underground set of passages, starting off with:

'You are in a small chamber beneath a 3 by 3 steel grate to the surface. A low crawl over cobbles leads inward to the west.'

Of Black Rods, Birds and Cages

Nearby you can find a black rod, a bird cage and the little bird itself, and your first problem solving comes in actually getting hold of the bird, since it isn't too fond of the rod. You'll also have to light the torch by now as well, as it gets dark this far underground. The torch is, in fact, an electric lamp, and it will sooner or later start running down the batteries you started with. However, you are given the helpful message:

'Your batteries are starting to run low. Better wrap it up soon, unless you can find new ones. I seem to recall that there's a vending machine somewhere in the maze.'

Finding the vending machine in the maze is no easy task, and even when you get there you must be armed with a set of coins which are to be found somewhere within the game, otherwise you won't be able to insert the coins to get the fresh batteries contained in the machine!

But back to the bird, the rod and the cage. Wandering on a little soon brings you to the first major room description of the game, which is when you start to realise why this game is a disk based one: some of these room descriptions can get quite long!

'You are at one end of a vast hall stretching forward out of sight to the west. There are openings to either side. Nearby a wide stone staircase leads downward. The hall is filled with wisps of white mist swaying to and fro almost as if alive. A cold wind blows up the staircase. There is a passage to the top of a dome behind you.'

Round about here you will also encounter a snake, which bars your way and refuses to let you pass.

Snaky Problems

Feeding animals is the usual way to calm them down, but attempting to feed the snake is not a particularly good idea, especially if you're carrying the bird at the time, since the snake eats the bird and then just sits there looking at you, still refusing to let you pass.

You can solve that one for yourself!

Round about here, you have a choice of routes, and one of them leads off across the floor of the hall as far as the aforementioned vast chasm, which is where the rod comes in useful. Going on from there will take you towards the maze with the vending machine in it via a back entrance, but it will also take you near another maze as well, which is significantly more difficult to get out of.

It also contains something a lot more interesting, but we'll come to that one later.

Going off in another direction leads you to the mysterious Y2 room, and nearby lies the equally mysterious bedrock room, which allows

you to explore around at random.

From Y2 you can go to another one of the game's fine room descriptions, which is one of the more puzzling points on the route for beginners to the game:

'You're at a low window overlooking a huge pit, which extends up out of sight. A floor is indistinctly visible over 50 feet below. Traces of white mist cover the floor of the pit, becoming thicker to the left. Marks in the dust around the window would seem to indicate that someone has been here recently. Directly across the pit from you and 25 feet away there is a similar window looking into a lighted room. A shadowy figure can be seen there peering back at you.'

Who, or what, is the shadowy figure?!

Dwarves and Pirates

From here we have a variety of routes, but by now a couple of things will probably have happened. One is that you will almost certainly have encountered a dwarf:

'A little dwarf just walked around a corner, saw you, threw a little axe at you which missed, cursed, and ran away.'

Charming!

And the other is a bearded pirate, who lurks about the caves, and who will occassionally appear and steal all your treasure:

'Out from the shadows behind you pounces a bearded pirate! "Har Har", he chortles, "I'll just take all this booty and hide it away with me chest in deep in the maze!" He snatches your treasure and vanishes into the gloom.'

Since some of the treasures have a useful function to fulfil, as well as just being valuable and scoring points when you get them back out to the building, this can be mighty inconvenient!

One of these dual purpose treasures is a trident, which lurks away near the bedrock room. As well as being jewelled, it will also enable you to solve one of the game's more puzzling features.

Mysterious Bivalves

Near Y2 there lives a giant clam, although we later find out that it is in fact an oyster. The program cheerfully tells us that it was never very good at identifying bivalves after this little bit of mistaken identity. Being an oyster, it will probably contain a pearl, and so you attempt to open the clam without success.

You can carry it about with you if you want to, although it is a little heavy, but you won't be able to open it until you find the jewelled trident, which is hidden away in a secret set of rooms, which are themselves reached via the two pit room, or twopit room, as one acquaintance used to call it.

In the two pit room is a plant, and like all plants it likes being watered. Water it enough and it will grow and grow until it reaches the height of a hole way above your head. You can then climb the plant and get into this new set of tunnels and corridors, until you realise that your progress is halted once again as you run into an old rusty door that needs oiling.

Oh well, there is some oil in here somewhere, so having found that you can then get past the door and find the jewelled trident. You'll have to get away from there then, which is none too easy, but can be accomplished.

A Breath-Taking Description

One of the longest of all room descriptions is to be found round about the low room, near bedrock, and is worth repeating here in full just to show you the sort of advantages disk-based systems can give you over programs stored purely in memory, in terms of the use of text to illustrate graphically what a room looks like :

'You are on the edge of a breath-taking view. Far below you is an active volcano, from which great gouts of molten lava come surging out, cascading back into the depths. The glowing rock fills the farthest reaches of the cavern with a blood-red glare, giving everything an eerie, macabre appearance. The air is filled with flickering sparks of ash and a heavy smell of brimstone. The walls are hot to the touch, and the thundering of the volcano drowns out all other sounds. Embedded in the jagged roof far overhead are myriad twisted formations

composed of pure white alabaster, which scatter the murky light int sinister apparitions upon the walls. To one side is a deep gorge, fille with a bizarre chaos of tortured rock which seems to have been crafte by the devil himself. An immense river of fire crashes out from th depths of the volcano, burns its way through the gorge, and plumme into a bottomless pit far off to your left. To the right, an immense geyse of blistering steam erupts continuously from a barren island in th centre of a sulphurous lake which bubbles ominously. The far righ wall is aflame with an incandescence of its own, which lends a additional infernal splendour to the already hellish scene. A dark foreboding passage exits to the south.'

Wow! Try getting all that into a single picture on the screen. The min can imagine far more readily what a place looks like from a descriptio like that than it can from a poor graphical illustration on the screen

Round about here you can also find an extremely narrow crack tha you can't get down with anything that you happen to be carrying and also a troll, who is not too fussed about eating, but who doe have a streak of avarice in him.

Trying to attack him produces the response :

'Trolls are brothers of the rocks and have skin like that of a rhinoceros He fends off your blows effortlessly.

Even if you try throwing an axe at him, all you'll get is :

'The troll catches the axe, examines it, and tosses it back to you saying "Good workmanship, but not very valuable".'

A tricky customer the troll, and you'll have your work cut out to ge around him without losing too many points.

On the other side of the troll is another set of passages, including th breathtaking view described earlier, and also including a bear, wh can be bribed with some food, and who can then be used to scar away the troll when you want to get back across the bridge again

However, the bear is heavy, and the bridge is old, and the inevitabl happens ... you plunge to your doom on the rocks below.

And so the game continues, through many different rooms and wit many different problems to solve, and space dictates that we can mention them all here. Even with what we've already told you, there

more than enough in this game to keep you occupied for a long time yet!

But one final feature does deserve mention, and that is the end of the game itself, after you've found all of the treasures and taken them back to the building.

The End Game

As you wander about the caves, convinced that there's nothing more to find, a sepulchral voice booms out and tells you that the caves are closing, and you'd better leave by the main exit.

But where is the main exit?

So off you scurry to try and find a way out, but always its too late, and the caves close! As they do so, mysterious forces snatch your keys out of your posession, and a few other items as well for good measure, and you find yourself:

'. . . at the northeast end of an immense room, even larger than the giant room. It appears to be a repository for the "Adventure" program. Massive torches far overhead bathe the room with smoky yellow light. Scattered about you can see a pile of bottles (all of them empty), a nursery of young beanstalks murmuring quietly, a bed of oysters, a bundle of black rods with rusty stars on their ends, and a collection of brass lanterns. Off to one side a great many dwarves are sleeping on the floor snoring loudly. A sign nearby reads DO NOT DISTURB THE DWARVES! An immense mirror is hanging against one wall, and stretches to the other end of the room, where various other sundry objects can be glimpsed dimly in the distance.'

And if you get out of that room? You enter this one :

'You are at the southwest end of the repository. To one side is a pit full of fierce green snakes. On the other side is a row of small wicker cages, each of which contains a little sulking bird. In one corner is a bundle of black rods with rusty marks on their ends. A large number of velvet pillows are scattered about on the floor. A vast mirror stretches off to the northeast.

'At your feet is a large steel grate, next to which is a sign which reads TREASURE VAULT. KEYS IN MAIN OFFICE.'

And what happens then? Well, you'll just have to play it all and fin out for yourself!

We've given a lot of exposure to Adventure here, because it was th first serious adventure game, and holds many a fond memory fo everyone who's ever played it, whether on a PDP-10, or a Commodor PET.

It also contains most of the ideas which have influenced othe adventurers over the years, and as such is more than worthy of it place here.

Try your local user group if you're thinking of getting hold of a copy It'll be worth it, but you won't get much sleep after you've got it.

But since Adventure, there have been many others to solve, so we take a look at some of those now.

Other Adventures

The other main contender in the adventure game stakes is obvious Scott Adams, who's done so much to popularise these games o microcomputers.

We've taken a brief look at some of his games earlier on in this book but to go into a little more detail on some of them, we'll start wit the very first one he wrote, Adventureland.

This is a very natural romp, in that most of the features you encounte are perfectly natural, such as bogs, lakes, and a tree (which mus become a tree stump before you can get very far into the game), a well as the nasty chiggers. Nasty what? Look it up in the dictionary

It's all very lighthearted, and a nice sense of humour runs throughou the game. A good starting point for anyone who's fairly new to th adventure world, as solving it is not too complicated. Nevertheles it should keep you entertained for quite a while.

As will the second Adams adventure, Pirate Adventure, with a sto line developed by his wife. This one probably more than Adventureland set the standard that Adams was to adhere to throughout his gam writing series.

There are four main locations for this adventure, including a Londo

Apartment, an Island, a Treasure Island, and Never-Never Land. It was one of the first games of this genre to give you a mission other than pure collection of treasures, in that you have to work out how to build a boat!

Along with some of the characters who inhabit this world, such as the parrot that keeps shouting 'Pieces Of Eight', and who does give you some helpful hints along the way, this is a nicely humourous game.

Mystery Fun House came next, and differs from the usual run of the mill games by taking place in a carnival fun house. All sorts of problems to solve, and many, many corridors to explore, and this was the first Adams adventure to pit you against a time limit, as well as all the other problems.

Mission Impossible has appeared on more computers than possibly any other Adams adventure, and is one of the most difficult ones that he's done. It's also a mission adventure, rather than a treasure collecting one, in that you're on a race against time (as in Mystery Fun House) to try to stop a nuclear reactor from being destroyed by unknown enemies.

Spaced Out

Strange Odyssey is set in another world altogether, as it starts with you all alone on a strange planetoid, with only a shattered spaceship and your own skills as an adventurer to protect you.

Many outer space games have appeared over the years, and in a brief aside we'll take a look at a couple of non-Adams ones, starting with A Stellar Trek.

This is another version of the final frontier, where you boldly go where no computer has gone before, you are in command of the starship Enterprise, and have the simple task of defending the galaxy against the threat of the invading Klingon empire and their friends the Romulans.

This is more of a role-playing game than the true textual adventure, in that you must begin by selecting your crew and adopting various tactics that will stick with you throughout the game.

None the less, our basic rules of ignoring nothing and drawing maps still apply, although as we'll see in another game there are instances

where examining everything in sight can lead you into great trouble!

This is basically a graphical game, and some may not find it to their liking if they're aficionados of the real thing. Still, an enjoyable and frustrating game, that should keep you out of trouble for a while.

Two other games that can dubiously be described as fitting into the adventure world, although really they are more at home with the Dungeons and Dragons fanatics, are Starfleet Orion and Invasion Orion. These are war in space games, with a lot of tactical planning and craft maneouvering going on, and so don't really belong as true adventures. But, like A Stellar Trek, they should keep you amused for a while.

Back to Normal

A new venture for Adams was away from space and into the world of vampires and other assorted nasties.

In The Count you are out to rid the world of Count Dracula once and for all, and, in the best traditions of ancient horror movies, you must race against time to catch the count in his human form before driving the stake home and removing him from the planet.

Voodoo Castle is set along similar lines, with you involved in an attempt to save the cursed Count Christo, which sets you off exploring the hallways and dungeons of Voodoo Castle. An entertaining game, with voodoo dolls, a juju man, and more.

The final two we'll mention from Adams are again set in two totally different worlds, with Pyramid of Doom taking you to an unexplored pyramid somewhere in the depths of Egypt. This is one of the more difficult Adams adventures, and many would say the hardest one he's ever put together.

When you begin writing your own adventures, you'll find that one of the most dfficult things to judge is precisely how difficult you're going to make the game.

Since you control the rooms, the objects in them, and the problems that have to be solved, the game can effectively be made as easy or as difficult as you like. As you're going through it, you may well find yourself thinking that this is a very easy game, and no-one would ever have any problems solving it.

Well, remember that other people haven't got access to your maps, your route diagrams, your list of objects and their original locations, and so on.

The easiest solution to this is to get an adventure playing friend to come around, once you're satisfied that the adventure is complete and bug-free (it won't be, of course-your friend will type in something you never thought of, and the computer will be equally as stumped), and have him sit down and play the game, while you hover nervously in the background.

From his reports, you can then modify the game, making it more or less difficult, depending on how it's all gone.

The Wild and Woolly West

In an adventure theme that hasn't seen too much experimentation, although Lost Dutchman's Gold comes near the same area, the last of the Adams games, Ghost Town, sets you in an American ghost town that has you expecting John Wayne and Audie Murphy to put in cameo roles.

Good fun, as you encounter saloons, jails, boot hill, piano playing ghosts, and a whole collection of ludicrous characters, this is a suitable Adams game to bow out with. A very enjoyable game.

There are plenty of other games out there that are worthy of exploration, but for our last one in this section we'll take a look at the game that's been described as being as much of an improvement on Adventure as Adventure was on Wumpus.

What's Wumpus? One of the most boring computer games of all time, where you have to walk around a few (typically 24) rooms trying not to bump into the Wumpus, an amiable beast who likes to spend most of his time asleep. A few arrows can be fired now and again, but overall it does not rate very high on the entertainment stakes.

To say a game can improve on the original Adventure by that much is a bold claim, but Zork has captivated everyone who has ever played it.

Now in three parts, sold on three separate disks, each part is a unique adventure in its own right, and pits you against wonderful problems in weird worlds, but with a number of great improvements over that original game.

Zork: the Greatest Adventure?

Zork was the brainchild of four people: Marc Blank, Tim Anderson, Bruce Daniels, and P. David Lebling, and (like our original Adventure), was written on a PDP-10.

However, as Zork grew and grew it began to run out of memory space even on that computer (at the time a giant megabyte, but that doesn't look too much now), and they decided to completely re-write the game for a microcomputer.

A strange decision? Well, not really, because most microcomputers even then had disk drives, and now of course these disk drives are growing in capacity.

However, to re-write Zork in order to make it all fit was no easy task. It might be possible to fit all the data and text required for the game onto a single disk, but what about the program to manipulate it all? Even the original Adventure control program takes up about 13K without running it, and as you probably know, as soon as the program is run, various variables are declared that take up even more memory space.

So Zork had to undergo a few drastic changes.

The first of these was to write a Zork-language, which could be swopped from machine to machine merely by changing that language to suit the machine, and then write all of the program in the Zork-language.

In other words, just as all micros require a different Basic interpreter, so the Zork interpreter swops around from machine to machine. However, the rest of Zork can remain the same, and so the actual work-load on the authors was considerably reduced. Only the interpreter had to keep on being re-written, and now exists for just about every popular make of computer.

The complete story behind all this can be read in a very interesting article in the July 1980 issue of *Creative Computing*, called 'How to fit a large program into a small computer', which was co-written by one of the authors of Zork, Marc Blank.

Having crammed Zork into a small machine, it was now available to many people, and some of its features are truly amazing.

The ability to say more than just DROP BOMB for instance, which can now be said in a variety of ways. For example, TELL THE ROBOT TO PUT THE BOMB ON THE SHELF, and other variations, do much to add to the power, and ease of use, of this game.

Such control over the vocabulary is beyond the scope of this book, although we will be taking an extensive look at string handling in chapter 3.

Suffice it to say that if you can get hold of a copy of Zork, do so! We've given you a few addresses at the back of the book.

But is Zork the ultimate adventure? With graphical and role-playing games coming more to the fore, let's take a look at some of those, and see if we can guess what will happen over the next few years.

Graphical and Role-Playing Adventures

We've already talked about graphical adventures of the future in an earlier section, and will end our discussion here with the same sort of conclusion as was reached then: not many people want to see fabulous displays on the screen, when fabulous descriptions can conjure up far more in the mind of the player and his alter ego as they wander about the universe created for them.

Instead, the future would tend to lie in the direction of role-playing games, best personified by the original Dungeons and Dragons games, and their variants such as Tunnels and Trolls, Traveller, and the countless other board games that have sprung up since the first game appeared.

In these games there is one great difference over the classic Adventure/Zork scenario: you adopt a character role, rather than just taking on the one that the computer conjures up for you, and your success or failure in the game depends to a very large extent on th type of character adopted.

In Adventure, you know that if you get the bear you can always get back past the troll again, and escape over the rickety bridge to (comparative) safety, but if the same situation were to occur in one of these games that might not always be the case. Your character might be unfriendly towards the bear, and the bear would bite your hand off, or some other dire fate might befall you.

Again, in Adventure, a fight with a dwarf will always have one of two options. You will either win and emerge unscathed, or lose and die.

A fight in a D and D game could have a number of different outcomes, as well as the two simple ones outlined above. You might win the fight, but suffer a gaping wound that leaves you temporarily below your best: an easy victim for the next antman who comes along.

So that is the chief difference: the games are more varied, and indeed one could argue a strong case for there being an infinite number of variations contained within the same game.

However, these advantages are not gained without some other advantages being lost.

In Adventure and Zork you have a vast vocabulary at your disposal (Zork can handle over 600 different words, with about 100 verbs to be used), but in D and D games you're usually restricted to a much smaller number. This is typically of the order of 20 commands, or even less: the much rated Temple of Apshai has a very small vocabulary indeed.

Still, you do have the option of choosing a character who is much more to your liking than a simple 'You are', appearing on the screen every time. It is far more satisfying to watch 'Pete The Great' stalking about the screen (or whatever you would choose, of course), and for some reason it seems to make the game a lot more realistic if you know that it is more specifically YOUR success or failure in the game that's at stake.

From Boards to Computers

One of the less attractive attributes of Dungeons and Dragons is that it takes a referee to make sense of it all, and bribing the referee, all part and illegal parcel of playing the game, has been known to sway many an outcome. The real life 'I'll buy you a pint when we've finished' is far more likely to influence the referee than a simple 'How about 20 gold pieces then?' whilst in the middle of a game.

Also, the referee's job is not an easy one, as most actions have to be decided by a concentrated study of maps, charts and rules, and thus a simple fight between two protagonists could take as long as half an hour, or even more, to resolve.

In computer simulations of these games the computer becomes the referee, and the screen the board on which all the action takes place, and as these games are always played in real time that action can sometimes be decided very rapidly. Our half-hour fight could be over in ten seconds, and it's back to the keyboard in a hurry to see what damage you and your trusty sword have suffered in the duel.

Character Traits

Your character in these games is determined by six factors: three mental (ego, intelligence and intuition), and three physical (dexterity, strength, and constitution).

In the old days these were decided by rolling three dice and adding up the spot scores, and thus any one attribute could range from a low score of three to a high score of eighteen. On the computer, you can usually choose from a total score and divide that score up amongst the six attributes, and since our scores can range from 3 to 18 for each one (or 16 different possibilities), we can create a massive 16 to the power 6 different characters, or over 16 million!

Since we are creating each character as we go along, we can also bring in characters from other games, which helps to explain the popularity of this type of game. If you've survived an exhaustive game of Dungeons and Dragons as Denis The Unsteady it helps to have the same character with you next time you set out to play The Curse of Ra, or whatever.

These six differing character straits interact subtly throughout any one game, and the final outcome of that game always depends on the role you have adopted. A highly intuitive character will find secret trap doors with ease, whereas one with low intuition would only find them by falling into them. Similarly, a high ego would keep going when the going got tough, but a low ego would probably cry and ask for his mum!

And so it goes on, with each attribute perhaps altering slightly as the game progresses and you discover magic potions, bargain in the Apothecary Shoppe, and in any one of a hundred different situations.

Perhaps this is the true way forward for adventure games in the future, and increasingly role playing will play a dominant part in this type of game.

The Ideal Way

The ideal would be to have a combination of the traditional textual adventure with a character playing role as well, as graphics are largely redundant in these games. Thus one would keep the advantages of a large vocabulary, a large number of locations, and a large number of hazards and problems to solve, whilst at the same time having a multitude of variations on the same game by being able to pick your character from one of the 16 million mentioned earlier.

But that must wait for another time, and for now we'll turn our attention to the game Underground Adventure, which will be featured heavily throughout this book, and we'll begin by explaining what it's all about.

Underground Adventure

This is a classic text-only one character game, because for writing your first adventure I don't feel that we should be too optimistic. You may well, after reading this book and understanding everything that's going on, want to go on and develop extremely complicated games, and if you do then the purpose of this book will have been achieved.

But for now, we'll describe a simple, straightforward adventure that is (I hope!) a lot of fun to play and solve.

Underground Adventure starts you off outside a series of caves, with dire warnings about the punishments that await anyone who enters. But, being a brave young lad with a heart for adventure you merrily march off into the caves, take three steps inside and CLANG! A massive gate falls shut against the entrance to the cave, and from then on it's a question of roaming around trying to find the key that will enable you to get out again.

There are a total of 16 problems to solve in this adventure, and I've tried to give you a feeling for the real thing by including a number of scenes that will be familiar to anyone who's ever played an adventure before. In later chapters we'll explore the actual writing of those scenes, and show that it is all possible in Basic, but for now we'll content ourselves with simple descriptions.

You start off immediately with a choice of three routes, heading either east, west or south. North is closed off to you because of the fallen gate.

To the east lies a massive underground tree, which completely blocks your path, so you know that one of the things you'll have to do is to find a way of getting past that tree. What do most people do when they want to remove a tree? They either drag it away or they chop it down, so you know you're looking for either some haulage equipment (unlikely in an underground cave), or something like a sword or an axe.

To the west, your path is blocked by an extremely large boulder, that fills up the whole path and prevents you from going any further. To get past this, you might first of all try pushing and pulling at it, or even attempting to pick it up, but the stone is too heavy for you to move that easily. So, again you must ask yourself the question 'what would anyone do when they wanted to remove a large boulder'. Well, again one could haul it away, but that seems a little unlikely. There could be a large animal around somewhere that might move it for you, or perhaps you'll need to blow the thing up with some dynamite. Of the latter two options, one is the correct one, so we keep an eye out for either a large animal or a keg of dynamite. Beware of large animals though: most of them are not very friendly on the first encounter.

To the south, all we can find is a vast chasm, but en route to it we've already picked up an iron staff, amongst other things. Examining the staff reveals that 'it has some useful properties', so we know that the staff is capable of solving something. Since we can't go west, east or north without finding yet more objects it's reasonable to assume that we'll have to go south somehow. Attempting to jump the chasm is not very rewarding, and in fact leads only to your death, so perhaps if we wave the staff...

Ah, perfect, and a bridge now spans the chasm. Good, we can now head south into the heart of the caves and see what we can find. If we're unlucky, a living gargoyle will appear, and throw a knife or two at you, and he must be engaged in combat before moving on, otherwise he follows you everywhere, continually throwing knives, and one of them may find its mark. How to kill a living gargoyle? Well, there must be something dangerous around somewhere, and sure enough we find an axe eventually. A sure throw with the axe finishes off the gargoyle (temporarily), and we remember that an axe was one of the things we were looking for, as a possible means of chopping the tree down.

Back across the bridge, chop the tree down, and we find some rope, which must come in useful somewhere for climbing up or down something, and a golden bear, who appears none too friendly. Obviously the bear must be calmed down somehow, but how ?

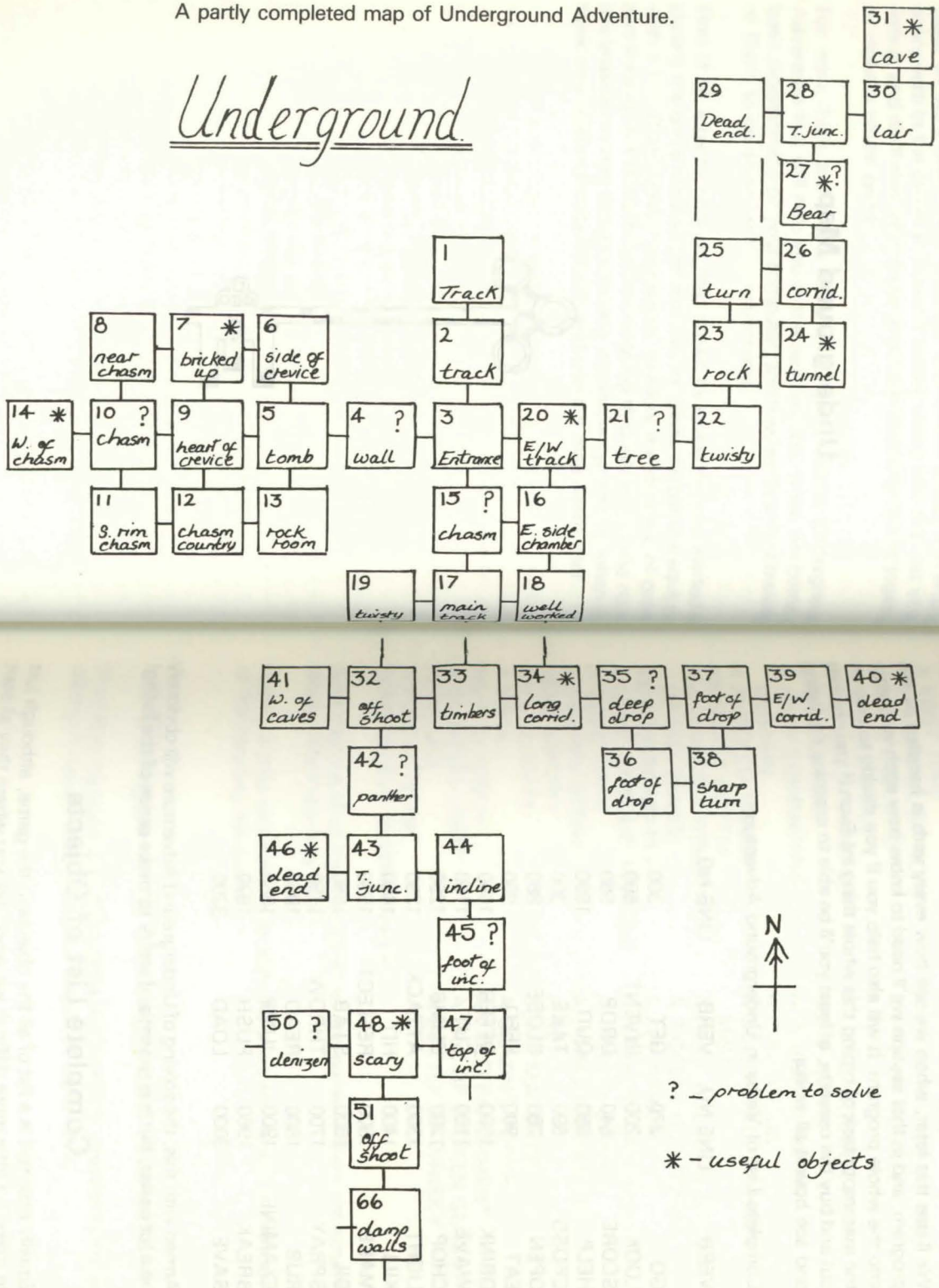
That one, and a few other problems, we'll leave up to you, but yo will have begun to get the idea of solving this adventure. Everythin is there for a reason, and solutions to problems are usually quite logica Later on we'll take a very thorough look at this game, and analys every verb in the game and how it works, along with the rest of th listing, and we'll also see how each part of the listing comes togethe to make a whole game.

For now, here are a few facts and figures about Undergroun Adventure that will help you in the next section, when we come t basic programming on your computer, and how we'll use a knowledg of Basic to go about writing adventures.

First of all, a partly finished map drawn by someone who started o playing this game but then ground to a halt. Since the game is equippe with a LOAD and SAVE procedure, to allow the stopping of game and subsequent re-starting without going through everything agai it is possible to stop this game at any convenient point (i.e. when yo think you're about to be killed), and use the map again later.

Underground Map

A partly completed map of Underground Adventure.



List of Verbs

We'll use this later, when we see how every verb is handled by the program, and in that section we'll need to know how each verb slots into the whole program. It will also help you if you decide to take on the mammoth task of typing this whole thing in! Even if you chicken out and buy the cassette, at least you'll be able to examine the listing and see how it all works:

Complete List of Verbs in Underground Adventure

VERB	LINE NO.	VERB	LINE NO.
GO	270	GET	300
LOOK	200	INVENT	500
SCORE	540	DROP	560
HELP	650	QUIT	1890
CROSS	690	TAKE	300
OPEN	780	CLOSE	880
EAT	900	FEED	950
DRINK	1000	OFFER	1050
WAVE	1100	CUT	1150
CHOP	1200	CLIMB	1250
LIGHT	1300	ATTACK	1350
KILL	1400	HIT	1450
MAKE	1500	REFLECT	1550
OIL	1600	STAB	1650
SPRAY	1700	THROW	1750
RUB	1800	READ	1850
EXAMINE	1900	JUMP	1950
BREAK	1960	PUSH	1970
SAVE	3000	LOAD	3200

Armed with this, the solving of Underground Adventure will obviously be a lot easier, but it is essential if we're to make sense of the listing!

Complete List of Objects

Equally essential is a list of all the objects in the game, although just to make it a little more difficult we won't tell you where they all start off. However, by the time you've finished this book you'll be able to work it all out for yourself, if you want to cheat!

OBJECT	OBJECT
A VAST CHASM	AN IRON STAFF
A VAST TREE	A STOUT AXE
A THICK COIL OF ROPE	AN ENCHANTED BRIDGE
SOME DYNAMITE!	A PILE OF RUBBLE
A GOLDEN BEAR	A BUN
A BIG BLACK PANTHER	A LONG WOODEN PLANK
A TALL LADDER	SOME NAILS
A HAZY SHIMMERING CURTAIN	A POLISHED MIRROR
A BLOCKED TRACK	A POOL OF OIL
AN EMPTY BOTTLE	A SOLID WALL OF HAZY MIST
THE GHOSTLY DENIZEN OF THE CAVES!	A HUGE BULBOUS SPIDER
AN ENORMOUS FLY!	A RICKETY OLD DOOR
A LUMP OF SOLID MORTAR	A FLY SPRAY!
A SOLID GATE	A NARROW CRACK
A SHINING STONE	A TRUSTY SWORD
SOME WHISKY	A LIVING GARGOYLE!
AN EVIL KNIFE	A KEY
A WALL	SOME MATCHES
AN OLD TORCH	A BLAZING TORCH
A GLOWING LIGHT	AN OLD PARCHMENT
PROGRAM	A PILE OF BROKEN GLASS
A BOTTLE OF OIL	A BOTTLE OF WHISKY
SOME NICELY SAWN TIMBER	

Note that not all of them are objects, and some of them are actually places. We'll see why later.

Finally, a little bit of dialogue with the program, which is the result of first starting the game.

A Dialogue with Underground Adventure

The following is one way that a game night start off, with the computer talking in upper case, and your entries in lower case:

YOU ARE ON AN OLD TRACK HEADING TOWARDS THE CAVES.

YOU CAN SEE :

AN OLD TORCH

YOU CAN GO : SOUTH

WHAT NOW? * (the prompt symbol)

get torch

OK.

WHAT NOW? *

s (or south, or go south)

YOU ARE GETTING EVER NEARER THE CAVES.

YOU CAN GO : NORTH SOUTH

WHAT NOW? *

s

YOU ARE AT THE ENTRANCE OF THE CAVES,WITH PATHS LEADING EVERYWHERE.

YOU CAN SEE :

A SOLID GATE
SOME MATCHES

YOU CAN GO : NORTH SOUTH EAST WEST

WHAT NOW? *

get matches

OK.

WHAT NOW? *

get gate

I CAN'T DO THAT!

WHAT NOW? *

light torch

OK.

inventory

YOU ARE CARRYING :

SOME MATCHES
A BLAZING TORCH

WHAT NOW? *

e

OH DEAR, THE GATE TO THE CAVES APPEARS TO HAVE SLAMMED SHUT!

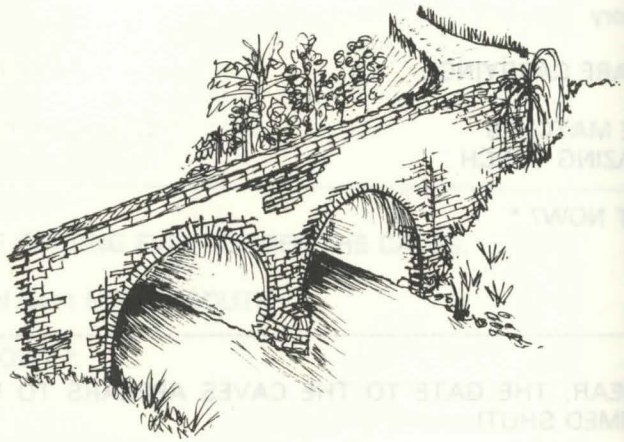
THAT'S TORN IT! YOU'LL HAVE TO FIND THE KEY NOW BEFORE YOU CAN GET OUT.

BUT DON'T WORRY. IT'S IN HERE SOMEWHERE!

WHAT NOW? *

And so it goes on, with your adventure now well and truly under way.

In chapter 3 we'll take a look at some of the knowledge of Basic required to produce this sort of program.



3

Programming Adventures in Basic

Why Bother?

This ranks alongside asking Chris Bonnington why he climbs mountains, or Patrick Moore why he looks at the stars. It's something that they enjoy doing, for some reason that probably they couldn't even explain if asked to sit down and actually state a concrete set of reasons.

So it is with programmers. People enjoy programming, just as much as some enjoy climbing mountains or some enjoy peering through telescopes. As with any other pursuit, there are a variety of ways of programming, and there are a variety of things to write programs about.

This book will teach you all about programming for one subject, that of adventure games. It will also only teach you one style of programming: of necessity, that will be the style adopted by the author.

However, it is to be hoped that originality will shine through on your part, and you'll go on to produce programs that are wildly different from the ones shown here.

Satisfaction

The major reason why people write any sort of program must be for their own satisfaction, rather than anything else, although one is sometimes tempted to think that the mercenary attitude shines through on occasions!

To complete a program that is over 30K long, as some of the adventures undoubtedly will be, is quite an achievement, and ev if no one comes along and says 'That was great!', at least you'll s be satisfied with it yourself.

All the better then when someone else does play the game, a congratulates you for writing it. It makes all the hours spent pori over the keyboard desperately trying to solve a programming proble worthwhile.

In return, it's nice to think of the person ultimately playing the adventu taking far longer to solve the program that it took you to write i

Money

We mentioned mercenary attitudes earlier, and that is obviously o reason why you should bother writing anything, let alone adventu games.

From one point of view there's always the possibility of seeing the go on sale and being marketed by a reputable company, and that very satisfying.

Another point of view would be that it saves having to buy a lot adventure games written by other people, but that must be a seconda reason. If you've written the program it won't take you too long t solve it, no matter how many random elements you've put in ther

Level of Skill

One does not have to be the greatest programmer in the world in ord to write satisfactory adventure games. You don't need a knowledg of machine code, and the amount of Basic coding that you have t be thoroughly proficient in is not too great: we'll be covering mo of what you'll require in the rest of this chapter.

Essentially we're concerned with string handling, and the number commands in Basic that allow you to manipulate strings is a fai limited sub-set of the language as a whole.

Once one is au fait with those, the rest of Basic required is main standard stuff, with one or two 'tricks of the trade' which we'll b showing you later.

How to Start

The worst thing in the world is to sit down in front of an empty computer, and think 'My God! I've got to write 30K of code!'.

It's akin to the old writer's syndrome of staring at a blank sheet of paper and not having a clue what to write or where to start. Obviously you map things out first, and we'll be looking at that in more detail in chapter 4, as we begin to pull all the separate pieces of knowledge we've learnt together into a coherent whole.

Writing an adventure game is not as daunting a task as you might at first think. Certainly, to look at a listing for an adventure program (perhaps you might care to glance at the listing for Tunnel Adventure, as that is presented in full at the back of the book) is to invite a feeling of nausea as you are confronted by a million and one IF ... THEN, GOTOs and GOSUBs sending program execution careering about all over the place.

However, the listing, when examined carefully and closely, as we shall be doing, does eventually begin to make sense, and you realise that every part of the program is playing its proper role in keeping the whole thing running, whether it's an INPUT routine that stops you entering the wrong type of information, or dropping out of the program; whether it's a routine to handle movement of the character from one room to another; or whatever it is doing, it's all there for a purpose, and later on we'll find out exactly what all those purposes are!

Cooking

What? No, you haven't stumbled into the wrong book, but a useful analogy with programming adventures is to think of the problems posed to a chef, when he/she is presented with a set of ingredients, and told to come up with the finished meal.

We'll present you with a set of program subroutines that handle various tasks, and in this first program we'll also give you the recipe and make them into the finished program.

We may not turn you into the Robert Carrier of the adventure programming world, but at least we'll get you doing more than just making beans on toast!

A Brief Outline

Just to let you know what's coming up, the next section in this chap will be devoted to learning the commands that are essential to t producing of good adventure games, with obvious emphasis on t string handling ones.

As well as covering the range of commands mentioned earlier (IF THEN, GOSUB and GOTO), we'll also take a brief look at all the oth necessary statements taken in conjunction with their use in adventu games.

This is not meant to replace the Basic programming guide in yo handbook, but at least will enable you to get going.

This will be followed by a short set of listings, all taken fro Underground Adventure, along with a thorough explanation of ho they all work, so that you can use them either as they stand, or suitab amended, in your own games.

Our final guide to writing adventures will concentrate on more examp listings, together with a set of helpful sections on a good procedu to adopt when sitting down and writing them yourself.

We've even given you a number of scenarios for possible games, whi you may like to adapt into your own first adventures!

Underground Adventure is gone through in great detail, with a coup of pages for each verb in our vocabulary, and an explanation of ho the code for that verb works, and by following that you should be ab to make out what Castle and Tunnel Adventures are doing. You shou also learn how to incorporate new words (as you will obviously nee to) in your own programs.

Finally, a round up of information on adventures generally, togeth with a useful set of addresses to contact for further help an information.

Adventure Programming in Basic

In this section we'll look at the commands available to us in Basic f string handling and data handling, and then start tying them up in useful routines.

Input

This is simply a way of typing in information, from a program, that the program will understand and then use in the rest of that program.

However, before we can start using Input, we need to talk about a couple of other things: screen design and the concept of variables.

Screen Design

As we are dealing with text-only adventures in this particular book, it is important to make the screen layout of the game as attractive as possible to whoever happens to be playing it. Games like *The Hobbit*, *Twin Kingdom Valley*, *Valhalla* and the like, can score quite heavily with their use of graphics. However, those programs only work successfully because of the amount of machine code routines contained within them (although even *Valhalla* has surprisingly little machine code in it), and if the graphics were taken away the game would lose a lot of its appeal.

For instance, *The Hobbit* on a Commodore 64 is a surprisingly different program from *The Hobbit* for the BBC computer. Surprisingly, because the story line is the same for the two programs. However, the BBC version has no graphics in it, whereas the 64 version makes extensive use of them. It's one thing to write an adventure program that is purely text based, and add the graphics later (as Scott Adams has successfully done), but it's a completely different matter to produce a graphical adventure and then remove the graphics.

Of the three games featured in this book, two of them rely on white text on a black background, and one on orange text on a yellow background. The difference is quite interesting. The black background games invoke a more 'sinister' atmosphere than the yellow background one, as befits the games. An attractive screen layout (use of upper and lower case, leaving gaps between your suggestions and the computer's responses), helps to keep the interest of the player.

Bearing all that in mind, let's turn our attention to the programming side of things, with a look at the sort of Atari statements that we'll be using.

Variables

A variable is simply a term used to describe a number, or some te
that can be stored in the computer.

There are just two different types of variables allowed on the Ata
namely string and numeric.

Numeric variables are just numbers, and any of the following is a leg
syntax for a variable NAME:

A, A5, AZ, BANANA, JAWS, etc.

That is, the name must be at least one letter long, and must start wi
a letter, and anything after that can either be a letter or a numbe
Moreover, on the Atari we are allowed up to 126 letters with whic
to identify a variable, and so fans of Welsh railway stations with lon
names could quite happily accommodate Llan....goch.

String variables can be numbers, letters, or a mixture of both, an
the restrictions on string variable names are the same as for real vari
bles, with one addition.

String variable names must end with a dollar '$' sign. Thus, all of th
following are legal string variable names:

A$, A1$, ZZ$, FRED$, HOWMUCHISTHATDOGGYINTHEWINDOW
etc.

Most of the more common home computers allow the use of intege
variables as well (e.g. A% = 10.5 would store the value 10 in A%
but on the Atari we have to use the INT command. More of that on
later.

The following would all be acceptable variables of their own individu
al types:

A = 26.45, A$ = "54", AA$ = "I'M A STRING VARIABLE", AZ2$ =
AM 1 2!", etc.

As with all home computers, you cannot use what are known as th
reserved variable words in your string or numeric variable descriptio

Thus a string cannot be referred to as CPTHEN$, as the machine will think of CPTHEN$ as one word, and ignore the fact that you're trying to issue a command like:

IFA<>CPTHEN20

That is, if A doesn't equal CP then go to line 20. Typing in the above command on an Atari will just give you an error message.

Back to Input

Input allows you to type some information into the computer from a program, and that information is stored as a variable. For example:

```
5 DIMA$(25)
10 PRINT "HELLO. WHAT'S YOUR NAME "
20 INPUT A$
30 PRINT "HELLO ";A$
```

would allow you to enter your name, and then say hello to you.

If you tried typing in 1.45 as your name, you'd have been referred to as 1.45! That's because we specified that we wanted a string to be input (A$). Try the following:

```
10 PRINT "HELLO, HOW OLD ARE YOU "
20 INPUT A
30 PRINT "THAT MEANS YOU'RE ";A*365;" DAYS OLD!"
```

If you'd typed in your age as FRED, the computer would have responded with an ERROR- 8 AT LINE 20, as it was expecting a number, not a string, and you'll have to type something sensible in.

If you press Return, and nothing else, the program will continue, but it will treat the string as a null one, i.e. one that contains nothing.

The above examples illustrate the only conventional way of combining PRINT and INPUT statements, since the Atari will not allow you to use text in an Input statement, unlike most other machines. However, there are ways of getting out of a simple input command like the above, so we'll be taking a look later on at some more elaborate ways of presenting input statements that stop the player of your adventure from crashing out of the program.

Data and the Inputting of it

We've already seen that we can get information, or data, into a program by using the input statement, and of course a lot of information could be typed in just by using a lot of input statements.

However, this could get exceedingly tedious if you were using the same information over and over again, hence the need for data statements.

Here the data is typed in as part of a program, read off from within the program, and then acted upon.

Not only does it save you typing in vast amounts of data each time you run the program, but it also allows you to change just one data item, and see how that affects the rest of the program.

In this short example we'll read ten numbers, add them up and then take an average of the whole lot.

```
10 PRINT CHR$(125):REM CLEAR THE SCREEN
20 READ A
25 IF A=0 THEN 40
30 B=B+A
35 GOTO20
40 C=B/10
45 PRINT "THE TOTAL IS ";B
50 PRINT "AND THE AVERAGE NUMBER READ WAS ";C
60 END
70 DATA 1,2,4,5,6,7,3,35,80,43,0
```

The IF ... THEN branching statement in line 25 will be explained more fully later, but here it allows us to stop adding up numbers when we've read ten of them, and reached a number of 0: the last data statement.

Data statements can be anywhere in a program, and if you're reading real numbers, that's what the data statements must contain. If you're reading strings, again they must contain strings. Otherwise you'll get an error message flung at you, and quite right too.

What you must remember is that data is usually read as it is encountered, so wherever it happens to be in the program, make sure that it corresponds to what you want to read.

Also, make sure that you don't try to read more data than you've actually typed in, otherwise yet another error will occur.

Since we make extensive use of data statements in these adventure listings, always ensure that the right data is being read by the right variable, and that the right amount of data is being read.

If you try to read the same data again, you'll usually get an error message appearing, unless you use the ...

RESTORE command

This allows you to re-read data, and takes the following syntax:

```
55 RESTORE
56 GOTO 20
```

GOTO, which transfers program execution from one part of a program to another, will be dealt with in more detail later.

One very nice feature of Atari Basic, and one that is used extensively in these games, is the ability to Restore to a specific line of data. For example:

```
10 DATA1,2,3,4,5,4,3,2,1
20 FOR I = 1 TO 9:READ A:NEXT I
30 RESTORE10
40 FOR I = 1 TO 9:READ A:NEXT I
50 END
```

Here we can read the same set of data twice, merely by making the data 'pointer' point back to line 10 again in line 30.

To finish with data for a while, here's a short example that mixes strings and numeric data:

```
5 DIM A$(5)
10 PRINT CHR$(125)
20 READ A$,B,C,D
30 PRINTA$;" IS ";B;" YEARS,  ";C;"  MONTHS  AND
";D;"  DAYS OLD!"
40 GOTO20
50 DATA PETE,26,5,25
60 DATA BERYL,25,8,22
```

When run, this will generate an error message, as we send it back to line 20 to read more data that isn't there, but the concept is, none the less, a sound one.

IF and THEN

Atari Basic differs from the norm here, since it doesn't have the standard GET command to allow us to read one character from the keyboard at a time. Also, we cannot use the INKEY$ command, since no such thing exists on the Atari.

The following program will illustrate this point:

```
10 PRINT "PRESS ANY KEY"
20 GET A$:IF A$=""THEN20
30 PRINT "YOU PRESSED ";A$;"!"
40 GOTO 20
```

On most computers, this will allow us to read one character at a time from the keyboard. However, on the Atari a more complicated approach has to be adopted, and the routine used in all these adventures to scan the keyboard can be found towards the end of chapter 4.

Using that routine, it is possible (with the aid of IF ... THEN) statements, to send program execution off to specific parts of a program depending on which keys have been pressed. This kind of selective key pressing is one of the principal uses of the IF ... THEN statement.

Its other main role is in decision making according to the value of string or numerical variables.

Strings or numbers can be compared using the greater than '>' and less than '<' operators, which have the following connotations:

```
A > B    : A greater than B
A >= B   : A greater than or equal to B
A = B    : A equal to B
A <= B   : A less than or equal to B
A < B    : A less than B
A <> B   : A not equal to B
```

Thus our program might contain a line something like:

```
100 IF A <= B THEN 200
```


Thus, if A is less than or equal to B then we go to line 200. If A is greater than B we simply slip through to the next line of the program.

Strings are compared alphabetically. Thus "AAAA" is reckoned to be less than "ABAA", and so on, and these can also be used in IF ... THEN statements as above.

Subroutines

Some sections of a program have to be performed time and time again, and it would become very tedious, as well as wasting a lot of memory, if you had to keep typing out the following lines every time you wanted the program to execute them:

```
10 A=B+C
20 D=E+F
30 H=A+D
40 PRINT H
50 REM GET ON WITH PROGRAM AGAIN.
```

Of course, if our program segments were only this long there wouldn't be too much trouble, but as we learn more and more commands the complexity of our programs will grow, and the need to perform repetitive calculations will grow with it.

Thus we have subroutines, lines which are used a lot within a main program, and which generally just perform one specific function.

We'll see how to 'call up' subroutines in the next couple of pages, but the point to be made here is that they too, like the rest of the program, should be REMmed. For instance:

```
5000 REM ************************************
5010 REM * START OF BORDER DRAWING SUBROUTINE *
5020 REM ************************************
5025 PRINT CHR$(125);
5030 PRINT " ++++++++++++++++++++++++++++++++++++++++ ";
5040 A=A+1:IFA=24THEN5060
5050 PRINT"+                                        +";
5060 PRINT " ++++++++++++++++++++++++++++++++++++++++";
5070 REM ************************************
5080 REM * END OF BORDER DRAWING SUBROUTINE *
5090 REM ************************************
```

I don't pretend for a minute that this is the most elegant way of drawing a border around the screen, but at least it works, using only the commands we've so far encountered.

By structuring programs in this way, the REM statement becomes a powerful ally in keeping your programs neat, tidy and intelligible.

More String Commands: LEN

LEN, as you might reasonably guess, is associated with the LENgth of a string.

For instance, if we assign a string A$ to be equal to "A LONG STRING", the command:

PRINT LEN(A$)

would return a value of 13, this being the number of characters (including spaces), contained within the string A$.

We can also assign another variable to be equal to the length of a string, thus:

```
5  DIM   A$(14)
10 A$="ANOTHER STRING"
20 B=LEN(A$)
30 PRINTB
```

Running this would give us the result 14, this being the value of the variable B, or in other words the number of characters in the string A$.

LEN comes into its own when taken in conjunction with the other major string command on the Atari, and the combination of the two is one of the key factors when producing our adventures.

String Handling

Similar in format to the Spectrum string splitting command, the one and only Atari command is limiting in some respects if you're used to the MID$, RIGHT$ and LEFT$ commands of other machines, but if taken in its own right can be an extremely powerful statement. As ever though, great care must be taken over its use, and that is why (for instance) the 'text decoding' routine in our adventures has to be

as long as it is: no one will ever type in what you expect them to.

To compare the Atari command to other, more common ones, we see that:

MID$(A$,I,J)

becomes (on the Atari) A$(I,I+J-1).

That is, take the string A$, and starting at the Ith character in that string extract from it the next J characters.

We'll assign the string A$ to be equal to the name of my home county, Lancashire. So, if we say A$="LANCASHIRE", A$ becomes a string of length 10 characters.

The command MID$(A$,I,J) takes the string A$, starts at the Ith character in that string, and takes J characters out of it.

To give a programming example.

```
10 A$="LANCASHIRE"
20 PRINT MID$(A$,4,4)
```

When run, this would print out the new string CASH: A$ is unaffected. On the Atari, we need to type:

PRINT A$(4,7)

to achieve the same result.

As with LEN, this can also be assigned to another variable. For instance:

```
10 A$="LANCASHIRE"
20 B$=A$(7,10)
30 PRINT B$
```

would result in the string HIRE being printed out, this being the value now stored in B$.

A second string handling command, that would normally be used in adventure writing but which has to be programmed around on the Atari, is LEFT$. Not as flexible as MID$, but none the less a command with its uses when handling strings.

It is a fairly safe bet to assume that this has something to do with the left-hand side of a string, and indeed it does.

Sticking with counties, we'll assign the string A$ to equal "DEVON".

When we issue the following command:

PRINT LEFT$(A$,4)

the result is printed to the screen as DEVO. Thus, with LEFT$ we always start at the first character in the string, and take as many characters as indicated in the argument.

So, in the following program:

```
10 A$="DEVON"
20 B$=LEFT$(A$,3)
30 PRINT B$
```

we would get the rather strange word DEV being printed out.

As you might imagine, to implement this command on the Atari requires a little bit of string manipulation, but the end result is quite straightforward.

The equivalent of LEFT$(A$,5) is:

A$(1,5)

The third commonly encountered string handling command on other machines is RIGHT$. This is concerned with the right-hand side of a string, and works in pretty much the same way as LEFT$.

Thus, if we assign the string A$="CORNWALL", the command:

PRINT RIGHT$(A$,4)

would print out the word WALL.

To achieve the same result on the Atari, we need to:

PRINT A$(LEN(A$)-3,LEN(A$))

A complicated approach, but luckily we don't rely on RIGHT$ commands (or their equivalents) in this series of adventure games.

STR$ and VAL

Two functions which are essentially the inverse of each other, and both of which are concerned with string and numeric manipulation.

Take a number A, equal to (say) 12.123.

The command:

PRINT STR$(A)

will print out the string 12.123, although the number A has remained the same.

This command is more useful when assigning variables, so the following program shows this in action:

```
5 DIMA$(9)
10 A=24.232425
20 A$=STR$(A)
30 PRINT A$
40 PRINT LEN(A$)
50 PRINT A$(1,2)
60 PRINT A$(4,LEN(A$))
```

When run, this program will print out the following:

```
24.232425
9
24
232425
```

So you can see, by finding the position of the decimal point, we can split a number up into its two components.

How do we do this?

Well, one way would be to use the inverse function, VAL.

VAL takes a string, and converts it into a number. Thus, if the string A$ was equal to "10", the command:

PRINT VAL(A$)

would print out the number 10.

If A$="12.123", VAL(A$) would also equal 12.123, but of course th time it would be in numerical format.

VAL comes to a halt when it comes across something that is not number.

Thus, if A$="88A88B", VAL(A$) would return just 88.

We can also print out straightforward variables. That is, in the follo ing program, we are defining the variable A to be equal to the VAL of various strings :

```
10 A=VAL("23.23")
20 PRINTA
30 A=VAL("A")
40 PRINTA
50 A=VAL("-100.9")
60 PRINTA
```

When run, the results on the screen would be:

23.23
ERROR- 18 at line 30

This is because we cannot take the value of a variable such as A. T eliminate this we need to use a command something like:

A=VAL(STR$(A))

To split a number up into its component parts, we must find the decim point by turning the number into a string, taking each number at time until we find the decimal point, and so on. Thus on most com puters we would have something like:

```
10 A=345.678
15 B=B+1
20 A$=MID$(STR$(A),B,1)
30 IF VAL(A$)=0THEN100:REM THE DECIMAL POINT
40 B$=B$+A$:GOTO15:REM KEEP ADDING NUMBERS UNTIL
WE REACH DECIMAL POINT
100 C$=MID$(STR$(A),B+1)
110 PRINT B$,C$
```


Just to explain a little:

Line 10 : define the number
Line 15 : increment our counter
Line 20 : turn A into a string, and take one character at a time
Line 30 : is that character a decimal point (VAL(A$) equal to zero)?
if yes go to line 100
otherwise, add the number to our string B$
(line 40)
Line 100 : C$ is made equal to the string equivalent of the number, starting at the character after the decimal point.
Line 110: print out the numbers

The resulting display would read:

345 678

As ever, the Atari forces us to adopt a different approach, since it treats a statement like B$=".":?VAL(B$) as an error (error 18, an invalid string character), and so the program becomes:

```
5 DIMA$(10),B$(10),C$(10),D$(10)
10 A=345.678
12 A$=STR$(A)
15 FOR I=1 TO LEN(A$)
20 B$=A$(I,I)
25 IF B$="." THEN100
30 NEXT I
100 C$=A$(1,I-1):D$=A$(I+1,LEN(A$))
110 ? C$,D$
```

CHR$ and ASC

Another two analogous functions, again concerned with string handling, but ASC in particular assumes great importance when talking about communicating from one microcomputer to another.

ASC is short for ASCII, the American Standard Code for the Interchange of Information, although even Atari are kind enough about their own adherence to this standard to refer to it as Atascii. Yes indeed,

it is another departure from the 'standard', and owes little to what was originally laid down.

Still, to print characters on the screen the following syntax is used:

PRINT ASC("A")

which would return a value of 65, or:

PRINT ASC(A$)

which would return the Ascii value of the first character contained in the string A$.

CHR$ is the opposite of this.

For instance, the command:

? CHR$(65)

prints a letter 'A' on the screen. Some CHR$ commands have special functions (e.g. PRINT CHR$(125) clears the screen), and your Atari manual has details for all of these.

Both of these commands can again be used to define other variables. For example:

A=ASC("A")

will put the value of 65 into the variable A, and

A$=CHR$(155)

will put the character string 155 (in fact, a carriage return) into the string A$.

FOR ... NEXT

Where would we be without FOR ... NEXT loops?

Although we've been instructing the computer to do the same thing a number of times over, by use of a simple incrementing variable, the 'loop' approach is far better, and far easier to operate.

For instance:

```
10 PRINT CHR$(125)
20 FOR I = 1 TO 100
30 PRINT I
40 NEXT I
```

This will just print out the numbers from 1 to 100 in rapid succession, but illustrates the point.

Line 20 is the start of our loop, and tells the computer that we want to do something 100 times. In fact, we want to print out the numbers from 1 to 100, and as the value of I increases, it is printed out in line 30. Line 40 then tells the computer NEXT I, i.e. there's more to come, and the program branches back to line 20.

It keeps on doing this until I has reached the value of 100, at which point it stops and our short program ceases execution.

Actually, I reaches the value of 101. Why? Well, when it has the value of 100, it prints it out as in line 30, sees the NEXT I statement in line 40, and increases the value of I to 101. However, on branching back the computer finds that the limit of the loop is when I is equal to 100, so it stops!

The syntax in line 40 has to be:

```
40 NEXT I
```

as we can have more than one loop active at a time. Like this:

```
10 PRINT CHR$(125)
20 FOR I = 1 TO 20
30 FOR J = 1 TO 3
40 PRINT J,I
50 NEXT J
60 NEXT I
```

The first time around, I is set to 1, and J counts through from 1 to 3. Thus the display goes something like:

```
1       1
2       1
3       1
```

Then J has finished, so we go on to line 60, where I is incremented again, so it's back through the loop once more, for:

```
1       2
2       2
3       2
```

and so on, until we finally reach:

```
1       20
2       20
3       20
```

at which point everything stops.

Lines 50 and 60 cannot be abbreviated to the rather more straightforward:

```
50 NEXT J,I
```

As Atari Basic doesn't allow this. Just make sure you keep everything in the right order, and don't have more than 26 loops in action at the same time, otherwise the computer will blow its stack (computing joke).

Loops can be made to count in steps as well, for instance:

```
20 FOR I=1 TO 100 STEP2
30 PRINT I
40 NEXT I
```

when run, will print out the numbers 2,4,6, ... 100. We can also go backwards:

```
20 FOR I=100 TO1 STEP-2
30 PRINT I
40 NEXT I
```

when run, will print out the numbers 100,98,96 .... 2.

For an interesting application, using only commands we've seen so far, can you work out what this program is doing (type it in and see, if you can't!)?


```
10 DIM A$(10),B$(10)
15 A$="ABCDEFG"
20 INPUT B$
30 FOR B=1TO LEN(A$)
40 IF B$=A$(B,B) THEN ? B$;:GOTO 20
50 NEXT B: GOTO 20
```

GOTO somewhere

We've already encountered this one. Basically it sends command of a program to somewhere else within the program, or back to the same line as in the previous example in line 50.

The syntax used is GOTO xxx, where xxx is an existing line number.

If it isn't, you'll get an error message to the effect that the line doesn't exist being thrown in your face!

As a short example:

```
10 PRINT CHR$(125)
20 PRINT "HELLO!"
30 GOTO 20
```

When run, this just prints up hundreds of HELLO!s, until you hit the break key.

Changing line 30 to read GOTO 10, produces a slightly flickering display.

One can also use GOTO xxx + A, where the line to go to depends on the value of A. Thus, if A were equal to 10, that statement would send program execution off to line xxx + 10.

GOSUB and RETURNing

Subroutines have been met before, as small, or maybe even large, segments of programs that have to be repeated many times.

Performing the same function over and over again is a repetitive task,

and having to type the code in each time you wanted it actioned would take a lot of time, and a lot of memory.

Thus subroutines were born, and the command used to send program control to them is GOSUB xxx (or GOSUB xxx + A, or indeed GOSUB any numerical expression), where xxx is the line number at the start of the subroutine.

Once actioned, the command to send control back to the main program again is RETURN.

Great care must be taken in matching up GOSUBs with RETURNs, otherwise an error will take place sooner rather than later.

As with FOR ... NEXT loops you can have up to 26 subroutines in action at the same time, but no more.

Thus you can jump about from one subroutine to another, and quite often it is necessary to do this, but it isn't really very good programming practice.

A few examples:

```
10 PRINT CHR$(125)
20 A=5:B=10
30 GOSUB 100
40 GOSUB 200
50 GOSUB 100
60 PRINT A,B
70 END: REM IMPORTANT, OTHERWISE PROGRAM FALLS
THROUGH!
100 A=A*A
110 A=A+5
120 RETURN
200 B=B-1
210 RETURN
```

When run, the first subroutine is encountered twice, the second once only, and the resultant printout is:

```
90        59
```

Of course, one can get a lot more complicated than this!


```
10 PRINT CHR$(125)
20 A=1:B=2:C=3
30 GOSUB100
40 GOSUB200
50 GOSUB300
60 PRINTA,B,C
70 END
100 A=A+B+C
110 GOSUB200
120 RETURN
200 GOSUB300
210 A=A+B+C
220 GOSUB300
230 RETURN
300 A=A+1
310 RETURN
```

What value will A have when this program is run? Try it and see!

What's GOing ON

Quite often within a program, the subroutine or line number you want to go to will depend on the value of a particular variable.

This could be achieved in the following way:

```
10 IF A = 2 THEN 100
20 IF A = 3 THEN 200
30 IF A = 4 THEN 300
40 IF A = 5 THEN 400
50 IF A = 6 THEN 500
60 etc.
```

Although this works, it could hardly be described as an elegant way of programming.

In its place we can use the ON ... GOTO command, and the similar ON ... GOSUB. As both work in the same way we'll take the former as an example, although with the latter you do have to take care over matching up RETURNs with GOSUBs.

```
10 ON A GOTO 100,200,300,400,500
```

Here, if A has the value 1, the program continues execution at line 100 onwards, a value of 2 and it goes to line 200, and so on up to a value of 5, when it goes to line 500.

A can be varied, to make it match our earlier IF ... THEN example as follows:

```
10 ON A-1 GOTO 100,200,300,400,500
```

Thus we now have an exact match of the original program, but in four fewer lines! Now, if A equals 2 program execution continues at line 100, and so on.

Just one example of this command in use could be something like this, which is an interesting use of string handling:

```
5 DIMK$(5),A$(1)
10 K$="ABCDE"
20 PRINT"ACTIVITY 'A' : PRESS A
30 PRINT"ACTIVITY 'B' : PRESS B
40 PRINT"ACTIVITY 'C' : PRESS C
50 PRINT"ACTIVITY 'D' : PRESS D
60 PRINT"ACTIVITY 'E' : PRESS E
70 INPUTA$
80 FORI=1TOLEN(K$)
90 IFA$=K$(I,I)THEN1000
100 NEXT I
110 GOTO70
1000 ON (ASC(A$)-64) GOTO 1100,1200,1300,1400,1500
1100 rest of program.
```

RaNDom INTegers

Like most of the home computers currently available, the Atari is not without a random number generator.

Alas, like most of them it isn't particularly random, and so a few operations have to be done before we can begin setting up 'genuinely' random numbers.

The syntax to be observed is RND (A), which will give a number in the range 0 to A-1.

The INT command comes in useful here, as elsewhere. It chops off

the numbers after the decimal point, so INT(2.24) becomes 2, as does INT (2.89).

INT of a negative number returns the next lower number. Thus, INT (-2.24) becomes -3.

So, to generate an integer random number, we could use:

```
10 PRINT INT(RND(0.5))
```

However, this will not be very satisfactory for generating future numbers, since RND always returns a number between 0 and 1. So, we need to scale things up a little:

```
10 PRINT INT(RND(.5)*10+1)
```

which will produce a number in the range 1 to 10.

To generate numbers within a given range, where X is the top limit and Y the lower limit, we must use the formula:

```
10 PRINT INT((X-Y+1)*RND(.5)+Y)
```

This is used in our adventures for producing random events e.g. the appearance of a gargoyle, or the success or failure of throwing a knife.

A New DIMension

We've already seen how numbers and strings can be stored as variables like A, A$, and so on. However, this gets a mite restrictive after a while, and we need to resort to other things. After all, there are only so many letters in the alphabet.

Let's say that we're generating ten random numbers, and we want to store them all as variables.

We could have a very lengthy program to do this:

```
10 A=INT(RND(.5)*10+1)
20 B= ..... etc.
```

but this is extremely space consuming, and there are better ways.

This is where arrays, otherwise called subscripted variables, come in.

The syntax for referring to these is A(0), A(1), etc., up to a limit of A(whatever), and these subscripted variables could be assigned numbers something like this:

```
10 FORI=0TO10
20 A(I)=INT(RND(.5)*10+1)
30 NEXT I
```

Where now we have the eleven different numbers stored in A(0), A(1) etc. up to A(11).

These numbers can then be selected at will. For example, PRINT (A(4)) will print the fifth number, or element, in our array A: remember that the first element is referenced as number 0.

To prove it, we could print them all out by adding to our program:

```
40 FORI=0TO10
50 PRINTA(I)
60 NEXT I
```

The numbers in an array can be assigned to other variables (e.g. A=A(3)), or even calculated dynamically by using another variable (e.g.PRINT A(B*2)).

However, more often than not we'll be wanting to use a lot more than eleven elements in an array, and this is where the DIM statement comes in.

The syntax for this is DIM A(199), or whatever, which sets aside a certain amount of room in the computer's memory for storing all the numbers that you might be wanting to save. Whether you use them all or not, that memory is reserved, so use arrays selectively.

Arrays are not limited to one dimension either. You can dimension something as A(7,7) if you like, for instance in a chess game, where you have a board 8 squares by 8.

The elements in that array are referred to as A(1,5), A(6,3), and so on. It is helpful to think of these values as being stored in rows and columns, where the first number refers to the row and the second to the column. Thus A(5,7) is the seventh column of the fifth row. Thinking of it all as boxes of numbers, or strings, stored in rows and columns will always help when you want to reference a particular one within a program.

We'll be using arrays extensively in all our adventures, so it's useful to learn how they operate!

Also, remember that all strings have to be dimensioned, and that we cannot have string arrays in Atari Basic. Thus, if you want to define A$="I'm a string", somewhere earlier in your program you'll need a DIM A$(12). If you'd only DIMmed A$(4), and then said A$="I'm a string", then A$ would in fact only be equal to "I'm ".

Getting Started

Now that we've learnt most of what we'll need to know about strings, data and dimensioning arrays, it's about time we started looking at the results of using this in an actual program.

Our example, as always, will be the Underground Adventure listing, so now we'll start explaining some of the variables that are used in this game, so that we can get an understanding of how the various parts of the program operate.

Lines 2 to 12 define one set of variables, which relate to the gate being open (GF), and door being open (DF), line 2 goes respectively to the subroutines that print up the introduction and read in all our data, as well as defining the variable CP, which is the Character Position, and relates to the room number that you happen to be in at the time.

The next set of 5 variables just contain messages that we'll use later on in the program. The dimension statements in lines 5 to 7 relate to the strings that follow, and the strings that are read in in the routines that check your entry when playing the game.

```
2 GOSUB 10000:GF=1:DF=0:CP=1:GOSUB 2000
5 DIM A$(1),CM$(27)
6 DIM NO$(10),V1$(10),VB$(5),VT$(5),N1$(10),NT$(5)
7 DIM PD$(21),IM$(22),GF$(21),D$(25),DF$(17),PA$(1
0),PR$(10)
8 PD$="It is now pitch dark."
9 IM$="You can't go that way."
10 GF$="The gate is now shut."
11 D$="Going down ... "
12 DF$="The door is shut."
```

Moving Around

Line 200 sends us off to the routine that checks for character movement:

```
200 GOSUB 5000
```

but before we look at that, we'll jump down to line 2000 and define a few more variables:

```
2000 NV=38:NN=53:LO=53:DIM P(100,3),OB$(25),DD$(5)
,OB(LO):GOTO 2110
```

This controls all our data reading which takes place in lines 2001 to 2220. These are reproduced in chapter 6, but the variables are set as follows:

NV = the number of verbs we're going to use, which in this case is equal to 38.

NN = the number of nouns we're going to use, which in this case is 53.

LO = the number of nouns again, but is used to control the LOcation of every object in the game, there being as many objects as there are nouns.

DIM P(100,3) = dimension the variable P to be equal to the number of rooms, with four sub-elements to each level of P. These are used to determine the direction one can take from within a room, and indicate NORTH, SOUTH, EAST and WEST respectively. Thus P(I,2) refers to the direction EAST from room I, P(I,1) would indicate the direction SOUTH from room I, and so on.

DIM OB$(25) = dimension the variable OB$ to be equal to the length of the maximum object description string used in the game.

DIM OB(LO) = dimension the variable OB to be equal to the number of nouns. OB then contains the position of each object in the game, by referring to its room number. Thus OB(I) refers to the Ith object, and if set equal to J puts the Ith object in the Jth room.

DIM DD$(5) = dimension the variable DD$ to be equal to the length of the maximum direction description (that is, NORTH or SOUTH).

Finally, we go to line 2110 to begin reading in the data that forms the map of the game. Be very careful if you're typing all this in, as one mistake here will cause errors to occur in lines that, on the surface, contain no errors at all. For instance, a line that is attempting to read in some numerical data might encounter some string data instead, and this will generate an error.

Now let's look at the actual room movement routine, contained in lines 5000 to 5025.

Room Movement Routine

This routine is used to handle all room movement in the game, so we'll take a close look at it.

```
5000 ? CHR$(125)
5001 IF OB(46)<>-1 AND (CP>4 AND CP<100) THEN ? PD
$:PD=1:RETURN
5002 GOSUB (2000+CP):PD=0
5003 IF CP=42 AND TB=1 AND P(42,1)=0 THEN 6054
5004 ? :? "You can see:"
5006 RESTORE 2200:FOR I=1 TO LO:READ OB$:IF OB(I)=
CP THEN ? OB$:OB=1
5008 NEXT I:IF OB=0 THEN ? "Nothing special."
5009 OB=0:IF CP=3 AND GF=0 THEN ? :? GF$
5010 FL=0
5012 ? :? "You can go:":RESTORE 2218:FOR I=0 TO 3:
READ DD$:IF P(CP,I)<>0 THEN ? DD$;" ";:FL=1
5014 NEXT I:IF FL=0 THEN ? "Nowhere."
5015 ? :IF NP=1 THEN 6000
5016 IF (CP>20 AND CP<88) AND (INT(RND(0.5)*10+1)>
9) THEN NP=1:GOTO 6000
5018 IF CP<>69 THEN RETURN
5020 IF P(69,3)=70 THEN RETURN
5022 IF OB(15)<>-1 THEN ? "You can't pass yet.":RE
TURN
5024 ? "The shimmering curtain washes away the  mi
st."
5025 P(69,3)=70:OB(15)=0:ZZ=ZZ-1:OB(20)=0:RETURN
```

Explanation

We'll take this line by line, so:

Line 5000 simply clears the screen by printing a character string 125.

Line 5001 checks to see if you're holding a blazing torch (OB(46)). If the variable is set to -1 it means that you're carrying it. If it's not equal to -1, the line carries on to see if you're in a room lying between room numbers 5 and 99. If you are, it then prints up the variable PD$ as defined in line 8 and returns to the WHAT NOW prompt, having set the darkness variable PD equal to 1. Any attempt to move now without lighting the torch will make you fall into a pit and plummet to your doom.

Line 5002 sends program execution to the line 2000 plus CP, CP being the number of the room that you happen to be in at the time. Whatever line you go to just prints up a description of the room, and then returns program execution to this point. Then the darkness variable PD is set to 0, since if we've moved, we can't be in darkness.

Line 5003 checks the 'bear following' variable TB. If the bear is following you, and you're in room 42 (which holds a fierce panther to begin with), and there is no path south from room 42, the program transfers execution to line 6054, which prints up what happens when the bear meets the panther!

Line 5004 is the start of the 'you can see' routine, which goes on to:

Line 5006, which checks to see if the location of any object, OB, is equal to the current room number CP. If it is, then it tells you that you can see it, but if nothing's there it just prints up nothing. The RESTORE 2200 statement sets the data pointer to line 2200, which is the start of the object descriptions, and every time the presence of an object is checked for, another object description is read in. Thus we avoid telling the player that, for example, a blazing torch is in the room, when the program knows that what is really there is a box of matches.

Line 5008 finishes off the loop, and if nothing of interest is found to be in the room, prints up the 'Nothing special' message.

Line 5009 checks to see if you're in room 3, and if the flag for the state of the gate (open or closed) is set or not (1 or 0), and if it is set prints up the variable GF$, as defined in line 25.

Line 5010 just clears the variable FL, used in the next routine:

Lines 5012 to 5014 go through the four possible directions from each room, and check to see whether you can go in any of them, by seeing if the relevant part of the variable P is set to 0, in which case you can't, or something else, in which case you can. It then prints up the right part of the variable DD$, which is set to equal the words NORTH, SOUTH, EAST and WEST by restoring the data pointer to point to line 2218, which contains as data statements the words north, south, east and west.

Line 5015 checks to see if there's a gargoyle chucking a knife at you, and if there is transfers program execution to line 6000, which we'll come to later.

Line 5016 goes through a random number generation, and if that number is greater than 9 (on a scale of 1 to 10), and if you're in a room number greater than 20 but less than 88, it sets the gargoyle present flag NP and goes off to line 6000.

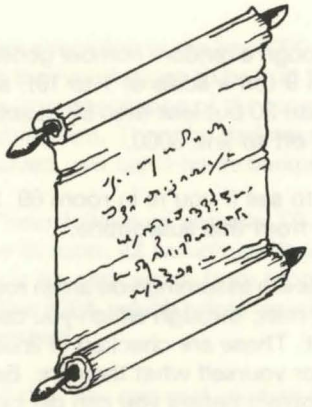
Line 5018 checks to see if you're in room 69. If you're not, program execution returns from this subroutine.

Lines 5020 onwards are assuming you are in room 69, which is initially guarded by a hazy mist, through which you cannot pass until various conditions are met. These are checked in lines 5020 to 5025, and I'll let you work out for yourself what they are. Basically you have to be carrying a certain object before you can get past, and if you are then obviously the hazard doesn't exist any more, and we have to change the relevant parts of the variable P(69) to allow us to move safely through here in future, which is done in line 5025. So you can see the checks that have to be made before we can allow our explorer to move through certain areas.

It would be an easy matter to alter this routine to suit your own adventure requirements, just by changing the conditions that have to be met, and checking for the right room numbers and the right flags being set.

As I said, you'll find all the data in chapter 6.

Now we've seen how one routine works. Let's sit down and write an adventure!



4

Writing Your Own Adventures

Let's Get Started

We've seen one of the major routines in the game now, that of handling the movement of the character within the adventure, once we've established from other routines whether or not the character can in fact go in that direction.

That is achieved using the verb GO, which we'll come to along with all the other verbs in chapter 6.

All the data that is necessary for this game, together with a thorough explanation of how it all works, what it all means and how it's all stored in the program, will also be found in chapter 6.

Meanwhile, there's an awful lot of additional coding which doesn't come into either of those sections, and the purpose of this chapter is to present you with the rest of it, including standard routines for the inputting of data, checking on the validity of a move, checking whether the words you've typed in make sense, and one or two other routines which are especially for this game (we couldn't just give you all of the listing bar a couple of lines!), but which could nonetheless be adapted for use in your own games.

You'll know the sort of occasions when it is necessary to include these special routines, what they're doing (and equally important, how they're doing what they're doing), and so you will be able to use variations on them in your own games.

So, between this chapter and chapter 6 you'll get the complete listing

for Underground Adventure, and perhaps by presenting it in small chunks like this you'll feel more inclined to type it all in!

If not, you could always buy the cassette containing the three adventures in this book, configured to run on your computer, and study the listing that way.

Summary So Far

You know what a number of the essential variables in this game are now doing, and can readily adapt them for use in your own games.

The variable CP for instance, which is used to keep track of the room number, and is updated as you move from room to room.

The variable NP, to detect whether or not a living gargoyle has emerged from the rocks and is about to engage you in mortal combat.

The variable PD to check for darkness, and the carrying of the blazing torch.

These, and the others, are the backbone of the game, and without them this adventure could not function. Without similar variables in your own games it would be equally impossible to play and/or write them.

Variables like these are there to make life easier for you. Use them in your own games, and the actual writing of a complete adventure will soon become relatively easy.

However, there's a lot to learn yet, like the drawing of maps, the placing of objects, the positioning of any hazards en route, and everything that goes up to make the total game.

In the next section we'll start again from scratch, and assume that you've sat down with a blank sheet of paper, and want to start writing an adventure game.

So let's get going!

The First Steps

Possibly the most difficult step of all is outlining the story that you're

going to have as the backbone of the adventure.

In effect it will have to be a miniature novel, involving (relatively) realistic concepts, although an ingredient of most adventure worlds is that little touch of magic that sets them aside from the real world.

The plot, just as in a good novel, must flow smoothly from one stage to the next, with no totally unexpected, inexplicable events. One adventure I know suddenly has a sword that you've been happily carrying along turn into a snake in your hands, which then bites you and kills you off.

This is totally inexcusable, and shouldn't find a home in any real adventure. The impossible happens quite often in these games, but at least there should be a warning that it's going to happen, and it should not be sufficient to kill off the character.

So if we're going to have magic, let's keep it on a fairly reasonable level, and stick to iron staffs being waved and causing a bridge to appear over the chasm.

Events that kill off the hero, like crossing a rickety bridge with a heavy bear in tow, should generally be as expected as possible, and only be the fault of the adventurer. In real life, would you expect a rickety bridge to support the weight of a heavy, lumbering bear?

In Underground Adventure, dynamite has to be employed in one instance before you can progress. It is reasonable to assume that lighting the dynamite whilst you're still holding it will not do you any good, and so it should be placed on the ground first of all.

On the other hand, some of the elements in this game, and others, are randomised to give the game some semblance of reality. Not that you'd often bump into a living gargoyle carved out of the rock, who then engages you in a duel to the death every time you meet him, but should such an event take place it is reasonable to assume that the outcome of the fight will not always be the same.

Thus you will sometimes get killed (though not very often, otherwise the game would get very tedious), and sometimes your throws will miss the gargoyle, but again you should conquer him (her?!) most of the time and live to carry on the game.

So anything that happens in the game must have a remote base in reality, and the inexplicable shouldn't really happen without at least

being safe to the player.

Getting the Idea

As we've said, this is possibly the most difficult part of all. Many adventures have now been written, and coming up with an original scenario each time is getting gradually harder and harder. Some possible ideas are presented in chapter 5, where we've gone through a number of adventure scenarios, and described them in some considerable detail.

However, there is of course no constraint on you to use them at all, so your own ideas will have to come from somewhere.

One tried and trusted idea is by dipping into a few books such as *Lord of the Rings*, in which there are a multitude of possible plots which could be turned into very reasonable games. However, as in all implementations of this sort one has to be very careful about the laws of copyright, as we've seen with the Hitch Hiker's Guide to the Galaxy game, so you'll probably have to change a lot of names to protect the innocent, i.e. you!

The traditional thud and blunder adventure, steeped in Gothic names and ancient runes, has been done by many authors, although obviously the scope here is vast for doing variations on a theme.

One possible answer might be to read a few science fiction novels (bearing in mind the author's copyright), such as the works of Michael Moorcock, and obtain a few ideas from there.

To the beginner though it must seem that just about every possible idea has been tried before, including exploring ancient tombs and crypts, jungle adventures that pit you against various natives and native problems, cowboy adventures, outer space adventures, underwater adventures, and the like, and that it would be impossible to come up with a new and original plot-line for your story.

But bear in mind that there have been many more novels written than there have been computer adventures, and people still keep managing to come up with original themes for those, so the ideas are always there: it's just a question of thinking them up.

Visitors from outer space, detective adventures, psychological adventures, biblical adventures, are all relatively new areas, and perhaps

combining one of these new ideas with the character choosing role discussed earlier could pave the way for a whole new set of computer games.

The work is up to you though, and your plot, whatever it consists of, must ring true throughout, and keep the player of the game constantly entertained, forever pitting him against new challenges, new tasks, and keeping the interest by finding out just that little bit extra with each game.

The Hazards

Now there's a television program! But no, nothing to do with car driving American lunatics in an otherwise sleepy mid-western village, one of the most important parts of any adventure game will be the constant search for new problems to set the player, new tasks that have to be accomplished before you can proceed further, and making those hazards solvable, but (preferably) as difficult as possible.

The number of problems set will always vary from game to game, and should to some extent depend on the number of rooms in the game. Perhaps on a 1 to 6 ratio, with a new task to be solved every half dozen rooms or so ?

Some games favour a constant source of worry, and indeed Underground Adventure does the same, with the living gargoyle coming up every now and again, along with a random chance that, as well as fighting with you, he might just nip in and steal a few useful items that you happen to be carrying and hide them in the maze.

As a helping hand, here's a list of the hazards presented in Underground Adventure, and the rooms in which they are first found:

A vast chasm that is too wide to jump: room 15
A massive tree that blocks your path: room 21
A deep drop that is to steep to climb down: room 35
A blocked wall that prevents you from going further: room 4
A golden bear that will not let you pass: room 27
A fierce black panther that stands in your way: room 42
Another deep chasm amongst the rocks: room 10
A steep incline that is to steep to climb up: room 45
A shimmering curtain of light that dazzles you: room 93
An old mining track that is blocked up: room 79
A hazy wall of mist that is too thick to pass through: room 69

The denizen of the caves, who will not let you through: room 50
A giant spider, out to eat you: room 84
A giant fly, out to kill you: room 74
An old door that blocks your path: room 60
A narrow crack, which you can't squeeze through: room 53

There are 100 rooms in Underground Adventure, so we fit nicely into our 1 in 6 ratio, with the above 16 problems to solve. We'll tell you some solutions along the way, but not all of them!

Constant Problems

As well as all of the above, there are a number of constant problems that keep recurring, like the gargoyles, and any reasonable adventure has the same kind of mixture. A good solid set of problems which give the player plenty to chew over, along with a reasonable set of constant events that can also give cause for worry.

However, whatever the kind of problem, be it in a set place or occuring at random, one golden rule of programming this type of game remains the same: if the player solves the problem, make sure the program checks for this and adjusts its variables accordingly.

There is nothing worse for a player than, having spent hours achieving one goal, to throw away the relevant object which has enabled him to do this (or perhaps have it taken away by the program once it has fulfilled its duty), and then to see a bug in the program causing the problem to re-appear!

In other words, don't make your adventures impossible, which is always a problem when you're manipulating a lot of objects. Just placing one of them in the wrong room could cause the program to become unsolvable: a cardinal sin.

One of the more common constant problems is that of a torch. If you're deep underground it's fairly safe to assume that you won't be able to see very much, and so a torch becomes vital.

To light the torch you will also need some matches, and these must also be hidden in the game somewhere.

Finding the torch and lighting it is usually no problem, but keeping it lit often is. A sudden gust of wind perhaps (which could easily be done in the earlier movement routine by checking for, say, room 52

or whatever, and whenever the player walks through there the torch gets blown out), or a swim through some water would do the trick. If you go through water, you would also get the matches wet, so how do you light the torch again?

A torch carries with it another problem. There is usually a limit on how much you can carry at a time, and certain objects will always have to be with you, like torches, axes, and so on, and so the problem becomes what do you carry at the same time.

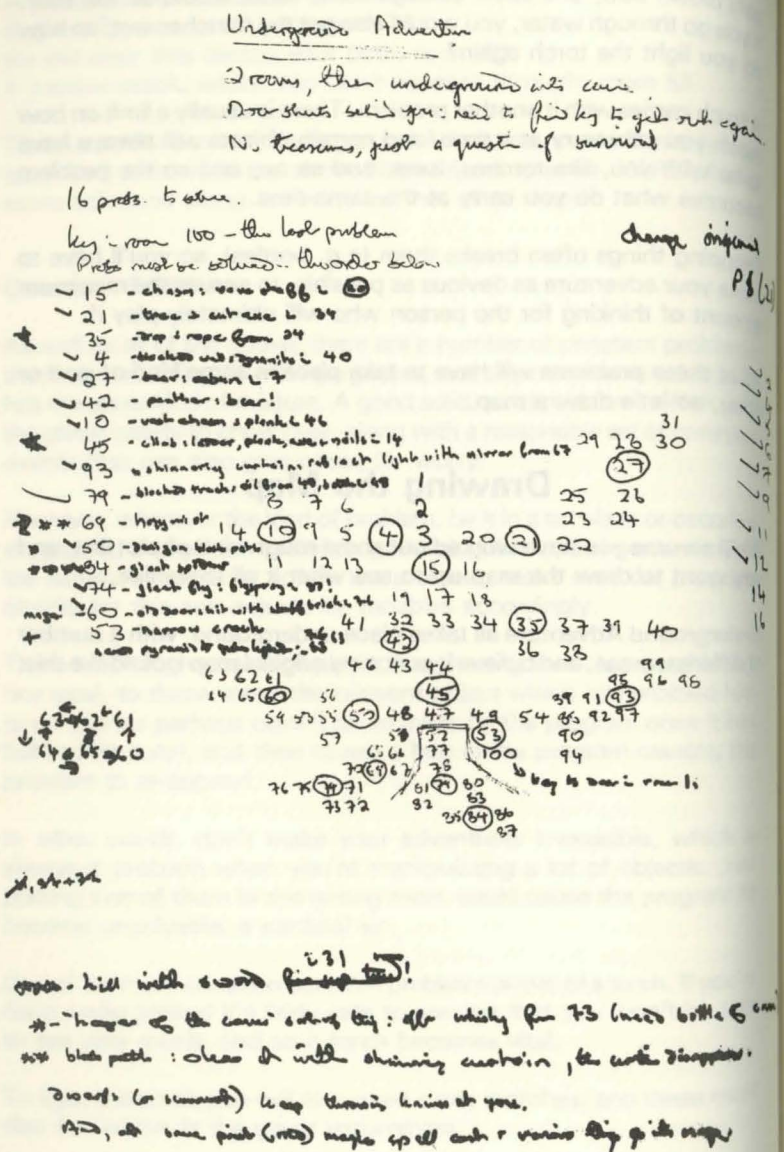
Dropping things often breaks them (e.g. bottles), so you'll have to make your adventure as devious as possible, to ensure the maximum amount of thinking for the person who will ultimately play it.

All of these problems will have to take place in some kind of land or other, so let's draw a map.

Drawing the Map

We'll assume you have worked out some rough kind of plot line, and you want to draw the map up to see what it all looks like.

Underground Adventure all takes place underground, with a number of different areas, and believe it or not my original map looked like this:



Refining the Map

Well, that was certainly nothing to write home about! However, it worked, because having drawn up all the room numbers I then had a much better idea of fitting the adventure together as a whole, and could commence setting up the problems for the player.

The first thing I did was to label 16 rooms (ringed, in the diagram), and decide that this was where the problems would occur. Then, I had to write down what each problem would consist of, and those are the notes at the left of the sheet.

The brief scrawl at the top was an indication of the general outline of the whole thing. There was to be no finding of treasures, it would all be a question of survival, with the all important mission being to find the key to enable you to open the door that had slammed shut, and get out again.

The notes at the bottom where there as guidelines for one or two of the problems, and from that map the whole game was written.

Well, that's not quite true!

A number of changes were made to the original plan, including the location of one or two of the objects in the adventure area, and before I set fingers to keyboard there were a number of other notes to be made first.

We'll see what they were in the next section.

But for now, you'll have drawn your map, however rough it may be, you've got some idea of the general plot for the whole story, and you know (again roughly) where all the hazards are going to present themselves.

You've got a fairly good idea of everything that will happen to our intrepid explorer, and in chapter 6 you'll see one way of turning these ideas into the necessary data statements that form the fabric of the entire game.

But we're concerned with the programming side of it, rather than the sheer slog of getting all the data statements typed in, so let's start making the transition to the computer.

Moving from Paper to Computer

One of the first steps is to draw a much more sensible looking map, as we've shown over the page for one of the other adventures in this book, the Castle Adventure.

This should be big enough to enable you to list everything you want to in each room, including any objects that are to be found in them, and any hazards that may be expecienced in that room.

Having done that, you'll obviously want to know what all of those objects are! So the next step is to look at the list of hazards as you originally drew them up, and decide what the solution would be to each hazard, bearing in mind that you can only move on to the next part of the adventure after you've solved the problem. In other words, don't put the solution further into the game than the problem!

A list of solutions will give you a healthy list of objects, and these will then form the basis of the list that we'll type into our program later.

With the program set up as it is, although obviously you could modify it if you want to, the routine that checks your data entry only looks at the first three letters of each word. Thus if you had a TRACK and a TRAM in your adventure the program listing would interpret them to be the same object, and you would get some very strange displays being shown up on the screen!

So, if you're going to follow the methods outlined in this book, it helps to give all the objects individual names. As we'll see later, there are enough problems coping with EMPTY BOTTLE, BOTTLE OF OIL and BOTTLE OF WHISKY in Underground Adventure as it is, so we don't want to encourage more of them!

This list of objects will have to be extended beyond a simple list of those generated by the problems and their solutions. We haven't mentioned lamps, or anything like that, so you'll have to have words for LAMP.

What happens if you drop a bottle? If you're going to have it break, you'll also need to have an object something like A PILE OF BROKEN GLASS.

These, and other problems will all have to be thought of before we

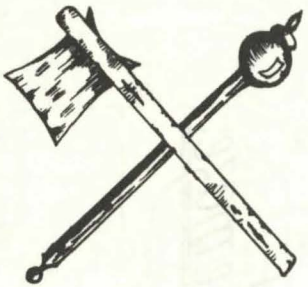
start typing anything in, but inevitably we'll have to add objects to our list as we go along developing the program, but in Basic that is no great difficulty.

Underground Adventure originally started out life with about 48 objects, but ended up having 53, due to circumstances arising during testing of the program that I just hadn't envisaged beforehand. It's nice to track everything first before you start though.

A New and Better Map

This is the final map for Castle Adventure, as re-drawn from an initial scrawl on a tiny sheet of paper.

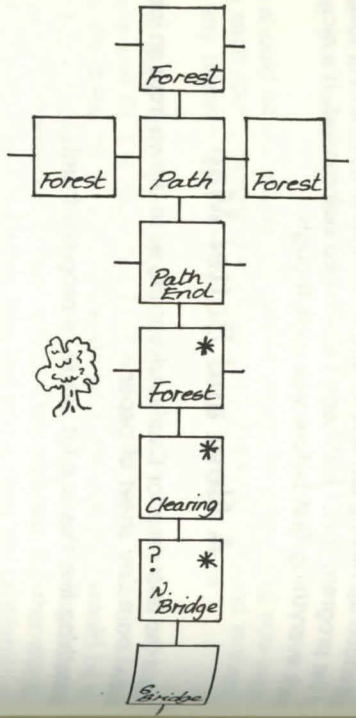
Something like this is a lot easier to program from!

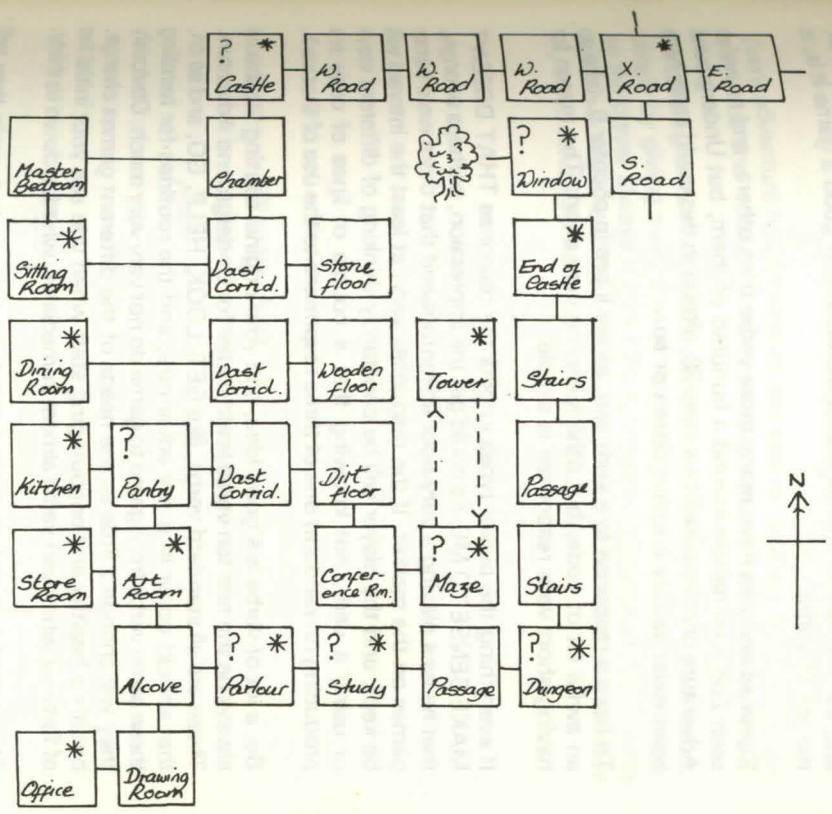


Castlemaze.

?— problem to solve

*— useful objects / treasures





And on to Verbs

As well as our list of nouns, the other great list in any adventure game, and the list that to a large extent dictates how good a game it is, is the list of verbs.

Some adventures have many more verbs than others, and as we've seen Zork can handle around a hundred of them, but Underground Adventure confines itself to a mere 38, although this could easily have been extended by another dozen or so.

To have a response to a verb can, as we'll see in chapter 6, take up an awful lot of code, but others can be very short. The reason for having short verb responses is simple.

If everything the player types in gets the response THAT DOESN'T MAKE SENSE TO ME, he could get the impression, perhaps wrongly, that he was playing a very poor adventure and that there were better games on the market. If the responses vary, at least the interest will be kept, and the player will be constantly thinking of different ways of using a verb, not knowing that a couple of lines of code are producing (at random) one of three responses to the use of that verb.

So, a lot of verbs is a good idea, and your original starting list should always be the first ten verbs listed earlier for Underground Adventure. These are all standard verbs, like GET, LOOK, HELP, GO, and so on, that should occur in every adventure, and the routines for handling these same verbs from game to game do not vary very much. Obviously they will change a little as the needs of the different games change, but it's a healthy and encouraging start when you see your initial list of forty (or whatever) verbs almost immediately whittled down to thirty.

The rest of the verbs are very much up to you, but again they will to a large extent be dicated by the problems that have to be solved.

There is no point in having a can of fly spray to kill the giant fly if the verb SPRAY is not included in the vocabulary. KILL is too woolly a word, and could produce the wrong response if the spray was not being held.

Additional verbs should also be there, just to encourage diversification of response from the computer, and keep the player's interest. A good idea is to give bizarre ideas on the part of the player equally bizarre

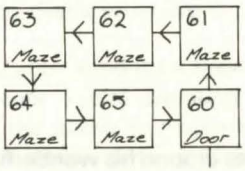
responses from the computer.

It all adds to the humour of playing this type of game.

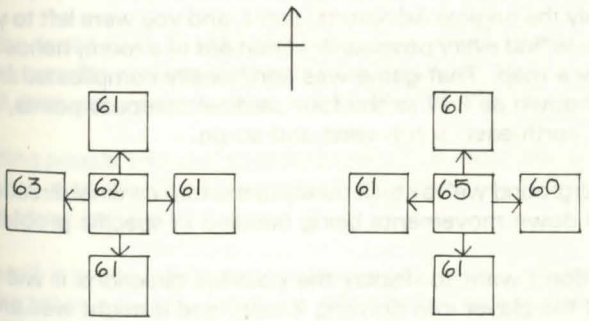
Amazing

Every adventure has a maze of one sort or another, and having got our verbs and nouns, it makes sense to put a maze somewhere.

As the diagram below shows, hard mazes are very easy to construct, simply by giving every one of (say) six rooms the same description, so the player always thinks he's in the same room, and if he makes a move in any one of the three directions you don't want him to move in, why, send him back to the start! Like this :



Construction of a simple maze using a one-way system.
Taking a wrong turning results in the player returning to room 61.



The only way through the maze is to go W→W→S→E→E.

Some General Rules

Although we've been looking at specifics for the last few pages, for the next half dozen pages or so we'll turn our attention to some general rules when writing these games, and concentrate on five of the most important parts of every adventure game:

1) Movement of characters

2) Responses to inputs

3) Screen displays

4) Picking things up & dropping them down

5) Problem solving

Movement

As your character moves around his wonderful adventure world, there are obviously certain rooms he will and will not be able to go into straight away.

Some rooms will be purely east-west or north-south corridors, in which case it would be rather silly to tell your character that he could move north/south and east/west respectively.

You may or may not display which directions he can move in at all. Certainly the original Adventure didn't, and you were left to your own devices to find every possible direction out of a room, hence the need to draw a map. That game was additionally complicated by having up and down as well as the four cardinal compass points, and also having north-east, south-west and so on.

In Underground we've stuck purely to the four cardinal directions, with up and down movements being handled in specific problem areas.

If you don't want to display the possible directions it will certainly prompt the player into drawing a map, and it might well annoy him considerably to be told over and over again 'YOU CAN'T GO THAT WAY', although interest could be sustained by the addition of the little word YET, thus making him think Aha! perhaps I can go along there later.

Personally, I'm in favour of displaying the available choice of directions, as it speeds up the playing process, but if necessary you can just resort to hints like 'A VAGUE TRACK HEADS OFF TO THE SOUTH', and the like.

It's up to you, but whatever style you pick, make sure that you stick to it throughout the game.

Screen Responses

This is obviously the factor that is most important in keeping the interest and attention of the player throughout the game, and if you want to resort to sound, colour and graphics that's up to you.

However, the simple text-only game without any sound has been used throughout this book, so that's what we'll concentrate on here.

In designing and writing your adventure there is an important factor to bear in mind whenever you're planning the responses to the statements typed in by the player in response to the WHAT NOW prompts, and that is that people playing adventures will never, ever type in what you want them to.

You may have a situation where a player comes to a halt in front of a gate that he can't climb over because the top of it is riddled with barbed wire (an escape from Colditz type adventure?), until he gets hold of a set of wire cutters. You have programmed all your responses to GET GATE, GET WIRE, and so on, and are waiting for the player to get the cutters and type CUT WIRE.

What if he types CUT GATE? What happens then? Or what about something typed in in sheer desperation, as people do, like EAT GATE? Does the gate get swallowed up in a display of apparent relish?

Anticipating people's lines of enquiry is one of the most difficult things to allow for, and will take up an awful lot of program code that will probably never be used.

Still, even if it is used only once at least you'll have the satisfaction of knowing that someone out there will consider that the game that he's playing is an extremely robust, well thought-out adventure.

Always try to anticipate the impossible. You'll never manage all of it, of course, and will have to rely on some stock I DON'T UNDERSTAND

type responses, but a few of those mixed up and one picked out at random will keep the interest from flagging.

And never forget the use of the word YET. It will keep a player trying long after the more straightforward 'YOU CAN'T OPEN THE GATE' will.

So the golden rule here must be to keep it interesting, and try to anticipate everything that the player might type in. You won't get them all, but at least you can conjure up some different responses.

Also, a large list of verbs is a great help here: even if the responses are only short and sweet, at least the player will be seeing something different on the screen.

Screen Displays

To a small extent we've covered this one already, but it's worth going over some of the ground again.

The use of graphics has been deplored often enough before now to render any comment here redundant, although you might think the odd display of a sword or amulet every now and again might liven things up a little. But nothing can beat the written word.

Sound is a different question, and the arguments concerning this are almost as legion as those concerning the use of graphics.

My own view is that if you're going to use sound, it must be done extremely well, as the computer is capable of a very complex series of sound outputs. If you're only going to give a little beep every now and again, it's hardly worth the effort of putting it in there in the first place, and you'll soon have people racing for the volume control and a blessed silence.

If done well, it can greatly enhance a game, as people who have played the Temple of Apshai on a Commodore 64 will know: the use of sound is very good here, and the whole atmosphere of moody, omnipresent danger is well presented.

On the other hand, all their programming efforts are wasted if somebody turns the volume down. Be prepared to have sound in your programs if you wish, but don't be disappointed if everyone immediately adopts to play out the game in silence.

The words that are displayed in the screen are obviously dictated by the responses you've allowed for, but an overall attractive layout is to be desired, usually using lower case, since most people seem to prefer that for some reason. Perhaps it's more restful on the eyes as you do battle against a giant troll!

Silly little things can so easily spoil a game in this area - if your room descriptions overlap the edge of the screen so that words are split up, or an inventory list causes some of the objects to be displaced against each other, or even if your output is riddled with spelling errors.

It doesn't take too long to check all of these things, and the results are well worth the trouble. A neat adventure is more likely to be played than a badly spelt, badly laid out one.

The golden rule here? Keep it simple, but keep it tidy.

Picking Things Up and Putting Them Down

Two of the most important words in the adventurer's catalogue are GET and DROP, and in chapter 6 we take a more detailed look at these two words as they apply to the game Underground Adventure. However, a few general words of advice before we get to that chapter.

Obviously, in any game there will be a number of things that you can pick up, and a number that you can't, with the former probably far outweighing the latter. Nevertheless, all possible occurences must be taken into account, and just because you know that the BARRED GATE is too heavy to carry, that won't stop virtually every player who comes along from attempting to pick it up and walk off with it.

Another annoying thing to find in any adventure program is a description that might read something like 'YOUR PROGRESS IS HALTED BY A SOLID WALL OF ROCK', and when you type in GET WALL, the only response is 'I CAN'T SEE ANY WALL HERE', or 'I DON'T KNOW WHAT A WALL IS'.

Look out for that one, for although it can be covered by a blanket response of NO!, that is not very good practice and will certainly not produce an excellent adventure game. Far better to have a response actually geared to the request like 'THE WALL CANNOT BE CARRIED', or something like that.

Some things in a game are only meant to be carried after certain actions

have taken place, in which case you'll need a number of variables to flag the progress of the adventurer, and you'll also have to use the word YET to keep the level of interest there. 'YOU CAN'T CARRY IT YET', will have someone attempting to carry whatever IT is until the cows come home, even if they never can carry it.

When dropping things, a subtle level of difficulty comes into the game. In Underground, after you've made friends with the bear and he's happily trundling around the caves after you, dropping anything will cause him to think that you're throwing things at him, and he'll disappear in a sulk to a random part of the caves, never to be GOT again.

Dropping bottles is usually a good one, since you can have them break on your adventurer, thus rendering them useless for the rest of the game. The original Adventure had as one of its treasures a Ming Vase, but dropping it caused it to smash into delicate little pieces, unless (of course!) you'd taken the precaution of placing a pillow underneath it.

GET and DROP are fun, and don't confuse GET with TAKE. The two words are not the same! For instance, people talk about TAKEing medicine, not GETting it!

Problem Solving

The key to any adventure is how good and how complicated the problems may be in a game, but don't make it too complicated to get started, or your adventurer might give up in disgust and never play an adventure game again.

Encourage people by at least letting them get started, and then pile the problems on, preferably making the first few lean towards the easy side, and have them get harder as the game gradually progresses.

The Scott Adams games are particularly good here, as it is always possible to get somewhere at a first sitting, even if that somewhere isn't very far, and you can gradually improve your progress just about every time you play the game.

Problems usually have to be solved in a set order too, in that solving one leads you to another, which gives you a clue to an earlier hazard you were puzzling over, which in turn sets you off somewhere else, and so on.

The number of problems in a game is obviously up to the writer of the game, but too many will soon discourage people. A problem every room will become totally boring after only a short playing session, but the intervention of a few rooms between hazards will soon perk up the player, even if he does walk into another one almost immediately.

Some problems will have to rely on a number of events taking place. In Underground Adventure, one of the hazards you're faced with is a very steep incline that you can't climb up by yourself, and the rope that you've previously used to shimmy down a steep drop isn't of any use to you here.

A little thought, or a read of the old parchment if you find it, leads you to conclude that you must build yourself a ladder, for which you need some wood (you recall a plank somewhere), some nails, and something to hit it all into shape with. Aha! The axe. But the wood has to be cut into shape first, before you can make a ladder. Only when you've got a collection of neatly cut timber can you make the ladder, and proceed to the next set of problems.

So, keep up the interest, and let people get a little further each time. And above all, don't make it an unsolvable adventure!

Program Listings From Underground Adventure

In this section we're going to give you all the lines of code that you haven't already seen, and which won't be found in the sections on verbs and data in chapter 6. So if you're going to type the whole thing in, this is the place to look at for that missing piece of code that's been puzzling you.

Of course, in common with the rest of the book we're not going to present the code without any sort of explanation. Each line will, where appropriate, be fully explained, along with an idea of how that line could be incorporated into a program of your own.

Some of the sections of the program that we'll be covering here include the rules about what happens when the bear is following you, the fights between the gargoyle and yourself, the checks to see whether you're carrying a bottle of oil, a bottle of whisky or just a plain old empty bottle, and most important of all the lines that deal with inputting data and analysing that data as it is typed in.

We'll take each section as it comes in the game, rather than diving about all over the place, so that hopefully you'll be able to see a coherent whole being slowly built up, with all the missing parts slotting logically into place, bearing in mind of course that you've already seen the movement listing, and that the data comes later on.

So, without further ado, let's get into the game.

If, by the way, you think that we've sometimes left rather large gaps on the pages, this is very true, but it's only there for you to use to put your own notes in when adapting these routines for your own purposes, so the book builds up to become more *your* book of exploring adventures rather than just a text book.

Don't worry: we'd have charged you the same even if we had filled up every page.

The Bear and the Verbs

This part of the program deals with the presence of the bear, and the handling of the verb number as it comes back from the routine in lines 390 to 413, which we'll come to in a minute.

```
206 IF TB=1 THEN OB(9)=CP:? :? "There's a tame bea
r following you."
209 IF TB=1 AND CP=45 THEN ? "Oh no! The bear snap
s the ladder in two.":OB(13)=0:P(45,1)=0
210 GOSUB 390
220 IF VB=34 THEN 1950
225 IF VB>9 AND NO$="" THEN ? :? V1$;" what?":GOTO
 210
227 IF NO$<>"" AND VB=1 AND NO=0 THEN ? "That does
n't make sense to me.":GOTO 210
240 ON VB GOTO 270,300,200,500,540,560,650,1890,69
0,300,780,880,900,950
250 ON VB-14 GOTO 1000,1050,1100,1200,1200,1250,13
00,1350
260 ON VB-22 GOTO 1400,1450,1500,1550,1600,1650,17
00,1750,1800,1850,1900,1950
262 ON VB-34 GOTO 1960,1970,3000,3200
```


Explanation

Line 206: check the bear flag (TB) and if this is set put the bear (OB(9)) into the current room (CP). Then we print up a simple message to inform the player that he's being followed around by a tame bear.

Line 209: check the bear is there, and if he is and you're climbing up the ladder in room 45, then the ladder snaps in two. The ladder disappears (OB(13) = 0), the south exit from room 45 is closed (P(45,1) = 0), and you are back in room 45.

Line 210: gosub to the input routine.

Line 220: if the verb number is equal to 34 (JUMP), then go to line 1950.

Line 225: if the verb number is greater than 9, but you've only typed in one word, print out a simple message and start again.

Line 227: if the verb number is 1 (GO), and you've typed in a non-recognised word (NO = 0, and NO$ = "something"), then print a simple message and try again.

Lines 240-262: take the verb number and go to the appropriate line in the program.

Data Validation Routine

This checks to see what you've typed in from the subroutine in lines 30000 to 30040, which we'll get to later, and splits your input up into a verb and a noun, where applicable.

```
390 ? :GOSUB 30000:N1$="":NO$="":VB$="":VB=0:NO=0:
H=0:CM=LEN(CM$):IF CM=1 THEN VB=1:NO$=CM$:GOTO 410
391 INC=0:FOR I=1 TO CM:IF CM$(I,I)=" " THEN INC=I
NC+1:IF INC=1 THEN H=I-1
392 NEXT I:IF H=1 THEN VB=1:RETURN
393 IF H=2 AND CM>3 THEN NO$=CM$(4,CM):VB=1:GOTO 4
10
396 IF H=0 THEN H=CM
397 IF H=2 THEN VB=1:RETURN
398 V1$=CM$(1,H)
399 VB$=V1$(1,3)
400 RESTORE 2214:FOR J=1 TO NV:READ VT$:IF VT$=VB$
 THEN VB=J:IF J<>NV THEN J=NV
401 NEXT J:IF VB>0 THEN 406
404 VB=1:N1$=V1$:GOTO 409
406 IF H+2>CM THEN RETURN
408 N1$=CM$(H+2,CM):IF LEN(N1$)<3 THEN NO=0:? :RET
URN
409 NO$=N1$(1,3)
410 RESTORE 2210:FOR I=1 TO NN:READ NT$:IF NT$=NO$
 THEN NO=I
411 NEXT I:IF NO=52 THEN NO=19
412 IF NO=46 THEN NO=45
413 RETURN
```


Explanation

Line 390: Go to the subroutine at 30000, set some variables to zero, check for the length of the input string. If it's equal to one, then the verb is assumed to be the first one ('GO'), and off to line 410.

Line 391: Carry on through the string until a space is found. If it is, set INC and H.

Line 392: Next step through the loop. If H is equal to one, that is, a space was found after the first character, then assume the verb is 'GO' and return from this routine.

Line 393: If the space was after the second character, and the length of the string is greater than three characters, then set NO$ to equal everything in the string from the fourth character onwards before going off to line 410.

Line 396: If the space wasn't found, then set H equal to the length of the string.

Line 397: If the space was after the second character, then assume the verb was 'GO' and return from this routine.

Line 398: Otherwise, carry on. Set V1$ equal to the first H characters of the input string.

Line 399: Set VB$ equal to the first three characters of V1$.

Line 400: Scan through all the verbs until we find a match with our verb list.

Line 401: Continue the loop, and if we've found a match then go to line 406.

Line 404: Otherwise, do the usual and assume that the verb is 'GO' before heading off to line 409.

Line 406: Check for length of string in case next line caused an illegal string split.

Line 408: Set N1$ equal to everything in the input string from the character after the space onwards. If the length of this string is less than three characters, then set the noun flag equal to zero and return from this routine.

Line 409: Set NO$ equal to the first three characters of N1$.

Line 410: Check through our list of nouns to find one we recognise.

Line 411: Continue the loop, and if a bottle of whisky is recognised, change it to an empty bottle.

Line 412: If a blazing torch is found, change it for an unlit one. The rest of the program takes care of any other possible cases of mistaken identity.

Line 413: End of the routine, so back to the main program.

Death or Glory!

This is the death routine, and is called up from a number of spots in the program in case of an untimely demise.

```
612 ? :? "You're dead!"
614 ? :? "Do you want to play again?"
616 INPUT PA$:IF PA$(1,1)="Y" THEN RUN
618 IF PA$(1,1)="N" THEN ? :? "Okay, bye.":END
620 ? "Pardon?":GOTO 616
```

Explanation

Line 612: print the 'you're dead!' message.

Line 614: ask for another game.

Line 616: if they've typed 'Y' then RUN the program again.

Line 618: if they've typed 'N' then print out a goodbye, and END the program.

Line 620: they haven't typed anything sensible, so loop back to line 616 and continue to do so until they do.

The Start and the End

These lines appear at the very start of the program, as the door slams shut behind you, and the very end, if you ever manage to get out alive.

In reverse order we have:

```
2510 ? "CONGRATULATIONS! YOU'RE FREE!!":END
```

Explanation

Line 2510: you're out, called up from another line in the program, in the OPEN routine, lines 780 to 794, so print a message of congratulations and end the program!

```
5200 ? CHR$(125):? "The gate shuts as you move.  O
h well,   you'll just have to find the key."
5205 P(3,0)=0:GF=0:GOTO 210
```

Explanation

Line 5200 : print message to say gate's closed behind you, called from line 277 in the GO routine.

Line 5205: close off the north exit from room 3 (P(3,0)), set the gate flag GF equal to 0, and go back to line 210 again.

Checking for Bottles and Torches

This routine is called up many times in the program, and is used to check to see whether you mean a lit or an unlit torch, or a bottle of whisky, a bottle of oil, or an empty bottle.

This is necessary because the data checking routine covered earlier will stop at the last noun it recognises, and the response in all the verbs will obviously depend on whether you've got the relevant torch or bottle. So we must adjust the noun number NO accordingly.

```
5300 IF NO=45 AND OB(46)=-1 THEN NO=46
5302 IF NO=19 AND OB(51)=-1 THEN NO=51
5304 IF NO=19 AND OB(52)=-1 THEN NO=52
5306 IF NO=18 AND OB(51)=-1 THEN NO=51
5308 IF NO=39 AND OB(52)=-1 THEN NO=52
5310 RETURN
```

Explanation

Line 5300: if the object number is for the old torch (OB(45)), and you're carrying the blazing torch (OB(46) = -1) then change the noun number accordingly. Return from this subroutine to whatever part of the program called it up.

Line 5302: if the object number is for the empty bottle (OB(19)), and you're carrying the bottle of oil (OB(51) = -1) then change the noun number accordingly. Return from this subroutine to whatever part of the program called it up.

Line 5304: if the object number is for the empty bottle (OB(19)), and you're carrying the bottle of whisky (OB(52) = -1) then change the noun number accordingly. Return from this subroutine to whatever part of the program called it up.

Line 5306: if the object number is for the pool of oil (OB(18)), and you're carrying the bottle of oil (OB(51) = -1) then change the noun number accordingly. Return from this subroutine to whatever part of the program called it up.

Line 5308: if the object number is for the pool of whisky (OB(45)), and you're carrying the bottle of whisky (OB(52) = -1) then change the noun number accordingly. Return from this subroutine to whatever part of the program called it up.

Line 5310: none of these options, so return from the subroutine.

The Hostile Gargoyle

This is the routine that handles the hostile gargoyle, and checks to see whether he or you have been successful in your knife and axe throwing attempts.

```
6000 ? :? "There's a hostile gargoyle watching you
 from the shadows.":IF INT(RND(0.5)*10+1)<5 THEN R
ETURN
6001 IF (INT(RND(0.5)*100+1))>99 THEN 6020
6002 ? "He has a knife; he throws it at you!":OB(4
0)=CP
```



```
6004 IF (INT(RND(0.5)*100+1))>99 THEN ? "He's kill
ed you!":GOTO 612
6006 ? "It missed!":RETURN
6008 IF (INT(RND(0.5)*10+1))>1 THEN ? "You've just
 killed a huge gargoyle.":OB(40)=0:GOTO 6014
6011 ? "You missed.":OB(40)=CP
6012 OB(4)=CP:ZZ=ZZ-1
6013 FOR I=1 TO 200:NEXT I:GOTO 200
6014 NP=0:GOTO 6012
```

Explanation

Line 6000: print out a hostile message. If the random number generated is less than 5 then the gargoyle remains watching you, but doesn't do anything else, so we return from this routine.

Line 6001: if the random number generated is greater than 99, in a range of 1 to 100, then the gargoyle turns into a thief at lines 6020 to 6040.

Line 6002: print out 'He's got a knife and throws it at you', and place the gargoyle in the room CP, meaning that he's here until the bitter end.

Line 6004: if the random number generated is greater than 99, on a scale of 1 to 100, the gargoyle has been successful and killed you. GOTO the death routine.

Line 6006: a shoddy shot and he missed, so return from this subroutine.

Line 6010: if the random number generated is greater than 1, on a scale of 1 to 10, then you've killed him, so jump to line 6014 and remove the gargoyle (0B(40) = 0).

Line 6011: yah boo! you missed, so the gargoyle stays there.

Line 6012: your axe (OB(4)) is placed in the room CP, the number of objects that you're carrying (ZZ) is therefore reduced by 1.

Line 6013: delay to enable player to read message on screen, then GOTO line 200

Line 6014: clear gargoyle present flag, since you've killed him, and GOTO line 6012

The Thieving Gargoyle

The gargoyle has turned into a thief, and here we check to see what he can take.

```
6020 ? "Oh no! A thieving gargoyle.":GS=0
6022 FOR I=1 TO 45:IF OB(I)=-1 THEN OB(I)=3:GS=GS+
1
6024 NEXT I
6038 IF GS=0 THEN ? :? "He takes nothing.  That wa
s lucky.":GOTO 6040
6039 ? :? "You'd better do an inventory."
6040 RETURN
```

Explanation

Line 6020: print simple message.

Line 6022: go through a loop to check if you're carrying any of the first 45 objects. Why 45? Object 46 is the blazing torch, and if that is removed it means that you can't continue the game. If you are carrying anything, then place it in room 3 and increase the thief counter GS. He's not really a thief, he's just mischievous.

Line 6024: continue the loop.

Line 6038: if the thief counter GS hasn't been set, then nothing has been stolen, so print a simple message on the screen and go to line 6040.

Line 6039: it has been set, so advise the player that he'd better do an inventory, since some of his belongings have been taken.

Line 6040: return from the thieving subroutine.

Of Panthers and Crevices

Two separate routines here, one for dealing with the panther in the presence of the bear, and one for the problem encountered in room 53: the narrow crevice.

```
6054 ? :? "The panther sees the bear, turns, and
 flees.":P(42,1)=43:OB(11)=0
6055 GOTO 5004
```

Explanation

Line 6054: print appropriate message, clear the south path from room 42, and remove the panther (OB(11))

Line 6055: go back to line 5004.

```
6300 OC=0:FOR I=1 TO LO:IF OB(I)=-1 THEN OC=OC+1
6302 NEXT I
6304 IF OC>1 THEN ? "You can't fit through.":GOTO
210
6306 IF OB(37)<>-1 THEN ? "You can't get through."
:GOTO 210
6308 CP=100:? "The stone glows with an eerie light
 and lets you through."
6310 GOTO 210
```

Explanation

Line 6300: set object counter OC to zero, and go through a loop LO times to check for the presence of every object. If you find one, increase the variable OC.

Line 6302: next time around!

Line 6304: if you're carrying more than one thing, then print suitable message and go to line 210

Line 6306: if you're not carrying object 37 print suitable message and GOTO 210.

Line 6308: put you in room 100, print message.

Line 6310: back to line 210 again.

May I Introduce You ?

This is just the introduction to the game, and doesn't really need any explanation. The first line just sets the screen, border and background colours, changes the screen width to 40 characters from its default of 38, and removes the cursor.

```
10000 SETCOLOR 4,1,6:SETCOLOR 2,0,0:SETCOLOR 1,0,1
4:POKE 82,0:? CHR$(125):POKE 752,1
10002 ? "          UNDERGROUND  ADVENTURE":RETURN
```


Input Subroutine

This all-important routine governs what can and what can't be typed in, and is also a way of stopping anyone using the cursor control keys to foul up the inputs.

It will allow you to delete characters only up to the input prompt, and won't allow you to press RETURN on a null prompt. If the STOP key was disabled, it would prevent crashing out of the program as well.

```
30000 OPEN #1,4,0,"K:A.FRED"
30010 CM$="":A$="":I=1
30012 ? "What now ? ";
30015 ? "*";CHR$(30);
30020 GET #1,X
30021 Z=X
30022 IF Z>127 AND Z<>155 AND Z<>156 THEN Z=Z-128:
POKE 694,0
30023 IF Z<32 OR Z=127 THEN 30020
30024 IF Z>95 AND Z<126 THEN Z=Z-32:POKE 702,64
30025 IF Z=156 THEN CM$="":I=1:? CHR$(156);:GOTO 3
0012
30026 A$=CHR$(Z)
30028 ZL=LEN(CM$):IF ZL>26 THEN 30032
30030 IF Z<>155 AND Z<>126 THEN CM$(I,I)=A$:? A$;:
I=I+1:GOTO 30015
30032 IF Z=155 AND ZL>0 THEN ? " ":I=1:CLOSE #1:?
:RETURN
30033 IF Z=126 AND ZL>1 THEN ? " ";A$;A$;:CM$=CM$(
1,I-1):I=I-1:GOTO 30015
30034 IF Z=126 AND ZL=1 THEN CM$="":I=1:? CHR$(156
);:GOTO 30012
30040 GOTO 30015
```

Explanation

Line 30000: open a file to the keyboard to allow us to get a single keypress at a time.

Line 30010: set input string CM$ equal to a null string, ditto for A$, and set I equal to 1.

Line 30012: print up the prompt in reverse field.

Line 30015: print a '*', and move the cursor back over it.

Line 30020: get a character from the keyboard.

Line 30021: set Z equal to X, which gives us the ASCII value of whatever key has been pressed.

Line 30022: if the ASCII value is less than 32 then do nothing other than go back for another key press.

Line 30024: check the ASC value of the key being pressed, and if it's greater than 95 and not equal to 126 (delete key) or 155 (return key) or 156 (delete line) then go back for another character. This stops any unwanted characters being accepted by the computer.

Line 30025: if Z equals 156 then print character string 156 to delete the line. Reset the input string to a null one, reset I to equal 1, and go to line 30012 to start again.

Line 30026: set A$ equal to whatever has been pressed.

Line 30028: take the length of the input string, and if it's greater than 27 then GOTO 30032, because we've had enough!

Line 30030: if the ASC value isn't 155 or 126, then it's a legitimate entry. Add it to our input string, and echo it back to the screen.

Line 30032: if we've pressed a carriage return, and the string length is greater than 0, then return from the subroutine.

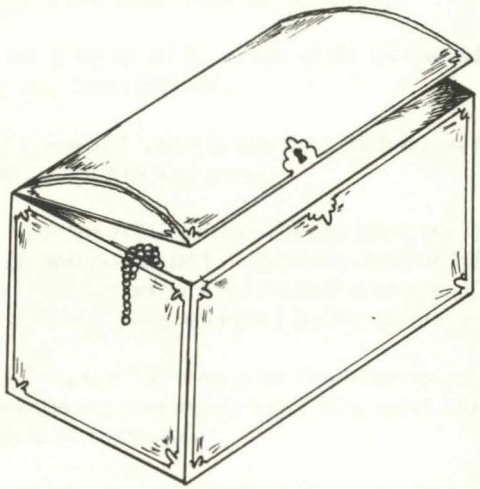
Line 30033: if we've pressed the delete key, and the string length is greater than 1, then the input string becomes the left side of the string, taking the ZL-1 first characters. Echo the character to the screen.

Line 30035 : if we've pressed the delete key and the string length is

equal to 1 then reset everything, print a character string 156 and go to line 30012.

Line 30040: go back to 30015 and start off again with the next character.

A powerful routine that could easily be adapted to trap even more characters if necessary.



5
Creating Your Own Adventures

Introduction

We've already mentioned that one of the hardest parts of creating an individual adventure game is making it just that: individual.

More and more brave new worlds are being explored every day, and a glance at any computer magazine, particularly the advertisements inside it, will reveal that there are many, many adventures on the market for all kinds of machines, and the themes used seem to range from the sublime to the ridiculous, from Colossal Caves to Pi-Men.

Five New Adventures

To the newcomer, eyeing this vast range of adventure games, it must seem that there is nothing new under the sun, and that any attempt to create a new, wonderfully different, adventure world is doomed to failure.

Nothing could be further from the truth, and in this section we're going to outline five full adventures for you, some old, some new, but all with one thing in common: they haven't been written yet.

Acknowledgment

So, if any of you take up the challenge, I hope one day to see adventures based on these themes on the market. No royalty would be charged, no copyright laws infringed, but an acknowledgment would be nice!

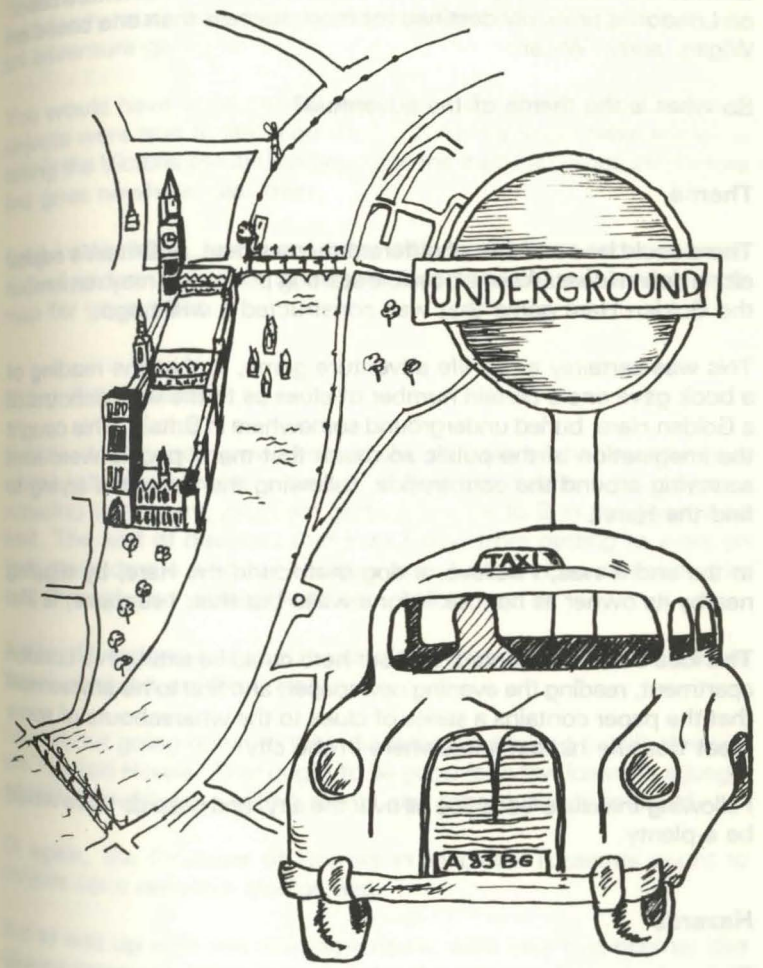
The five areas that we'll look at are all individual in their way, and none of them cross over into any of the others: they are five unique scenarios that could easily be built up into complete and enjoyable games.

We won't be giving you any maps, so that you can construct the entire game for yourself, but an overview of the game, along with a selection of possible problems, and the corresponding objects to go with them will be given.

To round off this section, we'll give a complete overview of the art of designing a new adventure.

But for now, let's head off in search of fame and glory, and arrive in...

The Streets of London



Introduction

This would be a relatively easy map to construct, since London is a well documented town. Of course, you could always choose your own town as the base for a game if you wanted to, but an adventure based on London is probably destined for more success than one based on Wigan: sorry, Wigan!

So what is the theme of the adventure?

Theme

There could be a number of different themes here, as Britain's capital city is rich in ideas. As one possible starting point, you may remember the Golden Hare game that was constructed a while ago.

This was certainly a real life adventure game, in that the reading of a book gave one a certain number of clues as to the whereabouts of a Golden Hare, buried underground somewhere in Britain. This caught the imagination of the public so much that many people were sent scurrying around the countryside, following the clues and trying to find the Hare.

In the end it was, I believe, a dog that found the Hare, by digging nearby its owner as he took it for a walk, but that, I suppose, is life!

This idea could be adapted, and our hero could be sitting in a London apartment, reading the evening newspaper, and find to his amazement that the paper contains a series of clues to the whereabouts of some great treasure hidden somewhere in the city.

Following the clues leads you all over the city, and hazards there would be a-plenty.

Hazards

The underground could go on strike, and you'd find yourself having to take a bus. None come for hours, thus losing valuable time, and then four of them turn up at once, only one going in the right direction. Which one do you catch?

You could try taking a taxi, but the taxi takes you on a scenic tour of London that takes hours before you get to your proper destination. Then the fare is too high, you haven't got enough money, and you have to haggle with a noisy taxi driver in the middle of the streets of London.

There are many other possible problems that one could construct, all based very much on real life in this re-construction of a real town into an adventure game.

You would have to be careful that the details about the locations of objects were true to life. You couldn't, for instance, have someone taking the Victoria line and ending up at the Barbican, since the Victoria line goes nowhere near there.

On the other hand, just about every diary ever printed contains a map of the London underground, so you could soon chart up a reasonable map for your game.

Other Adventures

Or indeed, the underground could also be used as the basis for your whole adventure, with a series of Reginald Perrin type disasters occuring to prevent you from getting from A to B in the given time limit. The sort of disasters that kept Perrin from getting to work on time every day: a wombat escapes from London Zoo and chews its way through the underground line, and so on.

A tour of London could give the would be adventure writer more ideas than just about anything else.

How about going down to Kew Gardens, and taking a walk through the Tropical House? That ought to be good for a few ideas for a jungle adventure, with man-eating plants and other hazards to avoid.

Or again, the Chamber of Horrors in Madame Tussauds ought to conjure up a demonic idea or two.

But to end up with one solid adventure, we'll take that original idea of some treasure being buried under the streets of London, and all you know is that it's in London somewhere.

Scenario

Reading the evening paper one Monday night in your apartment, you discover a strange article that seems to point to the location of a buried treasure buried deep underground somewhere in the city of London.

The only clue that the article gives to this location is that the treasure originally came from 'Underneath the Arches', and was moved from there many years ago.

You decide to set off in search of adventure, and head towards the arches.

Thus we could start off, and the first problem could be to get from the apartment in Muswell Hill to the Arches, which (in our adventurer's mind) would presumably be the arches behind Charing Cross Station.

After solving that problem (GET BUS, BUY TICKET, and so on), arriving at the arches would reveal a pub called the Ship and Shovel.

Is this the next clue? Does our intrepid hero have to go off and acquire a shovel and find a ship? Or does he merely go into the pub?

ENTER PUB

OK.

THE BARTENDER IS AUSTRALIAN, AND SAYS THAT 'DOWN UNDER IS THE ONLY PLACE TO BE'

WHAT NOW *

Down under? Another clue, and so we go off in search of a shovel, and somewhere to dig underground.

This could be the start of a very intriguing adventure, set as it is in real life situations (one of the bartenders really is Australian!) that would give the player a sense of familiarity, but pitching those situations into a different role from the norm.

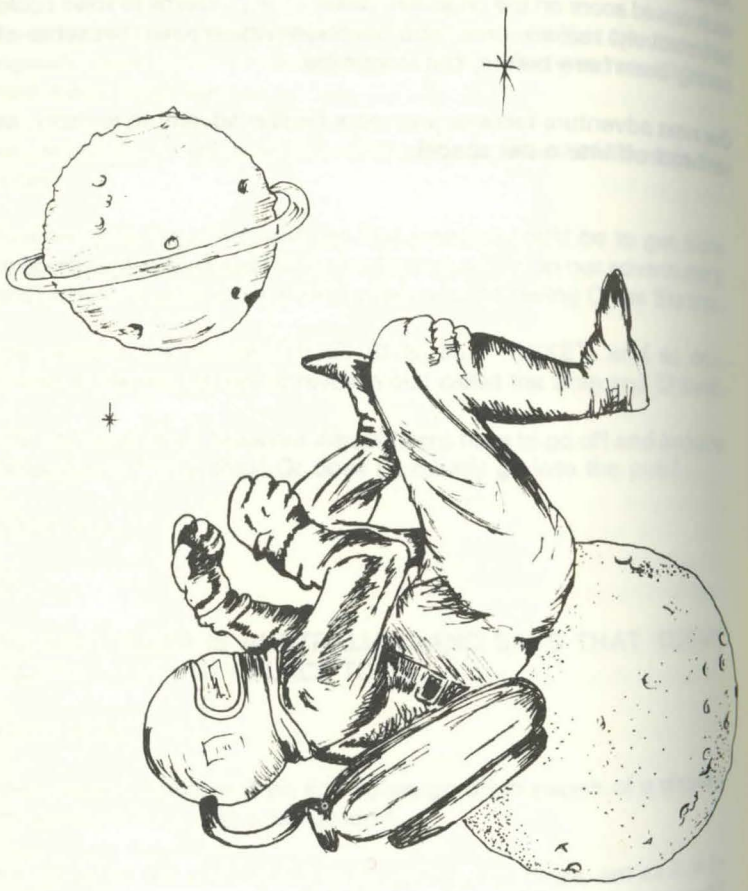
The game could encompass many famous London landmarks, each holding a clue on the trail, and each presenting its own particular problems. Big Ben would presumably feature somewhere, and, as in the famous scene in the re-make of the Thirty Nine Steps, a hazardous climb out onto the clock face could be another hazard to overcome.

Conclusion

An adventure like this is a departure from the usual themes, and as such would score on the originality stakes. The problems to solve could be (relatively) realistic ones, and the player would have that sense of having been here before, but in real life.

Our next adventure takes us into more familiar adventure territory, as we head off into outer space!

Lost in Space



Introduction

There have been a number of adventures set in outer space, and the classic Star Trek series of games that have been written for every computer under the sun, were probably the inspiration for a number of early games in this genre.

However, most of the Star Trek ones tend to be tactical battles, rather than true adventure games, and one has to go beyond the usual 'You are in command of the US Enterprise, and your mission is to destroy the Klingons' type of game, and put the player into a true adventure setting.

Theme

One possible idea would be to have your hero cast up on a dim and distant planet, deep in space, with a damaged spacecraft that needs rebuilding before he can take off again and get back home.

Here we could use some of the more traditional ideas of adventure games, but put into a modern setting. For example, the majority of thud and blunder adventures require that you carry a torch around with you. This could be replaced in this game by an oxygen tank, with a limited amount of gas, so that the mission would have to be completed in a set time.

There would be a number of different settings in this sort of adventure. One part would take place on board the damaged ship, in a search for plans, more oxygen, and equipment to repair the damage, and if the hero was silly enough to be wearing the oxygen tank on board he would lose valuable time when it came to going out onto the planet's surface.

Having thoroughly explored the ship, and cut past tangled metal, opened locked doors, and any other hazards you could dream up, the time would come to go outside, with oxygen, and the living gargoyles and little dwarfs that inhabit older adventure worlds could be replaced by hostile aliens and strange life forms.

Alien Hazards

To any reader of science fiction there should be no problem in coming up with a million and one problems for an adventurer to solve as he explores the surface of a hitherto undiscovered planet. Undiscovered, because then he won't be able to anticipate any of the problems that might arise.

Here too, as in the Streets of London, a reasonable amount of realism must come into the game, but your imagination can have a much freer rein deep in outer space.

Perhaps one could use the discovery of planet-like bodies around the star Vega, in the constellation of Lyra. A mission could be sent to explore, but a technical hitch causes the ship to crash and leaves you as the sole survivor. Being a good few light years away from earth it's impossible to signal for help, and in any case the radio probably wouldn't work, so you'd be on your own in a do-or-die mission oriented adventure.

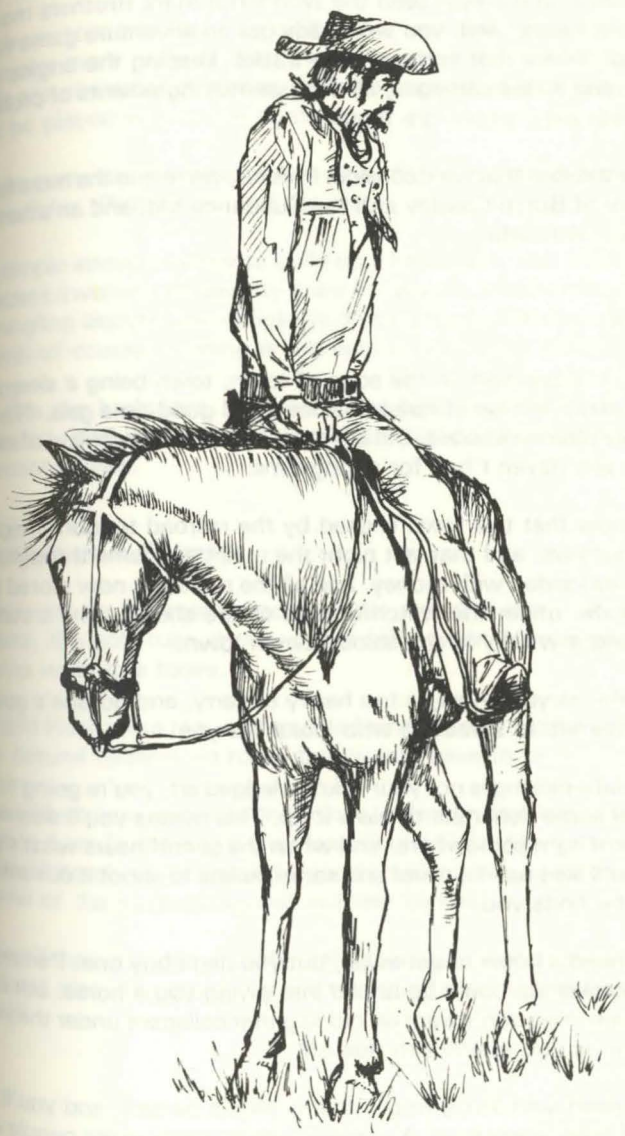
This could even be written as a two-stage adventure, in that you get the spaceship working again, but instead of steering your course for home you head off into the wilds of outer space, since the steering device hasn't been fixed properly, and then the exploration would take place aboard the ship in an effort to correct the mistake before it was too late, and you ended up in Andromeda or something. I knew I should have turned left at the Pleiades!

Conclusion

Outer space is rich in many things, and it is certainly a rich source of inspiration for the would-be adventure writer. A nice touch could be added by having various cameo roles from E.T., Darth Vader, Patrick Moore, and other stars of screen and space.

But now we'll turn our attention down home again, and travel back in time to the wild west!

Go West



Introduction

To anyone who's ever seen the wonderful Marx Brothers movie of the same name, well, you've already got an adventure game written for you! Trains that come off the tracks, keeping the engine going by burning all the carriages, all the essential ingredients of problems, disasters and humour are there.

But for the idea that we'll consider in detail, we're into the more familiar territory of Butch Cassidy and the Sundance Kid, and an attempt to rob the town safe.

Theme

You're a desperado on the edge of town, town being a sleepy little mid-west collection of hotels, saloons and good-time gals. The stars twinkle in the skys above, but are not joined by the twinkling of money, which you haven't had for a long time.

You know that this town is used by the railroad to store freight on long journeys, and that last night the mail train came through. That train was loaded with money, and all the money is now stored in the town safe, under the watchful eyes of the sheriff, who's currently watching a whisky in the saloon down town.

The safe, as you know, is too heavy to carry, and no one's going to sell dynamite to someone who looks like you!

Since safe-picking is not your acknowledged art, you're going to have to steal some dynamite to blow it up. This means you'll also need a source of light somewhere, and when the sheriff hears what's going on, you'll also need a pistol and some bullets to shoot it out with him when he finds you.

You'll need a horse to get away, but you can't buy one. Perhaps the local blacksmith could be bribed into giving you a horse, but only a good one. You don't want an old nag that collapses under the weight as soon as you attempt to ride off.

You'll need something to put the money in as well, and you'll need a small light to work by. A powerful torch would make people come and investigate, and the game would be up, you'd be slung in jail,

and somehow you'd have to get out again.

Building up the Game

The above scenario could be built into a long and enjoyable game, with many more hazards than the ones we've detailed above. The pitfalls are obviously immense, and the number of different scenes could be played with a fine humour.

Perhaps some real characters from days of old could be included, like Doc Holliday, Buffalo Bill and the rest.

It's a simple enough matter to build up a town plan, and some of the characters involved are already there for you, in terms of the sheriff, the bungling deputy who obligingly drops a key on the floor: just out of reach of course, nothing is too easy in adventure games.

From this one basic idea, there are many other themes that could be developed, and which readily lend themselves into adaptation as adventure games.

Variations on a Theme

We haven't so far mentioned Indians, the civil war, the railroad pioneers, the gold rush, or any of the other great themes that made America what it is today.

The Gold Rush would be ideal as an adventure, panning for gold, with many natural hazards en route that would have to be overcome.

You could explore underground mines, although that has been done before in Lost Dutchman's Gold and Fool's Gold. Nevertheless, the area is still barely touched, and a good adventure could still make use of some of the ideas presented in these games.

But for all that, the idea of robbing the town safe is probably the best, untapped, idea, that could lead to a very good adventure indeed. Good writing!

Murder Mystery

Introduction

One of the great untouched ideas in adventure game writing is the solving of a mystery, not necessarily a murder, although that is what we'll look at here, but any mystery.

It's hard to explain why this should be so.

Certainly detective novels sell in vast quantities year after year, and there would definitely be no shortage of plots for the adventure writer who would like to concentrate on creating a series of mystery adventures, perhaps with a connecting link like Agatha Christie's Hercule Poirot, or Conan Doyle's Sherlock Holmes (not forgetting Doctor Watson!), so that the games are linked together as a whole, although each one enjoys a separate identity as a full adventure game.

The sort of game that could be created would depend to some extent on the character adopted as the adventurer. 'Of all the adventure games in all the world, you had to walk into mine' players would enjoy a different game from 'it's all part of life's rich pageant' bungling French detectives, so the game itself would have to take on a character akin to that of the adventurer solving it.

The Story

As the great detective, a new case is brought to your attention. In the old manor belonging to the squire of the local village, a few village notables were sitting down to a pleasant evening meal when one of them pitched over, dead!

Obviously, the body is examined and found to contain an overdose of some poison, which narrows the number of suspects down to the people who were sitting down to the meal, plus all the servants who normally attend the house. In total, a dozen people are suspected, and you have to find out who the real villain was.

Developing the Story

In essence, this is a variation on the old Cluedo theme, the popular board game from Waddington's, in that there are a number of suspects within a confined area, and you have to eliminate everyone bar one

person: the murderer.

Exploration of the manor in search of clues could provide the basic adventure scenario, whilst the questioning of the suspects could be kept on a very simple level, in order to accomodate our two-word adventure type of game.

In a more advanced game of the Zork variety one could well indulge in elaborate question and answer routines, but here we'd have to restrict ourselves to much simpler ideas, perhaps using TAKE STATEMENT when you're in the same room as one of the suspects, or something like that. EXAMINE SMITH, or EXAMINE SQUIRE, might reveal some vital clue about their person.

Building the story up in this way could then provide the basis for an enjoyable romp, with the detective having to do an awful lot of work to uncover the truth.

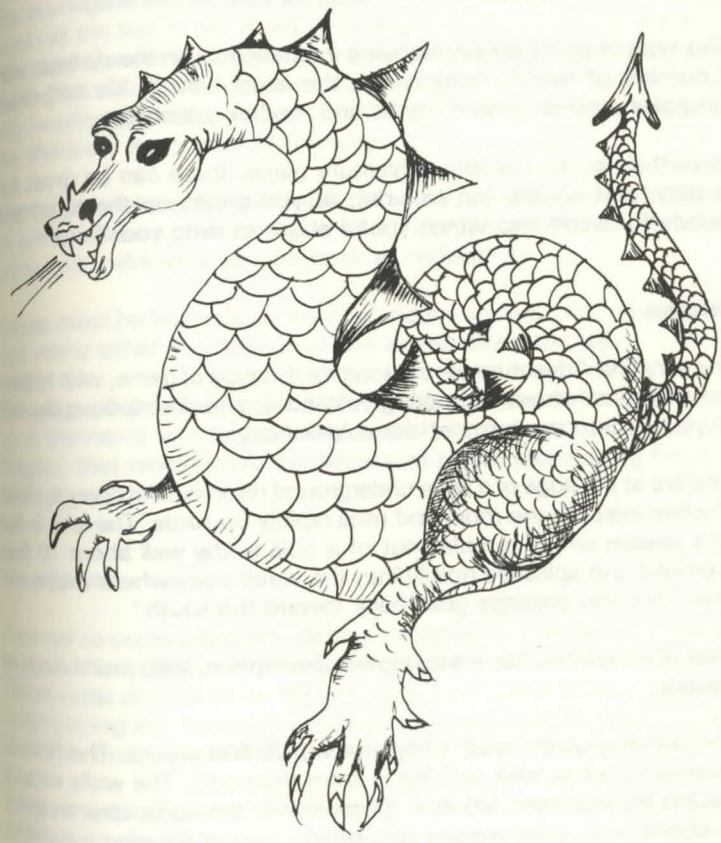
Conclusion

Detective games of this nature, that is, combining an adventure with a little bit of amateur sleuthing, have been very much neglected, and could lead to some good games if developed properly.

Not only would the exploration of the manor, or whatever environment you pitch our adventurer into, provide some entertaining diversions, by way of locked doors, guard dogs, hidden tunnels, and other hazards, but the level of brainwork required could combine to produce a good few hours entertainment.

But now, a much more traditional theme, as we enter the Valley of Death!

The Valley of Death



Introduction

The Valley of Death! You can tell from the title alone just what sort of world we're about to enter, and it is very much the traditional home of the adventure writer, with mythical beasts and dragons, hobgoblins, orcs and trolls, necromancers and black riders, and a myriad of other illustrious villains from the halls of the mountain king, or more specifically the pages of books such as *Lord of the Rings*!

This type of game is now enjoying a renaissance in the cinema, with a number of terrible films pitting the super-hero in life and death struggles against ancient myths and modern animation.

Nevertheless, as a serious adventure game, these can be great fun to play, and equally fun to write, as you dream up the weird and wonderful world into which you're about to send your hero.

Origins

The very first Adventure set the tone for this type of game, with hidden corridors, vast chasms, erupting volcanoes, and descriptions like this as you go into the heart of the colossal cave:

'You are at the edge of a large underground reservoir. An opaque cloud of white mist fills the room and rises rapidly upwards. The lake is fed by a stream which tumbles out of a hole in the wall about 10 feet overhead and splashes noisily into the water somewhere within the mist. The only passage goes back toward the south.'

Or how about this, for a true Gothic description, with just a dash of humour:

'You are in a north/south canyon about 25 feet across. The floor is covered by white mist seeping in from the north. The walls extend upward for well over 100 feet. Suspended from some unseen point far above you, an enormous two-sided mirror is hanging parallel to and midway between the canyon walls. (The mirror is obviously provided for use by the dwarves, who, as you know, are extremely vain.) A small window can be seen in either wall some 50 feet up.'

Tremendous stuff! You know straightaway the kind of world you're walking in, where characters from a Jules Verne novel like *Journey to the Centre of the Earth* might be expected to appear at any moment.

The Story

All good, traditional stuff, but the area is so vast that many adventures are still to be written that put the adventurer into a world filled with strange creatures, and countless hazards to overcome.

The story of the valley is a simple one. Stranded (you can work out how!) at the top of the valley, you have to make your way down to the mouth, walking alongside the river as it gushes down to the sea, sinking into quicksand, building canoes that do little more than pitch you headlong into the rapids, with hostile natives stalking you from the shadows every step of the way.

Strange, terrible creatures inhabit the valley, and you have to kill them all with a mixture of dexterity, wit and courage before you can safely leave and make your escape back to civilisation.

Ropes must be built across the river, native arrows must be avoided, and many other problems must be solved along the way.

The range of story lines in this sort of field is vast, and one could conjure up a thousand and one tales of sword and sorcery, dungeon and dragon, that would leave the adventure player just waiting for your next game.

Conclusion

Here we've explored just five different areas out of the many thousands that could be used to form the basis of a good, solid, adventure game. Many areas are still to be touched, and it is worth taking your time in developing an adventure scenario, as the plot and story line are major points in the success or failure of writing an adventure game.

So too are the problems that must be solved, and the ease or difficulty with which the player can progress to other levels in the game, but none the less it is the story line that will initially attract a player, and start him playing your game rather than any other.

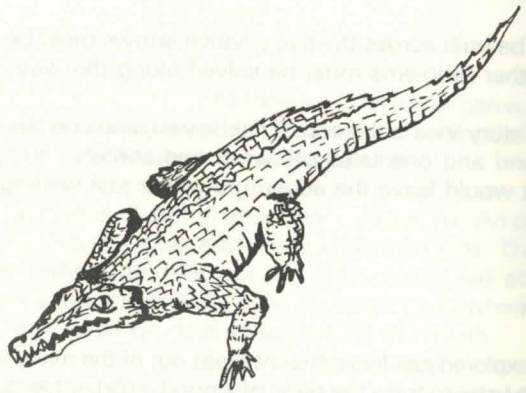
We mentioned earlier the Bible as a source of inspiration, and there are an infinite number of stories in there that could be turned into long adventure games. I'm not suggesting you wander across the desert

for forty years, but you might have fun trying to cross the Red Sea.

In the end, it is your own mind that is going to conjure up a good or a bad adventure, and the story must hold true throughout the entire game, or people will just tire of it and not consider any more of your games, not matter how good.

It is a lot easier to bore people than it is to entertain them!

So, at the risk of boring you with a lot of writing, let's take a look at the construction of Underground Adventure, and the entire selection of verbs that are used in the game.



6
Underground Adventure

In this chapter we are going to present you with the rest of Underground Adventure, to complement the listings that you've already seen in chapters 3 and 4.

All that's left to do now is to look at the data, which we'll list in full, followed by three pages of explanations for the verb data, the objects data, and the rooms data, and the entire list of verbs that are used in the game.

As you've probably never written an adventure before, we're going to go through each verb in turn, giving on one page the listing for that verb, and every part of the program that handles it, and on the opposite page we'll give an explanation for the listing, line by line.

Some verbs take up more room than others, and in particular the GET and DROP routines are quite long. Others do not take up so much space in this adventure, and so there will be a fair amount of blank space left on a number of pages. This is there for your own notes, because in many instances the verb will require a lot more code in your own games than we've used here.

Thus the space can be used to amplify on the original listing, without having to have lots of separate sheets of paper lying around everywhere.

The Scenario

You are outside a set of caves that look invitingly out at you. They

seem worthy of exploration, and so off you go into the caves and the darkness within. Finding an old torch and some matches, you light the torch, and the blazing light fills the caves. As you step further inside you hear the reverberating sound of a solid gate being slammed shut behind you, and your avenue of escape is blocked.

Somewhere in the caves lies the key to the gate, which you must find before you can escape. You got yourself into the caves, now only you can get yourself out.

We took a fairly detailed look at this adventure earlier on, so the description of the perils involved in finding the key can be read there, but it's worth pondering a while on the story line as we've got it set out here.

The Story Line

This game is set in traditional adventure territory, deep underground, fighting off mythical creatures and exploring some unusual terrain.

The tunnels and corridors much loved by Crowther and Woods have been incorporated here, together with a few swamps, a little touch of magic, and a hazy, misty land that is difficult to pass through. Some of the hazards will be familiar to players of other adventures, while some will be new, as will be the manner in which these puzzles have to be solved.

This mixture of old and new has been adopted (a) to put the player at ease with familiar territory, and the writer with a good stock of useful verbs and subroutines that can be used in other stories, and (b) to have enough new material to keep the player interested, and give the writer some idea of how new verbs can be incorporated into his own adventures.

The Writing

This is not to say that this is the only way to write adventure games - of course it isn't. But it does produce a fairly fast response from the computer, and it does allow a large range of verbs and nouns to be accommodated quite easily.

One of its weaknesses is the length of the room descriptions: these tend to be rather short, and it is sometimes difficult to produce a

different and meaningful description for each room. This problem could be surmounted by the addition of a few extra lines of code in the routine from line 5000 onwards, e.g.

```
5011 IF CP=24 THEN 8000


8000 PRINT "You're in a long,dark tunnel that has
     been carved out of the rocks."
8002 PRINT "The rocks have weathered over the years
     into a thousand and one"
8004 PRINT "fantastic formations. The light from
     your torch flickers eerily"
8006 PRINT "amongst the shadows, causing the light
     to dance about from the rocks."
```

and so on, before returning back to the main program again.

Other than that, it works, so let's look at the verbs, starting with GO.

The Complete List of Verbs

The verbs are to be covered one at a time, with two pages reserved for each verb, one for the listing and one for the explanation of that listing.

GO

This verb covers all movement in the game, in the four cardinal directions.

```
270 IF NO$<>"" AND NO=0 THEN ? "I don't know that
word.":GOTO 210
272 IF NO>28 OR NO<21 THEN ? "You'll have to re-ph
rase that.":GOTO 210
274 IF NO>24 THEN NO=NO-4
276 NO=NO-21
277 IF NO>0 AND CP=3 AND GF=1 THEN 5200
278 IF NO>0 AND PD=1 THEN ? "You've fallen into a
pit.":GOTO 612
288 IF P(CP,NO)=0 THEN ? IM$:GOTO 210
289 IF CP=53 AND NO=1 THEN 6300
290 CP=P(CP,NO):GOTO 200
```


Explanation

Line 270 - if the noun string is not equal to zero, but the noun number is, then the word is not recognised, a message is printed, and back for another input.

Line 272 - if the noun number NO is greater than 28, or less than 21, then it is not one of the eight movement nouns (NORTH, SOUTH, EAST, WEST, N, S, E, W), and so the computer doesn't understand!

Line 274 - just adjust NO, if it's greater than 24, to lie between 21 and 25.

Line 276 - adjust NO to lie between 0 and 3.

Line 277 - if we're moving in room 3 and the gate is open (GF = 1), then it's the start of the game, so GOTO5200 to set the start up by shutting the gate.

Line 278 - if we're moving but it's pitch dark (PD is set), then print message and GOTO death routine.

Line 288 - if P(room number,direction) is equal to zero, then we can't go that way, so print out message and back for more input.

Line 289 - if we're in room 53 and we're trying to go south, then GOTO 6300

Line 290 - update the room number CP from the variable P, then GOTO 200

GET

This verb handles the picking up of all objects in the game.

```
300 IF NO=0 THEN 1900
302 GOSUB 5300
304 IF OB(NO)=-1 THEN ? "You've already got it.":G
OTO 210
306 IF OB(NO)<>CP THEN ? "I can't see it here.":GO
TO 210
308 IF NO=18 AND OB(19)<>-1 THEN ? "You need a con
tainer.":GOTO 210
310 IF NO=39 AND OB(19)<>-1 THEN ? "You need a con
tainer.":GOTO 210
312 IF NO=39 AND OB(19)=-1 THEN OB(39)=0:OB(19)=0:
NO=52:ZZ=ZZ-1:GOTO 389
314 IF NO=18 AND OB(19)=-1 THEN OB(18)=0:OB(19)=0:
NO=51:ZZ=ZZ-1:GOTO 389
315 IF NO=1 OR NO=3 OR NO=6 OR NO=9 OR NO=11 OR NO
=17 THEN ? "You must be joking.":GOTO 210
316 IF NO=20 OR NO=29 OR NO=30 OR NO=31 OR NO=32 O
R NO=35 OR NO=36 THEN ? "You can't get that.":GOTO
 210
318 IF NO=40 OR NO=41 OR NO=43 OR NO=49 THEN ? "No
t on. chief.":GOTO 210
320 IF NO=8 OR NO=50 THEN ? "What for?":GOTO 210
322 IF ZZ>4 THEN ? "You're carrying too much.":GOT
O 210
324 IF NO=12 AND CP=10 THEN P(10,3)=0
326 IF NO=15 AND SC=0 THEN ? "Not yet, you don't."
:GOTO 210
389 ZZ=ZZ+1:? "Ok.":OB(NO)=-1:GOTO 210
```


Explanation

Line 300 - if the noun number is zero, then we don't know what the noun is, so GOTO 1900 to print out message.

Line 302 - GOSUB to routine to check bottles and torches.

Line 304 - if the object number is set to minus 1, we're already carrying it!

Line 306 - if the object number isn't equal to the room number, then it isn't here, so print message and try again.

Line 308 - if you're trying to get object 18, the pool of oil, but you're not carrying the empty bottle, object number 19, then you can't have it!

Line 310 - ditto for trying to get object 39, the whisky.

Line 312 - on the other hand, if you want the whisky and you are carrying the bottle, then you can have it. The pool of whisky disappears, change the noun number to refer to the bottle of whisky, object 52, set the empty bottle to disappear, and decrement the number of objects being carried counter ZZ before GOTO 389

Line 314 - ditto for the pool of oil

Line 315 - list of objects (see data tables later) that you can't carry: mainly big things that would be too heavy, so if you're trying to get one of them, print out a suitable message and go back for another input.

Line 316 - more unobtainable objects.

Line 318 - and yet more, including object 49, the word PROGRAM: someone would type it in!

Line 320 - silly objects that people might try to pick up, a pile of rubble and some broken glass.

Line 322 - check to see how much is already being carried.

Line 324 - if you pick the plank up from room 10, then you can't get past the chasm again, so adjust everything accordingly.

Line 326 - if you're trying to get the shimmering curtain, but you haven't

worked out how to get past it (in which case the shimmering curtain counter isn't set), then you can't have it!

Line 389 - everything's OK, increase the number of objects being carried counter, put the object in your possession, and go to line 210

INVENTORY

This verb is used to give a list of everything that you're carrying, so you can take stock of a given situation, and decide what to leave behind.

```
500 ? :? "You are carrying:":GS=0:ZZ=0
502 RESTORE 2200:FOR I=1 TO LO:READ OB$:IF OB(I)=-
1 THEN ? OB$:GS=GS+1:ZZ=ZZ+1
504 NEXT I
506 IF GS=0 THEN ? "Nothing much."
508 GOTO 210
```


Explanation

Line 500 - print out a simple message, and set the variable GS to zero, and also the number of objects being carried to zero.

Line 502 - restore the data pointer to line 2200, the start of the object descriptions. Start a loop up that will be gone through LO (number of objects) times, and check to see if the object is being carried i.e. if OB(I) is equal to minus one. If it is, then print up on the screen the object's description from the variable OB$, and increment the two counters GS and ZZ.

Line 504 - NEXT step through the loop.

Line 506 - if GS is equal to zero, then you can't be carrying anything, so just print out 'Nothing much'.

Line 508 - go back and get another input.

DROP

This verb is used to drop anything that you might be carrying.

```
560 IF NO=0 THEN 1900
562 GOSUB 5300
564 IF OB(NO)<>-1 THEN ? "You haven't got it.":GOT
O 210
566 IF NO=19 OR NO=51 OR NO=52 THEN ? "Smash...":O
B(NO)=0:OB(50)=CP:ZZ=ZZ-1:GOTO 210
572 IF NO=16 THEN ? "Oh no! It vanishes in a spark
le of magiclight.":OB(16)=0:ZZ=ZZ-1:GOTO 210
574 IF NO=46 THEN OB(NO)=0:OB(45)=CP:ZZ=ZZ-1:? "Ok
.":GOTO 210
575 IF NO<>12 THEN 580
576 IF CP<>10 THEN 580
578 ? "Brilliant! Now you can walk across the  pla
nk."
579 OB(12)=CP:P(10,3)=14:ZZ=ZZ-1:GOTO 210
580 PRINT "Ok.":ZZ=ZZ-1:OB(NO)=CP:IF TB=1 THEN 584
582 GOTO 210
584 ? "The bear thinks you've thrown it at him,and
 sulkily walks off.":TB=0
586 OB(9)=INT(RND(0.5)*38+3):ZZ=ZZ-1:GOTO 210
```


Explanation

Line 560 - noun not recognised, so GOTO 1900

Line 562 - GOSUB 5300 to check the bottle and torch situation.

Line 564 - if you're not carrying it then you can't drop it!

Line 566 - if you drop the bottle, the bottle of oil or the bottle of whisky, then it breaks, the empty bottle disappears, a pile of broken glass appears in the room CP, the object counter is decreased, and it's back for another input.

Line 572 - if you attempt to drop object number 16, the mirror, it vanishes, so print out a suitable message, remove mirror and decrement object counter, then GOTO 210 for more input.

Line 574 - dropping the blazing torch causes the blazing torch to disappear, the old torch to appear in the room, the object counter to decrease, the word OK to be printed on the screen, and GOTO210

Line 575 - if you're not trying to drop object 12, the plank, then off to line 580.

Line 576 - if you're not trying to drop the plank in room 10, then off to line 580 as well.

Line 578 - print out a message of congratulations at doing something right.

Line 579 - put the plank in the room, enable the player to go west from room 10, change the description for room 10, decrement the object counter, and GOTO 210

Line 580 - everything's all right, we can drop something! Print OK, decrement the object counter, put the object in the room, and check to see if we've got the bear with us. If we have then GOTO 584

Line 582 - back to 210 for more input.

Line 584 - the bear thinks you're throwing something at it, so runs away! Set the bear flag to zero and put the bear (object 9) in a room somewhere between rooms 3 and 41.

Line 586 - decrement the object counter, and back to 210 again.

QUIT

This is the verb used to end a game, and has to ask you a couple of questions before you can actually leave the game.

It's used to give you the chance of saving your progress onto tape, should you choose to do so.

```
1890 ? "Ok.":? "Do you want to save your progress
onto  tape (Y or N)?"
1892 INPUT PR$:IF PR$(1,1)="Y" THEN 3000
1894 IF PR$(1,1)="N" THEN 614
1896 ? "Yes or no will do.":GOTO 1892
```


Explanation

Line 1890 - print out a message, and give the player the chance of saving the game onto tape, for starting again at the next session without having to go through the whole rigmarole of playing the game again.

Line 1892 - input the player's answer, and if it's a 'Y' then GOTO3000, the start of the SAVE routine.

Line 1894 - if they've pressed 'N' then GOTO 614, which gives you the chance of running through the game again before definitely finishing.

Line 1896 - they haven't typed either yes or no, so go back and wait until they have.

CROSS

This verb is used whenever the player wants to get across something, and can't be bothered to type in a direction.

In this game the verb doesn't really have any use, but in other adventures it could be a very useful way of getting from one place to another, which just by a logical NORTH or whatever the player couldn't do.

In that case, you'd have to check the room number CP, and provided that there's something in place that they can cross, whisk them across to the other side by changing CP to the appropriate value.

```
690 IF NO=0 THEN 1900
692 IF CP<>15 AND CP<>10 THEN ? "There's nothing h
ere to cross.":GOTO 210
694 IF NO<>1 AND NO<>6 AND NO<>12 THEN ? "Mmmm. Bi
zarre idea.":GOTO 210
696 ? "You'll have to type in a direction.":GOTO 2
10
```


Explanation

Line 690 - if you don't understand what noun has been typed in, go off to the standard routine at line 1900.

Line 692 - if they're not in rooms 15 or 10, the only two rooms that have got chasms in them, then there's nothing to cross, so tell the player so and go back to 210 again.

Line 694 - if they're not trying to cross the chasm, the bridge or the plank, then assume they're trying some strange breeding program, print out a suitable response on the screen, and go back to line 210 again.

Line 696 - a cop-out, since we don't really use this verb in the program, so suggest that they type in a direction instead, and then back to 210.

OPEN

There are a number of things that can be OPENed in this game, or at least that the player can attempt to open, like gates and doors, so this verb deals with all of that.

If you had treasure chests or something in your games, the relevant lines to handle opening of the chest could be used here.

```
780 IF NO=0 THEN ? "Open what?":GOTO 210
782 IF CP<>60 AND CP<>3 THEN ? "There's nothing he
re to open.":GOTO 210
784 IF CP=60 THEN 790
786 IF GF=1 THEN ? "It's already open.":GOTO 210
788 IF OB(42)<>-1 THEN ? "You haven't found the ke
y yet.":GOTO 210
789 ? "The gate swings slowly open.":GOTO 2510
790 IF DF=1 THEN ? "It's already open.":GOTO 210
792 IF OB(33)<>-1 THEN ? "You've nothing strong en
ough to open it with.":GOTO 210
794 ? "You'll have to try and say this some    oth
er way.":GOTO 210
```


Explanation

Line 780 - if you don't recognise the noun, ask them what they're trying to open, and GOTO210

Line 782 - if they're not in rooms 60 or 3 then there's nothing to open, so tell them so, and GOTO210

Line 784 - if they're in room 60 then GOTO790

Line 786 - if the gate flag is set then the gate is already open, so tell them, and GOTO210

Line 788 - if they're not carrying object 42, the key, then they can't open it anyway, so tell them again, and GOTO210

Line 789 - they can open the gate, so print a suitable message. This signifies the end of the game, so GOTO2510 to print a congratulatory message and end the game.

Line 790 - if the door flag is set then it's already open, so tell them so and GOTO210

Line 792 - if they're not carrying the lump of mortar they've nothing strong enough to open the door with, so tell them so by printing a suitable message on the screen, and GOTO 210

Line 794 - you don't open a door with a lump of mortar, you have to do something else, so tell them so, and GOTO210

CLOSE

This is used whenever the player attempts to close something in the game. In Underground Adventure the only things that he can close are the gate or the door, so we check for that accordingly.

```
880 IF NO=0 THEN ? "Close what?":GOTO 210
881 IF NO<>32 AND NO<>35 THEN ? "Pardon?":GOTO 210
882 IF CP=3 THEN 890
884 IF DF=0 THEN ? "It's already closed.":GOTO 210
886 ? "On your head be it.":P(60,0)=0:DF=1:GOTO 21
0
890 IF GF=0 THEN ? "It's already closed.":GOTO 210
892 ? "The gate is a magical one,and cannot be clo
sed by a mere mortal like you.":GOTO 210
```


Explanation

Line 880 - if you don't recognise the noun, ask them what it is they want to close, and GOTO210

Line 881 - if they're not trying to close the old door or the gate, then tell them that you don't understand the request, and GOTO 210

Line 882 - if they're in room 3 then GOTO 890

Line 884 - if the door flag is set to zero then the door is already closed, so tell them so and GOTO 210

Line 886 - print an appropriate message, close off the south exit from room 60, set DF equal to zero, and GOTO 210

Line 890 - if the gate flag is set to zero then it's already closed, so tell them so and GOTO 210

Line 892 - just print out a simple message telling them that the gate is a magical one, and cannot be closed by you! Then off to 210 again for another input.

EAT

Most adventure games seem to feature food of one sort or another, and although this food is very rarely intended for the consumption of the player, it is inevitable that sooner or later someone is going to attempt to eat it themselves.

Hence this routine, which copes with greedy adventure players.

```
900 IF NO=0 THEN 1900
902 GOSUB 5300
904 IF OB(NO)<>-1 AND OB(NO)<>CP THEN ? "I can't s
ee it anywhere.":GOTO 210
906 IF NO<>10 THEN ? "Are you trying to kill yours
elf?":GOTO 210
908 ? "Aaah. An exceedingly good bun.":OB(10)=0:ZZ
=ZZ-1:GOTO 210
```


Explanation

Line 900 - if you don't recognise the noun then GOTO the routine at 1900 to print out a suitable message.

Line 902 - check to see if anyone's attempting to eat the bottle or the torch. They do, they do!!

Line 904 - if the object that they want to eat isn't in their possession, and it isn't in the room either, then they can't have it, so print out a suitable message and GOTO 210

Line 906 - if they're attempting to eat anything other than object number 10, a bun, then warn them off with a suitable message.

Line 908 - fair enough, the delicious bun is eaten, with an appropriate message, the bun then disappears (inside the player's stomach), the object counter is decremented, and we go off to 210 for another input.

FEED

Since there is some food about, someone has obviously got to feed it to something, and you'd be surprised by the things some adventure players try and force on the unsuspecting occupants of the adventure world.

In Underground Adventure, the only thing that's interested in eating is the bear. and the only thing it wants to eat is the bun, apart from you, perhaps.

```
950 IF NO=0 THEN ? "I don't understand.":GOTO 210
952 IF NO<>9 THEN ? "It isn't hungry!":GOTO 210
954 IF OB(10)<>-1 THEN ? "You've nothing to feed i
t with.":GOTO 210
956 GOTO 1072
```


Explanation

Line 950 - if you don't recognise the noun, then tell the player so, and go back to line 210 again for another try.

Line 952 - if you're trying to feed anything other than the bear, then it suddenly feigns a lack of hunger, so we print a suitable message on the screen and go back to 210 again.

Line 954 - if you're not carrying the bun, object 10, then whatever you're trying to feed fobs you off with an excuse, and GOTO210

Line 956 - Aha! We're trying to feed the bear, so we go to line 1072, where this same sequence of events is handled by another verb, OFFER, in case someone decides to OFFER BUN, rather than feeding the bear.

DRINK

An occupation favoured by many adventure players, but when it comes to actually playing a game of adventure people will try and drink some very odd things indeed.

Like eating in adventures, the drink is usually reserved for someone else's use rather than that of the player, and consumption by the player will, in the end, result in an adventure that can't be finished.

Still, they don't know this when they start, and so the appropriate routine has to be inserted to handle this.

```
1000 IF NO=0 THEN 1900
1002 GOSUB 5300
1004 IF NO<>51 AND NO<>52 THEN ? "You must be joki
ng.":GOTO 210
1006 IF NO=51 THEN ? "Urgggh! Revolting.":OB(51)=0
:OB(19)=-1:GOTO 210
1008 ? "Glug Glug Glug ... HIC!":OB(52)=0:OB(19)=-
1:GOTO 210
```


Explanation

Line 1000 - if you don't understand the noun, then it's off to the subroutine at line 1900

Line 1002 - GOSUB 5300 to check that we're not mixing up bottles and torches.

Line 1004 - if they're not trying to drink either of the two liquids in the program, i.e. the oil or the whisky, then print out a suitable statement and GOTO 210 as usual.

Line 1006 - some people try to drink strange things in these games, and oil is one of them. However, if that is what they want to drink they must face the consequences, so print a'this is revolting' message on the screen, remove the bottle of oil from the player's possession, replace it with an empty bottle, and GOTO 210.

Line 1008 - if you will drink whisky! Print out the message, remove the bottle of whisky, replace it with an empty bottle, and GOTO 210.

OFFER

This is one of the commonest ways of transferring possessions from the player to someone else, and in this adventure there are two things that change hands and get you through a couple of awkward spots.

```
1050 IF NO=0 THEN 1900
1052 GOSUB 5300
1054 IF OB(NO)<>-1 THEN ? "You're not carrying it.
":GOTO 210
1056 IF NO=10 THEN 1070
1058 IF NO<>52 THEN ? "No-one's interested.":GOTO
210
1060 IF CP<>50 THEN ? "No-one here wants it (excep
t you).":GOTO 210
1062 ? "The denizen of the caves downs it in onedr
aught, and then slopes off."
1066 OB(52)=0:OB(19)=-1:P(50,3)=55
1067 OB(29)=0:GOTO 210
1070 IF CP<>27 THEN ? "Nobody's interested.":GOTO
210
1072 ? "The bear wolfs down the bun, and stands as
ide to reveal a new path. He attaches himself to y
ou like a limpet."
1074 P(27,0)=28:OB(10)=0:ZZ=ZZ-1:TB=1:GOTO 210
```


Explanation

Line 1050 - if you don't recognise the noun, then ask them what they want to offer by going to the routine at 1900, and thus GOTO 210

Line 1052 - check that we're not confusing the various bottles and torches by the subroutine at 5300

Line 1054 - if they're not carrying the object you can't offer it, so print out the message and GOTO 210

Line 1056 - if they're talking about the bun then GOTO1070

Line 1058 - if they're not carrying the bottle of whisky, then forget it! Print out message and GOTO 210

Line 1060 - if they're not in room 50 then there's no one who's interested in the whisky, except them, so print out the message and GOTO 210

Line 1062 - aha! The denizen of the caves gratefully accepts their kind present, so print out a suitable message.

Line 1066 - remove the bottle of whisky, and replace with an empty one. Allow them to go west from room 50 to room 55.

Line 1067 - remove the denizen of the caves, and GOTO 210

Line 1070 - if they're not in room 27, then no one's interested. GOTO 210

Line 1072 - the bear eats the bun! Print out message.

Line 1074 - allow them to go north from room 27 to room 28, decrement the object counter, set the bear flag so that he tags along behind the player, and GOTO 210

WAVE

One of the key features of most adventures is waving something, which can quite often cause a magical feat, and usually this happens relatively early on in a game.

This early success seems to go to some people's heads, and they then merrily wave anything they can get their hands on, so we have to check for all of that.

```
1100 IF NO=0 THEN 1900
1102 GOSUB 5300
1104 IF OB(NO)<>-1 AND OB(NO)<>CP THEN ? "It isn't
 here.":GOTO 210
1106 IF NO<>2 THEN ? "Wave, wave, wave, but nothin
g happens.":GOTO 210
1108 IF CP<>15 THEN ? "Nothing happens.":GOTO 210
1110 IF BR=1 THEN ? "Silly person.":BR=0:OB(6)=0:P
(15,1)=0:P(15,2)=0:GOTO 210
1112 ? "A crystal bridge appears (lucky!)":OB(6)=C
P:P(15,1)=17:P(15,2)=16:BR=1:GOTO 210
```


Explanation

Line 1100 - if you don't recognise the noun, then GOTO 1900.

Line 1102 - our usual trip to the subroutine at line 5300.

Line 1104 - if the object is not in the player's possession, nor in the room, then print out a suitable message and GOTO 210

Line 1106 - if they're not waving object number 2, the staff, then print out silly message and GOTO 210

Line 1108 - even if they're waving the staff, nothing will happen unless they're also in room 15, so print out the message and GOTO 210 as usual.

Line 1110 - if the bridge flag is set, then tell them that they've already stood here and waved a staff and prepare the consequences by not allowing them to exit south and east from room 15, before going to 210 again.

Line 1112 - print the magic message, put the bridge in the room, allow them to go south from room 15 to room 17, set the bridge flag, allow the player to go east from room 15 to room 16, and finally GOTO 210

CUT and CHOP

In this adventure the two words are synonymous, in that both achieve the same object in the same way.

However, some games may care to give them a different meaning, so we've left them both in here.

Usually used to cut something up or chop it down, like a tree, or a tangled mass of vines, or something of that ilk.

```
1200 IF NO=0 THEN 1900
1202 GOSUB 5300
1204 IF OB(NO)<>-1 AND OB(NO)<>CP THEN ? "It ain't
 here!":GOTO 210
1205 IF NO<>3 AND NO<>5 AND NO<>12 AND NO<>15 AND
NO<>32 THEN ? "No chance.":GOTO 210
1206 IF OB(4)<>-1 THEN ? "You've no axe.":GOTO 210
1208 IF NO<>3 AND NO<>12 THEN ? "Your axe isn't st
rong enough.":GOTO 210
1210 IF NO=3 THEN 1220
1212 ? "The plank cuts beautifully.":OB(12)=0:OB(5
3)=-1:GOTO 210
1220 ? "Timberrrr!":P(21,2)=22:GOTO 210
```


Explanation

Line 1200 - if you don't recognise the noun, off to 1900

Line 1202 - as usual, check with the subroutine at 5300 before proceeding further.

Line 1204 - if the object he's trying to cut or chop isn't in the player's possession, and it isn't in the room, then print out a suitable message and GOTO 210

Line 1205 - if they're not trying to chop the tree, the shimmering curtain, the rope, the plank, or the old door, then tell them that it can't be done, and GOTO 210

Line 1206 - if the player is not carrying object number 4, the axe, then he has nothing to chop anything with, so tell him so and GOTO 210 again.

Line 1208 - unless they're trying to cut the tree or the plank, then the axe isn't strong enough and they'll have to try a different tack, so tell them so, and GOTO 210

Line 1210 - if it's the tree they're after, then line 1220 handles it.

Line 1212 - print a message about the plank, remove the plank, put the neatly sawn timber in their possession (a fine piece of axemanship!), and GOTO 210

Line 1220 - print message about the tree, let them go east from room 21 to room 22, and GOTO 210

As it stands, this will let players repeatedly chop down the ex-tree, should they choose to do so, but a simple test could be carried out to disable this.

CLIMB

In most adventures there is a degree of climbing somewhere along the way, but the ability to climb something usually depends on the player having already collected or made something else.

Such is the case with Underground, where we need (a) to find a rope, and (b) build a ladder, before we can climb the two obstacles presented to us.

```
1250 IF NO<>3 AND NO<>5 AND NO<>13 THEN ? "I beg y
our pardon?":GOTO 210
1252 IF NO=3 THEN ? "What, and ruin your suit?":GO
TO 210
1254 IF NO=5 THEN 1266
1256 IF OB(13)<>CP THEN ? "I don't see it on the g
round anywhere.":GOTO 210
1257 IF CP<>45 AND CP<>47 THEN ? "There's no point
 here.":GOTO 210
1258 CP=92-CP:OB(13)=CP:GOTO 200
1266 IF OB(5)<>CP THEN ? "It doesn't appear to be
on the ground.":GOTO 210
1267 IF CP<>35 AND CP<>36 THEN ? "There's no point
 here.":GOTO 210
1269 CP=71-CP:OB(5)=CP:GOTO 200
```


Explanation

Line 1250 - if the object concerned isn't the tree, the rope or the ladder, then the player can't climb it, so a suitable response is given before going back to 210

Line 1252 - if the player is attempting to climb object 3, the tree, an excuse is given as to why he can't, and back to 210

Line 1254 - if the player is trying to climb object 5, the rope, then off to 1266

Line 1256 - if the ladder isn't in the room then the player can't climb it, so print a message and GOTO 210

Line 1257 - if the player isn't himself in rooms 45 or 47 then there's no point in climbing the ladder, so print a message and GOTO 210

Line 1258 - if the player's in room 45, then put him in room 47, put the ladder in room 47, then GOTO 200 for a LOOK. Otherwise, put the ladder and the player in room 45 again, and GOTO 200

Line 1266 -if the rope isn't in the room, then the player can't climb it, so print a message and GOTO 210

Line 1267 - if the player himself isn't in rooms 35 or 36 then there's no point in climbing the rope, so print a message out and GOTO 210

Line 1268 - if the player's in room 35 then put him and the rope in room 36, print a message and GOTO line 200. Otherwise, put the player and the rope in room 35, and GOTO 200

LIGHT

Torches are quite a common feature of adventures, and obviously they'll have to be lit at some time or other during the course of the game.

Occasionally other objects will have to be lit as well, as in the case of Underground where the dynamite has to be used, and checks must be made to see what the player is trying to light, and if he's got the necessary equipment to light something with: usually matches.

```
1300 IF NO=0 THEN 1900
1302 GOSUB 5300
1304 IF OB(NO)<>-1 AND OB(NO)<>CP THEN ? "It ain't
 here.":GOTO 210
1306 IF OB(44)<>-1 THEN ? "You've nothing to light
 it with.":GOTO 210
1308 IF NO<>45 AND NO<>7 THEN ? "It won't light.":
GOTO 210
1310 IF NO=7 THEN 1320
1312 IF OB(46)=-1 THEN ? "It's already lit.":GOTO
210
1314 ? "Ok.":OB(46)=-1:OB(45)=0:PD=0:GOTO 210
1320 IF OB(7)=-1 THEN ? "You've just blown yoursel
f up (careless)":GOTO 612
1322 IF CP<>4 THEN ? "Booom! The dust clears, to r
eveal that  nothing's changed.":OB(7)=0:ZZ=ZZ-1:GO
TO 210
1324 ? "Booom! The wall is blown to smithereens.":
OB(7)=0:ZZ=ZZ-1:P(4,3)=5:GOTO 210
```


Explanation

Line 1300 - unrecognised noun, so GOTO 1900

Line 1302 - the usual check using the subroutine at 5300

Line 1304 - check to see if the object they're trying to light is either being held or in the room, and if not print a message and GOTO 210

Line 1306 - if the player isn't holding object 44, i.e. the matches, then he can't light anything, so print message and GOTO 210

Line 1308 - if they're not trying to light the torch or the dynamite, then it can't be lit, so print message and GOTO 210

Line 1310 - if they're trying to light the dynamite then GOTO 1320

Line 1312 - if they're carrying the blazing torch, object 46, then there's no point lighting the torch, so say so and GOTO 210

Line 1314 - OK, the player can light the torch, the old torch disappears, the blazing torch is placed in the player's possession, the darkness flag PD is set to zero, and we can GOTO 210

Line 1320 - if the player is holding the dynamite while trying to light it, this is understandably fatal, so GOTO 612 for the death routine.

Line 1322 - if the player isn't in room 4, then the dynamite blows up but nothing else happens, so make the dynamite disappear, decrement the object counter, and GOTO210.

Line 1324 - print a suitable message, remove the dynamite, decrement the object counter, enable the player to go west from room 4 to room 5, change the description of room 4, and GOTO 210

ATTACK

Adventure players seem to be a bloodthirsty lot when they get a keyboard in front of them, and quite often like to attack things.

Usually it doesn't do any good, although here we've let them off with a mild warning. However, the routine could easily be adapted to include things like killing the player if he attempts to attack a dragon, or something of that ilk.

```
1350 IF NO=0 THEN 1900
1351 GOSUB 5300:IF OB(NO)<>CP AND OB(NO)<>-1 THEN
? "Where is it?":GOTO 210
1352 IF NO<>9 AND NO<>11 AND NO<>29 AND NO<>30 AND
 NO<>31 THEN ? "Strange idea. No!":GOTO 210
1354 ? "This is not your best idea to date.":GOTO
210
```


Explanation

Line 1350 - the noun isn't recognised, so go to the routine at line 1900 and print an appropriate message.

Line 1351 - GOSUB the subroutine at 5300 to clear up any doubts about bottles or torches, and then check that the thing to be attacked is either in the player's possession or in the room. If it isn't, print a suitable message and GOTO line 210

Line 1352 - if the player isn't trying to attack the bear, the panther, the denizen of the caves, the giant spider or the giant fly, then he's probably trying to attack an inanimate object, so print out the message and go back to 210 for another input.

Line 1354 - warn the player gently that this is not a very good idea, and go back to 210 and try something else.

KILL

Another verb that people try to use quite a lot, although in Underground Adventure we haven't utilised the verb at all.

Nevertheless, in case people do try to type in the command to KILL BEAR, or whatever, a few lines of code are necessary in order to deal with the request.

This is certainly one of the routines that could be expanded in your own games. For instance, attempting to kill the dragon in the original Adventure game produces the following series of responses:

kill dragon

WITH WHAT? YOUR BARE HANDS?

yes

CONGRATULATIONS! YOU HAVE JUST KILLED A 20 TON DRAGON! HARD TO BELIEVE, ISN'T IT?

```
1400 IF NO=0 THEN 1900
1401 GOTO 1351
```


Explanation

Line 1400 - if it's an unknown noun, request the player to type in what it is he really wants to try to kill using the subroutine at 1900, then go back to line 210

Line 1401 - GOTO line 1351 and deal with everything from there.

Killing with a straightforward command like this isn't allowed in this game!

HIT

Another common verb, this one is usually used when people are getting fed up with not being able to get anywhere, and are typing in commands almost at random with the vain hope of getting something to happen.

In one of the games printed in this book, hitting a wall does produce some response other than getting a sore hand. However, it isn't this one, so all this routine consists of in Underground Adventure is:

```
1450 IF NO=0 THEN 1900
1451 GOSUB 5300:IF OB(NO)<>CP AND OB(NO)<>-1 THEN
? "Where is it?":GOTO 210
1452 ? "Do you really want to bruise your hand?":G
OTO 210
```


Explanation

Line 1450 - if it's an unknown noun, request the player to type in what it is he really wants to try to hit by using the subroutine at 1900, then go back to line 210

Line 1451 - GOSUB the routine at 5300 to clear up the problem of the bottles and the torches, and then check to see whether the object of the player's affections is either in the room or in his possession. If it isn't, then ask him where it is and go back to 210 again.

Line 1454 - just print a message and go back to 210 again.

Hitting anything in Underground Adventure gets you nowhere, other than hurting your hand!

MAKE

Most adventures require you to do a lot more than just trundle around, solve a few problems and find a few treasures. In order to complete the adventure, you'll usually have to make something along the way in order to get from one location to another.

In Pirate Adventure for instance, you have to build a boat, and in one of the other games listed in this book you have to make your own dynamite, since it isn't provided for you.

In this one you have to make a ladder, and the materials to do so are fairly obvious: an axe to chop the wood with, some nails to hold it all together, and of course the wood itself.

```
1500 IF NO=0 THEN 1900
1502 IF NO<>13 THEN ? "You baffle me at times.":GO
TO 210
1504 IF OB(53)<>-1 OR OB(14)<>-1 OR OB(4)<>-1 THEN
 ? "You need more materials.":GOTO 210
1506 ? "What craftmanship!":OB(13)=-1:OB(14)=0:OB(
53)=0:ZZ=ZZ-1:GOTO 210
```


Explanation

Line 1500 - if it's an unrecognised noun then print up a simple statement to that effect using line 1900 and go to line 210

Line 1502 - if the player isn't trying to make a ladder, admit that you've lost faith in his ability as an adventurer and GOTO line 210 again.

Line 1504 - check to see if the player is holding the nicely sawn timber, the nails and the axe, and if he isn't inform him that he needs to collect something else yet before he can make a ladder, and then GOTO 210 again.

Line 1506 - brilliant! you make a ladder, so print out the right message, put the ladder in the player's possession, remove the nails and the timber, and decrement the object counter by one, since we've swopped some nails and some timber for a ladder (two objects for one). Finally, back to line 210 again.

REFLECT

This is a verb that I haven't seen in any other adventure, and is used to solve a problem peculiar to this one.

It is illustrated here as an example of how easy it is to add new commands to the player's vocabulary, but like all commands there must be some clue as to the actual word involved. Most players wouldn't try to REFLECT an axe for example, but give them a mirror and it is a word that they might well try to use.

Having used it once, they will then try to reflect everything under the sun, so have a few suitable responses ready.

```
1550 IF NO=0 THEN 1900
1552 IF NO<>47 THEN ? "Do what?":GOTO 210
1554 IF CP<>93 THEN ? "Nothing happens.":GOTO 210
1556 IF SC=1 THEN ? "Again?":GOTO 210
1558 ? :? "The light is reflected back and the
 curtain falls aside."
1560 P(93,0)=95:OB(15)=CP:SC=1:GOTO 210
```


Explanation

Line 1550 - an unrecognised noun, so print an appropriate message via line 1900 and GOTO 210 to try again.

Line 1552 - if the object that the player is trying to reflect isn't the LIGHT, then print an 'I don't understand' message and try again.

Line 1554 - if the player isn't in room 93, where the shimmering curtain is, nothing happens, so print the message and back to 210

Line 1556 - if the SC counter has been set, then print a message to the effect that the player is repeating himself, and go back and try again.

Line 1558 - print the all important message.

Line 1560 - allow the player to go north from room 93 to room 95, put the shimmering curtain in room 93 (CP = 93 of course, since we're in that room), change the room description for room 93, set the SC counter, and GOTO 210

Astute readers will realise that we should also have a line like:

1551 GOSUB5300 : IFOB(16) < >-1THEN?"YOU AREN'T HOLDING THE MIRROR":GOTO210

To check that the mirror is in the player's possession!

OIL

Oil frequently occurs in adventure games, and is usually used to remove something that is being sticky and refusing to budge.

Obviously players will attempt to oil everything, so suitable responses must be made. If a player makes a mistake and oils the wrong thing, then some kind of message must be printed up, and the oil must slowly trickle away, never to be found again, thus rendering the adventure unsolvable through the fault of the player.

```
1600 IF NO=0 THEN 1900
1602 IF OB(51)<>-1 THEN ? "You've no oil.":GOTO 21
0
1604 IF CP<>79 THEN ? "Nothing worth oiling around
 here.":GOTO 210
1605 IF NO<>17 THEN ? "That was a waste of time (a
nd oil).":OB(51)=0:OB(19)=-1:GOTO 210
1606 ? "The track slides to one side.":OB(51)=0:OB
(19)=-1:P(79,2)=80:P(79,3)=81:OB(17)=0:GOTO 210
```


Explanation

Line 1600 - usual check for an unrecognised noun.

Line 1602 - if the player isn't carrying the bottle of oil, then he can't oil anything, so print the message up and GOTO 210

Line 1604 - if we're not in room 79 then there isn't anything worth oiling, so we inform the player and then go back to line 210. This is being kind to the player really, since he could have wasted his oil. Such largesse!

Line 1605 - nevertheless, if he now doesn't oil the track, object 17, he wastes all the oil, so we print up the message, remove the bottle of oil and replace it with an empty bottle, and go back to 210

Line 1606 - print the message of success, remove the bottle of oil, replace it with an empty bottle, and allow the player to go east from room 79 to room 80, and west to room 81, get rid of the track now that we've solved the problem, and GOTO 210

STAB

Not a verb that is commonly encountered, and again the use here should serve to show how easy it really is to add tailor-made commands to any adventure scenario.

It is not a word that a lot of people would at first think of, although the presence of a sword should trigger off the idea in the minds of a few players.

Still, those familiar with *Lord of the Rings*, who will have read the passage about Shelob, should know that every good Hobbit always stabs a nasty spider with his sword, and that is indeed the use of the verb in Underground Adventure.

```
1650 IF NO=0 THEN 1900
1651 GOSUB 5300
1652 IF OB(NO)<>CP AND OB(NO)<>-1 THEN ? "It isn't
 here.":GOTO 210
1653 IF OB(38)<>-1 THEN ? "You've nothing to stab
it with.":GOTO 210
1654 IF NO<>30 THEN ? "I don't think so, somehow."
:GOTO 210
1656 ? "You've killed the spider!":P(84,2)=86:P(84
,3)=85:OB(30)=0:GOTO 210
```


Explanation

Line 1650 - usual check for an unrecognised noun.

Line 1651 - usual check for bottles and torches.

Line 1652 - if the object the player is trying to stab isn't in his possession and isn't in the room, then inform him of that fact and GOTO 210

Line 1653 - if you're not holding the sword, object 38, then you can't stab anything, so print the message up and go back to line 210

Line 1654 - if the player isn't trying to stab the spider, then print a suitable message and GOTO 210

Line 1656 - print the message. Allow the player to go east from room 84 to room 86, and west to room 85, and back off to line 210 again.

Again we didn't put in any checks to make sure that the player was in the correct room (IF CP< > etc.), but the checking in line 1652 takes care of that. Similarly, we didn't set any variable to check for the fact that a player may want to kill the spider several times, but again line 1652 handles this one for us.

SPRAY

This could well be the first adventure to feature this word! I certainly can't think of any others with the word, although there are no doubt some floating around somewhere.

Being an unusual word, one has to give the player some encouragement to use it, and finding the can of fly spray after eliminating the spider should give most people the right kind of idea.

A check is made to see if it is the fly that you're trying to spray, but as usual we've been kind to the player and not exhausted the fly spray if he sprays the wrong thing.

```
1700 IF NO=0 THEN 1900
1701 GOSUB 5300
1702 IF OB(NO)<>CP AND OB(NO)<>-1 THEN ? "It isn't
 here.":GOTO 210
1703 IF OB(34)<>-1 THEN ? "You aren't holding any
spray.":GOTO 210
1704 IF NO<>31 THEN ? "Cough cough splutter splutt
er.":GOTO 210
1706 ? "You've killed it!":P(74,3)=75:OB(31)=0:GOT
O 210
```


Explanation

Line 1700 - you should be used to this by now!

Line 1701 - ditto!

Line 1702 - and again, our usual check for the presence of an object.

Line 1703 - if the player isn't holding the fly spray then he can't spray anything, so we print up the message and GOTO 210 as usual.

Line 1704 - if the object to be sprayed is not the fly, object 31, then the player only succeeds in making himself cough, and we go back to 210

Line 1706 - print the message, allow the player to go west from room 74 to room 75, and go back to line 210.

THROW

This is often given the same meaning as drop, but just as in real life, here we differentiate between a simple dropping of something and a determined throw into the middle distance.

If we attempt to throw anything other than the lump of mortar or the axe, then it is treated as if the player just wished to drop the object in question, but those two particular objects have two very important roles to fill, as we shall see:

```
1750 IF NO=0 THEN 1900
1751 IF NO<>33 AND NO<>4 THEN 562
1752 IF NO=4 AND NP=0 THEN 562
1753 IF NO=4 AND NP=1 THEN 6008
1754 IF CP<>60 THEN ? "Ok. It vanishes in a cloud
of dust.":OB(33)=0:ZZ=ZZ-1:GOTO 210
1756 ? "The door flies open!":P(60,0)=61:P(60,3)=6
5:OB(33)=0:ZZ=ZZ-1:DF=1:GOTO 210
```


Explanation

Line 1750 - as usual.

Line 1751 - if the player isn't throwing the mortar or the axe then transfer program execution to the drop routine starting at line 562

Line 1752 - if the player is throwing the axe and the gargoyle present flag isn't set, then assume he just wants to drop it, so go to line 562 again.

Line 1753 - if the player is throwing the axe, and there is a gargoyle hanging around, then we go to line 6008 and continue the fight between player and gargoyle.

Line 1754 - if the lump of mortar is thrown anywhere other than in room 60, the room with the old door, then it just disappears in a cloud of dust, nothing else happens, we decrement the object counter, and GOTO 210.

Line 1756 - it's been thrown in the right place, so print the appropriate message, allow the player to go north from room 60 to room 61, and west to room 65, and remove the lump of mortar from the game. Decrement the object counter, and set the door flag, before going off to line 210 again.

Astute readers will realise that we should also have a line like:

1753a GOSUB5300 : IFOB(33) < >-1THEN?"YOU AREN'T HOLDING THE MORTAR":GOTO210

To check that the mortar is in the player's possession! You also, round about this point, realise the value of numbering programs in steps of three or more, as we would now have to renumber line 1754 to be line 1755, in order to fit this new line in.

RUB

There are a number of things that one might be inclined to rub during an adventure, but the usual one is a lamp or torch, perhaps mindful of Aladdin and his lamp.

Indeed, rubbing the lamp in the original Adventure produces an interesting response, when you're told for the first time that the lamp is, in fact, an electrical one, so nothing much happens.

In Underground Adventure, nothing happens either, but people are wont to type in anything they can think of, so the listing goes something like:

```
1800 IF NO=0 THEN 1900
1801 GOSUB 5300
1802 IF OB(NO)<>CP AND OB(NO)<>-1 THEN ? "Where is
 it?":GOTO 210
1804 ? "Interesting, but unrewarding.":GOTO 210
```


Explanation

Line 1800 - usual check for the presence of an unknown noun.

Line 1801 - usual check in subroutine at 5300 for bottles and torches.

Line 1802 - usual check to see if an object is in the player's possession, or is in the room, and if it isn't print some kind of response and GOTO 210 again for another input.

Line 1804 - print the standard response to all RUBbing suggestions.

READ

Quite often one will find objects scattered about inside an adventure that look as if they might have something written on them, so the obvious command is to read object, to see what it says.

The replies are usually meant as helpful hints for the playing of the game, and set you off thinking in a direction you might otherwise not have thought off.

Sometimes, however, they are anything but, and give you something like the weather forecast for five years ago, although even that usually makes you think of something. Occasionally they're not even written in English, as is the case with Spelunker Today, the magazine to be found in the original Adventure, which is written in Dwarvish!

```
1850 IF NO=0 THEN 1900
1851 GOSUB 5300
1852 IF NO<>48 THEN ? "There's nothing written on
it.":GOTO 210
1854 IF OB(48)<>-1 THEN ? "You're not carrying it.
":GOTO 210
1856 ? "There's materials in here to build a   la
dder, if you look hard enough.":GOTO 210
```


Explanation

Line 1850 - usual check for an unrecognised word.

Line 1851 - usual check for bottles and torches with the subroutine at line 5300

Line 1852 - if they're not trying to read object 48, the old parchment, then tell them that nothing is written on it, and go back to line 210

Line 1854 - check to see if the object is in the player's possession, and if it isn't print a suitable message and GOTO 320 again.

Line 1856 - print the message contained on the old parchment, and go back to line 210 for the next input.

EXAMINE

This is one of the most useful words in the adventurer's vocabulary, as any object should be able to be examined, and the examination of it will reveal valuable clues about it.

Even if the result of EXAMINE TORCH reveals nothing more than IT'S JUST AN OLD TORCH, it at least tells you that the torch has no magical powers (although someone might be fooling ...!)

More often, you'll be told something about the object, about its value, its usefulness, or its actual design.

In Underground Adventure, you're just told whether it's magical or not.

```
1900 IF NO=0 THEN ? "What's ";N1$:GOTO 210
1901 IF NO=43 OR NO=1 OR NO=6 THEN ? "There's noth
ing interesting here.":GOTO 210
1902 GOSUB 5300
1904 IF OB(NO)<>-1 AND OB(NO)<>CP THEN ? "It's not
 here.":GOTO 210
1905 IF NO=2 OR NO=16 OR NO=33 OR NO=37 OR NO=38 T
HEN ? "This has useful powers.":GOTO 210
1906 ? "It's not very interesting.":GOTO 210
```


Explanation

Line 1900 - our familiar check for an unrecognised noun.

Line 1901 - if they want to examine the wall, the chasm, or the bridge, then tell them there's nothing interesting here, and GOTO 210

Line 1902 - off to 5300 to check for torches and bottles.

Line 1904 - if the object is not in the player's possession, and isn't in the room, then it can't be examined, and so back to 210.

Line 1905 - if the player is examining the staff, the mirror, the brick, the shining stone or the sword, then he's told that it has useful powers, before GOTO 210

Line 1906 - otherwise just print that it's not a very interesting object, and go back to line 210 again.

JUMP and BREAK

These are grouped together here because they don't take up much code, and they don't perform a great function in this particular game.

Nevertheless, jump could be a useful command in some games, enabling a player to jump across gaps that he couldn't simply walk across, if any player chose to take the risk.

Break is again not used here, but sometines it could be used as a test of the player's ingenuity. Something could only be broken if, say, the bear was following the player, in which case the bear would have the strength to break the object for the player.

```
1950 IF CP=15 OR CP=10 OR CP=45 THEN ? D$:GOTO 612
1952 ? "Wheee!":GOTO 210
1960 IF NO=0 THEN ? "Break what?":GOTO 210
1962 IF OB(NO)<>CP AND OB(NO)<>-1 THEN ? "Where is
 it?":GOTO 210
1964 ? "Sorry, you're not strong enough.":GOTO 210
```


Explanation

Line 1950 - if the player tries to jump in room 15, 10 or 45 (i.e. across a chasm or down a steep incline) then print the variable D$, and go into the death routine.

Line 1952 - otherwise, print out a silly message and GOTO 210 again.

Line 1960 - usual check for unknown noun.

Line 1962 - if the object isn't in the player's possession and isn't in the room then he can't break it, so tell him so and then go back to 210

Line 1964 - tell the player the sad news, and GOTO 210 again.

PUSH

A verb that is used in a number of games, and one that could have been used in this one. As it is, an attempt to push the one thing that moves results only in the player being told to try doing this in another way.

```
1970 IF NO=0 THEN ? "Push what?":GOTO 210
1971 GOSUB 5300
1972 IF OB(NO)<>-1 AND OB(NO)<>CP THEN ? "It's not
 here.":GOTO 210
1974 IF CP<>79 THEN ? "You can't.":GOTO 210
1976 ? "Try doing this some other way.":GOTO 210
```


Explanation

Line 1970 - familiar.

Line 1971 - familiar again.

Line 1972 - and again.

Line 1974 - if the player isn't in room 79, i.e. the one where the track is stuck, then there's nothing to move, so the player is told before sending program control back to line 210 again.

Line 1976 - print out a helpful message before going to line 210

SAVE

A useful, one could say vital, part of any adventure game, is the ability to stop a game in mid-flight and save one's progress onto tape, including all the room descriptions that change, the object descriptions, the positions of all the objects that have moved, the flags that indicate the successful or otherwise completion of a problem, and of course the room number.

In Underground Adventure, this is achieved by typing in SAVE PROG, in response to the WHAT NOW * prompt. It does just save the data, not the whole program!

```
3000 ? :? "Insert tape and press 'space'."
3002 OPEN #1,8,0,"C:DATA"
3004 FOR I=1 TO LO:PUT #1,OB(I):NEXT I
3006 PUT #1,CP:PUT #1,TB:PUT #1,GF:PUT #1,PD:PUT #
1,ZZ:PUT #1,SC:PUT #1,DF:PUT #1,BR:PUT #1,NP
3008 PUT #1,P(45,1):PUT #1,P(10,3):PUT #1,P(3,0):P
UT #1,P(60,1):PUT #1,P(50,3)
3010 PUT #1,P(27,0):PUT #1,P(15,1):PUT #1,P(15,2):
PUT #1,P(21,2):PUT #1,P(4,3)
3012 PUT #1,P(93,0):PUT #1,P(79,2):PUT #1,P(84,2):
PUT #1,P(84,3):PUT #1,P(74,3)
3014 PUT #1,P(60,0):PUT #1,P(60,3):PUT #1,P(69,3):
PUT #1,P(42,1):CLOSE #1:GOTO 200
```


Explanation

Line 3000 - tell the player to put a tape in the cassette unit and press space when ready.

Line 3002 - open a file for writing to the cassette unit called DATA.

Line 3004 - save the position of all the objects onto the tape using a straightforward FOR ... NEXT loop.

Line 3006 - save all the variable flags and counters.

Line 3008 - save some of the room direction data that changes as problems are solved in the course of the game.

Line 3010 - save some more

Line 3012 - and more.

Line 3014 - and more. Finally, close the file and go to line 200 again to continue the game.

LOAD

Vital of course, since we have got a save routine, and this just reads all the data back and starts the game off again at the point where it had finished.

To use this, just type in LOAD PROG in response to the first WHAT NOW * prompt.

```
3200 ? :? "Insert tape and press 'SPACE'."
3202 OPEN #1,4,0,"C:DATA"
3204 FOR I=1 TO LO:GET #1,OB:OB(I)=OB:NEXT I
3206 GET #1,CP:GET #1,TB:GET #1,GF:GET #1,PD:GET #
1,ZZ:GET #1,SC:GET #1,DF:GET #1,BR:GET #1,NP
3208 GET #1,P:P(45,1)=P:GET #1,P:P(10,3)=P:GET #1,
P:P(3,0)=P:GET #1,P:P(60,1)=P:GET #1,P:P(50,3)=P
3210 GET #1,P:P(27,0)=P:GET #1,P:P(15,1)=P:GET #1,
P:P(15,2)=P:GET #1,P:P(21,2)=P:GET #1,P:P(4,3)=P
3212 GET #1,P:P(93,0)=P:GET #1,P:P(79,2)=P:GET #1,
P:P(84,2)=P:GET #1,P:P(84,3)=P:GET #1,P:P(74,3)=P
3214 GET #1,P:P(60,0)=P:GET #1,P:P(60,3)=P:GET #1,
P:P(69,3)=P:GET #1,P:P(42,1)=P:CLOSE #1:GOTO 200
```


Explanation

Line 3200 - tell the player to put a tape in the cassette unit and press space when ready.

Line 3202 - open a file for reading from the cassette unit called DATA.

Line 3204 - read the position of all the objects.

Line 3206 - read all the variable flags and counters.

Line 3208 - read some of the room direction data that changes as problems are solved in the course of the game.

Line 3210 - read some more

Line 3212 - and more.

Line 3214 - and more. Finally, close the file and return program execution to line 200.

The Rest of the Verbs

Just four more to go now, and they all perform fairly minor functions, so we'll group these up two to a page. There should still be enough space for your own notes later.

LOOK

This doesn't even have a line of its own, but just goes to line 200. This sends it off to the subroutine at line 5000.

SCORE

One line only, and this is:

```
540 ? "What do you think this is, a cricket    mat
ch?":GOTO 210
```

Just a simple message, and no points to be scored at all in this game. All you have to do is survive and get out!

HELP

Another simple one, this could be used to great effect in some games, in giving vital clues for the sake of taking points away, but in Underground Adventure you get no help at all, like this:

```
650 ? "Help?  Work things out for yourself!":GOTO
210
```

Only a simple message that tells you to keep examining things and work everything out for yourself.

TAKE

In this game, TAKE functions in exactly the same way as GET, so program execution is just transferred to line 300 and everything dealt with in the usual way.

It could be useful in some ways, as we've already mentioned, in that one talks of TAKEing medicine, rather than GETting it, and there are other ways in which the two words are different.

However, in Underground Adventure they behave in the same way.

That's the end of the verbs!

Let's get on and look at some data now.

Linking Everything Together

We've had to split up the various separate parts of Undergound Adventure in order to be able to explain properly how each section works.

Consequently, the listing is split up into a vast number of different sections scattered around the length and breadth of this book. However, every single line is in here somewhere, and the only section that we haven't yet seen is the data, which follows immediately after this page.

It includes the data for the 100 rooms contained in the game, although some of these rooms are little more than tunnels and corridors. Whether you have this many in your games is up to you, since some people prefer the 'less rooms, more objects' principle of adventure writing.

This is all very well, and the trade-off in memory space saved is usually the equivalent of something like four or five rooms per object on the kind of system that we've been employing throughout the book. By all means have more objects than we've used here, but do realise that this will mean a corresponding rise in the number of verbs used. No bad thing, but it all takes up memory space, and whether you want a lot of rooms, or a lot of objects, is up to you.

Personally, I prefer the more rooms approach. It gives you lots of space to explore about in, and means that the problems presented can be spaced out at reasonable intervals, rather than coming one after the other, with little chance for the adventure player to get a good feel for the game, and for the area he is exploring.

It also seems more realistic, in that a stroll underground in a set of caves is hardly likely to throw up hundreds of objects in each room, but will provide a lot of cross-linking tunnels and corridors for you to walk along.

But we've elected to go for a hundred rooms in total, and as we've seen we'll be giving you all the descriptions in a moment.

To sum up the job of typing in this entire listing: it is scattered about all over the place, but it is all here somewhere, with the data here, the verbs earlier on in this chapter, most of the routines in the last section of chapter 4, and the moving room routine in the last section of chapter 3.

Of course you can always buy the cassette of the game and save yourself a lot of time and trouble.

The Data

This is the complete collection of data for the entire adventure, and runs through the room data first, including description and direction, the initial locations and descriptions of the objects, the shortened forms of the object names, and of course the all important verbs.

A description of how each piece of data is used follows the listing.

```
2001 ? "You're on an old track heading south    to
wards the caves.":RETURN
2002 ? "You're very close to the caves.":RETURN
2003 ? "You're at the gloomy entrance to the    ca
ves, with paths leading off in all    directions."
:RETURN
2004 ? "You're in front of a wall of rocks and  ru
bble.":RETURN
2005 ? "You're in a subterranean tomb, dotted   ab
out with cracks and crevices.  Many   narrow paths
 lead off.":RETURN
2006 ? "You're walking around the side of the   cr
evice room, getting hopelessly lost.":RETURN
2007 ? "You're surrounded by old shored-up wallsth
at look mighty unsafe to me.":RETURN
2008 ? "You're near a great chasm in the rock,  wh
ich plunges down for hundreds of feet.":RETURN
2009 ? "You're in the heart of the crevice room wi
th tunnels leading off everywhere.":RETURN
2010 ? "You're in front of a great chasm that ismu
ch too wide to jump across. Mist swayseerily to an
d fro.":RETURN
2011 ? "You're on the southern rim of the cavernin
 a maze of rocks and boulders.":RETURN
2012 ? "You're wandering about in chasm country.":
RETURN
2013 ? "You're in a room full of rocks, rocks,  ro
cks and rocks.":RETURN
2014 ? "You're on the west side of the chasm.":RET
URN
2015 ? "You're faced with a crack in the floor  th
at even Bob Beamon couldn't jump      across.":RET
URN
```

```
2016 ? "You're stuck in an east side chamber.":RET
URN
2017 ? "You're on the main track through the      ca
ves.":RETURN
2018 ? "You're away from the centre track with    a
choice of routes as the tunnels widen out into wel
l-trod paths.":RETURN
2019 ? "You're on a sharply twisting corridor,    an
d you keep banging your elbow on the  wall.":RETUR
N
2020 ? "You're on a long east-west track into     th
e heart of the mines.":RETURN
2021 ? "You're on the site of an old undergroundfo
rest, now sadly depleted.":RETURN
2022 ? "You're heading down a twisty path into    an
 old animal's lair.  Large paw prints are much in
evidence.":RETURN
2023 ? "You're surrounded by rock in a mixture    of
 corridors.":RETURN
2024 ? "You're walking along an old tunnel that ha
s been carved out of the living rock.":RETURN
2025 ? "You're at a break in the tunnel, and thepa
th twists sharply round a corner.  It soon goes ou
t of sight.":RETURN
2026 ? "You're walking along a fairly large       co
rridor, with several tunnels leading  off into the
 gloom.":RETURN
2027 ? "You're in an old room that was used manyye
ars ago to hold bear-baiting contests.A wicked spo
rt.":RETURN
2028 ? "You're at an underground T-junction       be
hind the old bear's lair.":RETURN
2029 ? "You're face  to face with a solid wall    of
 rock.":RETURN
2030 ? "You're near the heart of the bear's       hi
ding place, amidst a mound of rocks   and rubble."
:RETURN
2031 ? "You're in a comfy old cave, once used asa
resting place by the bear in bygone   times.":RETU
RN
2032 ? "You're travelling down an off-shoot fromth
e main mine.  The path looks rather   dis-used.":R
ETURN
2033 ? "You're faced with a choice of routes as yo
u stand here surrounded by propped up timbers and
walls.":RETURN
2034 ? "You're heading down a long, open tunnel,th
at is beginning to look sadly neg-     lected.":RET
URN
2035 ? "You're in front of a great incline, muchto
o steep to jump or climb.  A misty     path exits b
elow you.":RETURN
```



```
2036 ? "You're below the great incline, with     pa
ths leading off in many directions.":RETURN
2037 ? "You're at the foot of the incline, with a
corridor heading east.":RETURN
2038 ? "You're in a twisty tunnel that turns      ar
ound many bends as it heads into the  gloom.":RETU
RN
2039 ? "You're in a long, low corridor heading    ro
ughly east-west.":RETURN
2040 ? "You're in a dead end.  The walls are      ma
de of solid rock, and it looks as if  you can go n
o further.":RETURN
2041 ? "You're at the west of the caves, and it lo
oks as if the path peters out ahead ofyou.":RETURN

2042 ? "You're in an area used by great cats as a
resting place for their young.  Their smell is eve
rywhere.":RETURN
2043 ? "You're faced with a crossroads.  Routes he
ad off in many directions, and dark    paths lead i
nto the gloom.":RETURN
2044 ? "You're near a great hole in the path,     th
at looks mighty uninviting.":RETURN
2045 ? "You're at the foot of a great hill lead-in
g up into the misty wastes beyond. Thehill looks e
erie.":RETURN
2046 ? "You're lost in another dead end.  The     ca
ves halt here, and the path ahead is  too narrow t
o continue.":RETURN
2047 ? "You're at the top of the great incline, su
rrounded by mist and eerie shadows.":RETURN
2048 ? "You're near an old, scary part of the     ca
ves that is reputedly haunted.Spiders'webs hang ev
erywhere.":RETURN
2049 ? "You're near an old magical part of the    ca
ves, where many mysterious things havehappened ove
r the years.":RETURN
2050 ? "You're in an old, old part of the caves,an
d feel the presence of spirits from   long ago.":R
ETURN
2051 ? "You're on an off-shoot from the main      tr
ack.":RETURN
2052 ? "You're on a well-trod path heading        ro
ughly north-south.":RETURN
2053 ? "You're in front of a very narrow path     he
ading south.  It looks like a very    tight squeez
e to me.":RETURN
2054 ? "You're near to the magical area of the    ca
ves, with an evil sense of foreboding in the air."
:RETURN
2055 ? "You're in a wide open corridor, with      ex
its leading off everywhere.":RETURN
```

```
2056 ? "You're stuck in an old, stuffy dead end.":
RETURN
2057 GOTO 2056
2058 ? "You're walking along a well-trodden          pa
th.":RETURN
2059 ? "The path bobs and weaves amongst the        ro
cks and trails off into the distance  along unfami
liar paths.":RETURN
2060 ? "An old door, marked with the words         BE
GONE STRANGER, lies at the end of the path ahead."
:RETURN
2061 ? "You're in a maze of twisting passages,     wh
ich look disturbingly alike.":RETURN
2062 GOTO 2061
2063 GOTO 2061
2064 GOTO 2061
2065 GOTO 2061
2066 ? "The path is very indistinct here, and      th
e walls look rather damp.":RETURN
2067 ? "The path stoops low to avoid some over-    ha
nging rocks, and the dampness fills   the air.":RE
TURN
2068 ? "You're walking along a low path, beneathwa
lls dripping with water from ancient, undisturbed
streambeds.":RETURN
2069 ? "You're in a hazy, misty area, with wispysh
ades of mist swaying about you on all sides.":RETU
RN
2070 ? "The mist thins out here, and the air      se
ems clearer as the path heads off intothe darkness
.":RETURN
2071 ? "The path is a lot clearer here, as it     tu
rns amongst the dark, foreboding      rocks.":RETU
RN
2072 ? "You're in a sharply twisting corridor.":RE
TURN
2073 ? "The rocks form fantastic geometric patt-er
ns, in what the ancients used to call the FLY room
.":RETURN
2074 ? "You're in the heart of the FLY room,      lo
ng known to the ancients and long un- used.":RETUR
N
2075 ? "The low east-west corridor here was onceus
ed as an escape route from the FLY     room.":RETUR
N
2076 GOTO 2056
2077 ? "The path, now narrowing between the       fo
rbidding rocks, still heads north-     south.":RETU
RN
2078 ? "You're at the foot of a long, low north-so
uth passage.":RETURN
2079 ? "This is the heart of mining country, andma
```



```
ny old tracks and tunnels can be seen heading off.
":RETURN
2080 ? "You're in a musty old room, not used forge
nerations, with ancient cobwebs hang- ing off the
room's top.":RETURN
2081 ? "You're on the western side of the old    mi
ning tracks.":RETURN
2082 ? "This was once known as the SALVAGE room,bu
t it's now terribly neglected.":RETURN
2083 ? "You're on the approach to the SPIDER      ro
om, a fearsome remnant of legends longago.":RETURN

2084 ? "You're in the heart of the SPIDER room.":R
ETURN
2085 ? "You're in an old chamber known as the     sp
ider's graveyard. It looks very spookyand deserted
 here.":RETURN
2086 ? "This is still spider country, as can be se
en from the webs and dead bodies lyingeverywhere."
:RETURN
2087 ? "Dead end.":RETURN
2088 ? "This is the centre of the magical part  of
 the caves, with paths leading off allover.":RETUR
N
2089 ? "You're away from the main path, walking al
ong quiet, deserted tracks.":RETURN
2090 ? "The air in the distance glows with a      th
in light, as though there were some   magic up ahe
ad.":RETURN
2091 ? "The corridor is dimly lit here, and it'sha
rd to see what eerie trouble lies      ahead.":RETU
RN
2092 ? "This is magic country, and the dim pathsle
ading off into the dark passageways    offer little
 comfort.":RETURN
2093 ? "You're at the very source of the magic, fr
om the times when magic was still a    source of fe
ar.":RETURN
2094 ? "Dead end.":RETURN
2095 ? "You're on the northern side of the       ma
gical lands.":RETURN
2096 ? "The path almost fades away here, apart  fr
om a shallow track heading dimly on.":RETURN
2097 ? "The gloomy light here only serves to     pi
ck out the distorted shapes of rocks  and rubble."
:RETURN
2098 ? "You're in a complete dead end, and the   wa
ll ahead is bricked up completely.":RETURN
2099 ? "You're on a northern off-shoot from the ma
in path, picking your way amongst the debris from
aeons ago.":RETURN
2100 ? "You're in an old magical warehouse.":RETUR
```

```
N
2110 DATA 0,2,0,0,1,3,0,0,2,15,20,4,0,0,3,0,6,13,4
,9,0,5,0,7,0,9,6,8,0,10,7,0,7,12,5,10,8,11,9,0,10
2111 DATA 0,12,0,9,0,13,11,5,0,0,12,0,0,10,0,3,0,0
,0,0,18,0,15,15,33,18,19,16,34,0,17,0,32,17,0,0,0,
21,3,0,0,0,20
2112 DATA 23,0,0,21,25,22,24,0,26,0,0,23,0,23,26,0
,27,24,0,25,0,26,0,0,0,27,30,29,0,0,28,0,31,0,0,28
,0,30,0,0
2113 DATA 19,42,33,41,17,0,34,32,18,0,35,33,0,0,0,
34,0,0,38,0,0,38,39,0,37,0,0,36,0,0,40,37,0,0,0,39
2114 DATA 0,0,32,0,32,0,0,0,42,0,44,46,0,45,0,43,4
4,0,0,0,0,0,43,0,0,52,49,48,0,51,47,50,0,53,54,47
2115 DATA 0,0,48,0,48,66,0,0,47,77,0,0,49,100,0,0,
0,0,88,49,56,57,50,58,0,55,0,0,55,0,0,0,0,0,55,59,
60,0,58,0,0,59,0,0
2116 DATA 61,61,61,62,61,61,61,63,61,64,61,61,61,6
1,65,61,61,61,60,61,51,67,0,68,66,0,0,69,0,69,66,0
2117 DATA 68,0,67,0,0,71,69,0,70,72,0,74,71,0,0,73
,74,0,72,0,0,73,71,0,0,0,74,76,0,0,75,0
2118 DATA 52,78,0,0,77,79,0,0,78,0,0,0,0,83,0,79,0
,82,79,0,81,0,0,0,80,84,0,0,83,0,0,0,0,0,84,0,0,87
,0,84,86,0,0,0
2119 DATA 89,90,92,54,0,88,91,0,88,94,0,0,0,92,93,
89,91,0,97,88,0,97,0,91,90,0,0,0,0,0,93,96,0,99,0,98
,95,93,0,0,92
2120 DATA 0,0,0,96,0,96,0,0,53,0,0,0,15,20,21,34,2
4,0,40,0,27,7,42,46,0,14,93,67,79,48,98,69,0,0,0,0
,0,0,0,0
2121 DATA 50,84,74,60,76,87,3,53,63,31,73,0,0,100,
0,3,1,0,0,39,0,0,0,0,0
2122 FOR I=1 TO 100:FOR J=0 TO 3:READ P:P(I,J)=P:N
EXT J:NEXT I
2123 FOR I=1 TO LO:READ OB:OB(I)=OB:NEXT I
2200 DATA A chasm,A staff,An old tree,An axe,Some
rope,A bridge,Some dynamite,Some rubble,A golden b
ear,A bun
2202 DATA A large black panther,A plank,A ladder,S
ome nails,A curtain,A mirror,A blocked track,A poo
l of oil
2203 DATA An empty bottle,A misty wall,n,s,e,w,n,s
,e,w
2204 DATA A ghostie,A large spider,An enormous fly
,An old door,Some mortar,A fly spray
2206 DATA A gate,A narrow crack,A stone,A sword,So
me whisky,A gargoyle,A knife,A key,A wall,Some mat
ches,An old torch
2208 DATA A lit torch,A glowing light,A parchment,
prog,Broken glass,A bottle of oil,A bottle of whis
ky,Some sawn timber
2210 DATA CHA,STA,TRE,AXE,ROP,BRI,DYN,RUB,BEA,BUN,
PAN,PLA,LAD,NAI,CUR,MIR,TRA,OIL,BOT,MIS,NORTH,SOUT
```



```
H,EAST,WEST,N,S,E,W
2212 DATA GHO,SPI,FLY,DOO,MOR,SPR,GAT,CRA,STO,SWO,
WHI,GAR,KNI,KEY,WAL,MAT,TOR,TOR,LIG,PAR,PRO,GLA,BO
T,BOT,TIM
2214 DATA GO,GET,LOO,INV,SCO,DRO,HEL,QUI,CRO,TAK,O
PE,CLO,EAT,FEE,DRI,OFF,WAV,CUT,CHO,CLI,LIG,ATT,KIL
,HIT,MAK
2216 DATA REF,OIL,STA,SPR,THR,RUB,REA,EXA,JUM,BRE,
PUS,SAV,LOA
2218 DATA North,South,East,West
2220 RETURN
```

Using the Data

Here we'll explain how all the data is used, and how it all works. In other words, what are all those words and numbers that you've just typed in!

We'll start off with the room data.

Data for the Rooms

There are one hundred rooms in all, and each one is given a description. Some of these descriptions are used for a number of different rooms, in particular in the maze where we want to confuse the player totally.

The room descriptions are contained in lines 2001 onwards, and using the routine starting at line 5000, program execution is directed to the correct line and the relevant description is printed on the screen.

Using data in this way tends to limit the length of description that we can give to a room, although it is possible to lengthen some of these by having more than one print statement per room, as has been done for some of the descriptions in Castlemaze Adventure and Tunnel Adventure.

This has the effect of looking better on the screen, and also gives the player a more realistic description of the room he is currently in. Remember though that there is a limit to how much memory we can consume.

Associated with each room are four numbers, stored in the variable P(I,J), where P(I,J) refers to the Jth direction from room I.

For instance, the four values for room 1 are 0,2,0,0. This means the player cannot go north, east or west, but can go south. Moving south will take him to room 2, which has the data 1,3,0,0. This signifies that the player can move north to room 1, south to room 3, but cannot move east and west.

In room three, we have our first choice of routes, since the data for room three is 2,15,20,4 : the player can go north to room 2, south to 15, east to 20, and west to 4.

Judicious use of room numbering can greatly enhance an adventure, although this is by no means the only system in use today. However, it is possibly one of the easiest to master, and is certainly easy to program.

Data for the Nouns

Just like the rooms, each noun, or object, has two variables associated with it, and these are OB$, used to refer to the description of the noun, and OB(I), which holds the current room number of the Ith object. If this number is a zero it isn't currently in the game, and if it is equal to minus 1, it is in the possession of the player.

In line 2123 we read in this data for each and every object, including some dummy data for the direction statements NORTH, SOUTH, EAST, WEST, N, S, E, W.

This dummy data consists of a collection of zeros, as these are used to hold the words NORTH, SOUTH, EAST, WEST, N, S, E, W respectively. This enables us to use both longhand forms of typing in a movement request (GO NORTH), and the simple one-word request like NORTH, or even just N.

The shortened forms for the nouns, that is, the words that we use when analysing any data that has been typed in, are stored in lines 2210 to 2212, and are used by the variable NO$.

Data for the Verbs

This is only of use when analysing what has been typed in, and obtaining a verb number, which is then used in lines 240 to 262 of the program in order to send program execution off to the correct part of the program.

The data, in three-letter format for speed of verb identification, is stored in lines 2214 to 2216, and is used by the variable VB$ in line 2264.

This data is used throughout the program to keep the adventurer on the move, and the large number of verbs provided ensures that a reasonable degree of interest should be maintained throughout the lifetime of the game.

The final lot of data, in line 2218, is only used once, in the routine starting up at line 5000, to print out the directions which our intrepid explorer can go off in.

It is read stored in the order that the numbers in the variable P(I,J) are read. That is, NORTH first, then SOUTH, EAST and WEST.

And that's it! A whole adventure!

Conclusion on Underground Adventure

It is not the world's greatest adventure, simply because we have explained it all in great detail, so that you now know precisely how it all works, and could probably solve it in a matter of one or two sittings.

Nevertheless, it is not to be decried because of that, if it achieves the aim it set out to do: that of presenting clearly and logically a complete adventure game listing, which anyone could take and adapt to produce their own compelling adventure games.

Machine Code Adventuring

This approach, in Basic, is obviously limited, and it would be possible to write much faster games in machine code. However, to write an adventure in machine code would be the work of many, many months, possibly even years, and most of us want to see results in far less time than that.

Using the approach outlined here, it should be possible to produce adventure games at a reasonable rate, although a programmer's utility is virtually essential for writing a program this long.

Finding all the occurrences of the variable P(54, anything), and others, are problems you want answers to all the time, and most Basics aren't equipped with such useful functions as these.

Role-Playing Adventures

We also haven't really considered role-playing adventure games, although this is a subject I may tackle at a later date. Still, we have given a few brief outlines here, and even the simple approach followed throughout this book could be used as the model for a role-playing game.

The number of rooms would have to be a little smaller, but within reason, and with some competent programming, the same level of difficulty, the same kind of vocabulary, and the same number of objects, could all be retained, to provide a fascinating game.

The one real limitation of this approach is that of the acceptance of an input from the user. We have restricted ourselves to the purely VERB OBJECT school, although this hasn't stopped a large number of adventures from being very successful programs in the past, viz. Crowther and Woods, Adams, *et al.*

Verbal Adventures

To go in for a greater level of response is possibly beyond Basic, as it would take a long time to sort through the response and break it down into its proper component parts. Just because the program can accept something like VERB OBJECT ACTION, i.e. like'Take the Box and Close the Lid', doesn't mean that the player will always want to use all of those options, and the program, unless cleverly and quickly written, could find itself getting into a terrible muddle.

But the purpose of this book, and the game Underground Adventure, was to get you exploring adventures and writing them, and on a good level we have, I hope, succeeded.

Have fun adventuring, and we'll leave you with two final listings, Tunnel Adventure and Castlemaze Adventure.

7
Castlemaze Adventure

Introduction

This is a full-blown adventure listing, written using the same routines as Underground Adventure, so you should be able to follow what's going on.

It isn't as sophisticated in looks as the first game, but it is a challenging adventure that should keep you occupied for many a long day. Of course, if you cheat by looking at the listing you'll solve it very quickly, but you wouldn't do that, would you...!

We've already given you the map for this, so you should know what's going on, but watch out for the evil sorceror and the Black Knight. Oh yes, and the deadly maze is VERY deadly!

Have fun!

```
0 GOSUB 20000
1 DIM CM$(27),A$(1),VB$(3),NO$(3),V1$(10),N1$(10),
NT$(3),VT$(3),DD$(5)
2 GOSUB 1700
3 DIM GB$(21),D1$(36),DI$(19),V$(61),DB$(32),S1$(2
0),S2$(20)
4 GB$="A gold bar falls out!"
5 D1$="Gulp-gulp-gulp.  You're shrinking!"
6 DI$="You haven't got it."
7 V$="Behind the sign is a vault in the wall. The
vault is locked."
8 DB$="You must supply a direct object."
9 S1$="I don't see it here.":S2$="Don't be ridicul
ous."
10 CP=49:X=0:ZZ=1:PI=12
20 GOTO 1500
30 IF CP=52 AND KN=0 THEN 1170
40 IF OB(2,0)=-1 AND PI=CP THEN 1240
50 IF CP=29 AND SP=0 THEN 1330
60 T=T+1:GOSUB 390:IF VB$="3.1" THEN P(30,2)=31:GO
TO 1500
70 IF VB=0 AND (NO>21 AND NO<30) THEN VB=1
90 IF VB$="CRO" AND (CP=52 OR CP=53) THEN CP=105-C
P:GOTO 20
110 IF VB<>30 AND (VB>10 OR VB=2 OR VB=6) AND NO$=
"" THEN ? :? DB$:GOTO 60
140 IF VB=30 THEN 1070
160 IF VB=0 AND NO<>0 THEN ? "You must supply a ve
rb.":GOTO 60
170 IF VB=0 THEN ? "I don't know how to do that.":
GOTO 60
190 IF VB>11 AND VB<>30 AND NO=30 THEN ? "I don't
know what that means.":GOTO 60
200 ON VB GOTO 210,280,20,350,380,420,920,470,830,
1370,510,520,550,280,660,740,770
205 ON VB-17 GOTO 660,840,840,550,420,460,940,950,
980,1010,420,1050,1070
210 IF (NO<22 OR NO>29) AND NO$<>"" THEN ? "I don'
t know how to do that.":GOTO 60
220 IF NO$="" THEN ? "Where?":GOTO 60
230 IF NO>25 THEN NO=NO-4
240 NO=NO-22:IF P(CP,NO)=0 THEN ? "You can't go th
at way.":GOTO 60
250 IF CP=1 AND NO=1 AND DF=0 THEN ? "The castle d
oor is locked.":GOTO 60
260 IF CP=17 AND NO=1 AND CF=0 THEN ? "The crack i
s much too small for you.":GOTO 60
265 IF CP=18 AND NO=0 AND OB(9,0)=-1 THEN ? "The l
arge painting won't fit through thecrack.":GOTO 60
267 IF NO=0 AND OB(20,0)=CP THEN ? "The Sorceror t
urns you into a frog!":GOTO 1220
270 CP=P(CP,NO):GOTO 20
```



```
280 IF OB(NO,0)=-1 THEN ? "You've already got it."
;GOTO 60
290 IF NO=0 THEN ? "What's a ";N1$;"?":GOTO 60
300 IF OB(NO,0)<>CP THEN ? S1$:GOTO 60
310 IF NO=16 OR NO=17 OR NO=20 OR NO=21 THEN ? S2$
;GOTO 60
320 IF ZZ>4 THEN ? "Your hands are full.":GOTO 60
330 IF NO=19 AND PF=0 THEN ? V$:PF=1:OB(16,0)=CP:O
B(19,0)=-1:ZZ=ZZ+1:GOTO 60
340 ? "Ok.":ZZ=ZZ+1:OB(NO,0)=-1:GOTO 60
350 IF ZZ=0 THEN ? "You're not carrying anything."
;GOTO 60
360 ? "You are carrying:":RESTORE 2000:FOR I=1 TO
LO:READ OB$:IF OB(I,0)=-1 THEN ? OB$
370 NEXT I:? :GOTO 60
380 GOSUB 384:GOTO 60
384 J=0:FOR I=1 TO LO:IF OB(I,0)=1 THEN J=J+OB(I,1
)
385 NEXT I:? "You've scored ";J;" points out of 10
0.":IF J<100 THEN RETURN
386 ? :? "WELL DONE!":END
390 ? :GOSUB 30000:N1$="":NO$="":VB$="":VB=0:NO=0:
H=0:CM=LEN(CM$):IF CM=1 THEN NO$=CM$:GOTO 410
391 INC=0:FOR I=1 TO CM:IF CM$(I,I)=" " THEN INC=I
NC+1:IF INC=1 THEN H=I-1
392 NEXT I:IF H=1 THEN RETURN
393 IF H=2 AND CM>3 AND CM$="GO" THEN NO$=CM$(4,CM
):GOTO 410
396 IF H=0 THEN H=CM:IF H=2 AND CM$="GO" THEN VB=1
397 IF H=2 THEN ? :RETURN
398 V1$=CM$(1,H)
399 VB$=V1$(1,3)
400 RESTORE 2214:FOR J=1 TO NV:READ VT$:IF VT$=VB$
 THEN VB=J:IF J<>NV THEN J=NV
401 NEXT J:IF VB>0 THEN 406
404 N1$=V1$:GOTO 409
406 IF H+2>CM THEN RETURN
408 N1$=CM$(H+2,CM):IF LEN(N1$)<3 THEN NO$=N1$:? :
RETURN
409 NO$=N1$(1,3)
410 RESTORE 2210:FOR I=1 TO NN:READ NT$:IF NT$=NO$
 THEN NO=I:IF I<>NN THEN I=NN
411 NEXT I:? :RETURN
420 IF NO<>0 AND OB(NO,0)<>-1 THEN 980
440 IF NO=0 THEN ? "What's a ";N1$;"?":GOTO 60
445 IF NO=18 AND OB(13,0)<>CP THEN ? "Oh dear.The
vase shatters into a millionpieces, and being magi
c, vanishes.":OB(18,0)=0
446 IF NO=18 AND OB(13,0)<>CP THEN OB(18,1)=0:ZZ=Z
Z-1:GOTO 60
450 OB(NO,0)=CP:ZZ=ZZ-1:? "Ok.":GOTO 60
460 IF NO=8 OR NO=14 THEN ? "Try 'SWING'.":GOTO 60
```

```
462 IF NO=1 OR NO=4 THEN ? "Try 'SHOOT'.":GOTO 60
464 IF NO=10 THEN ? "Try 'SHARPEN'.":GOTO 60
465 IF NO=31 THEN ? "Try 'JUMP'.":GOTO 60
466 IF NO=13 THEN ? "Just 'DROP' it where you need
 it.":GOTO 60
468 ? "I don't understand.":GOTO 60
470 IF CP<49 AND CP>44 THEN CP=CP-25:GOTO 20
474 IF CP<24 AND CP>19 THEN CP=CP+25:GOTO 20
477 ? "You can't do that here.":GOTO 30
510 ? "All right then .... ";N1$:GOTO 60
520 IF OB(NO,0)<>-1 THEN ? DI$:GOTO 60
530 IF NO<>7 THEN ? S2$:GOTO 60
540 ? D1$:ZZ=ZZ-1:OB(7,0)=0:CF=1:GOTO 60
550 IF NO<>16 AND NO<>30 AND NO<>31 THEN ? "I don'
t know how to open such a thing.":GOTO 60
560 IF NO=16 AND OB(16,0)<>CP THEN ? "What vault?"
:GOTO 60
570 IF NO=16 AND OB(2,0)<>-1 THEN ? "You don't hav
e the key.":GOTO 60
580 IF NO=16 THEN ? "The vault is open.":VF=1:IF O
B(15,0)=0 THEN ? GB$:OB(15,0)=CP
590 IF NO=16 THEN 60
600 IF NO=31 THEN 1140
610 IF CP<>1 THEN ? "What door?":GOTO 60
620 IF OB(2,0)<>-1 THEN ? "You don't appear to hav
e the key.":GOTO 60
630 ? "The door is open.":DF=1:GOTO 60
660 ? "How?":GOTO 60
740 IF OB(NO,0)<>-1 THEN ? "You're not carrying it
.":GOTO 60
750 IF NO<>3 THEN ? "There's nothing written on it
.":GOTO 60
760 ? "It says:  'A secret passage lies nearby'"
761 ? "          'It opens if you number PI    '":GO
TO 60
770 IF OB(NO,0)<>-1 THEN ? DI$:GOTO 60
780 IF OB(1,0)<>-1 THEN ? "You have no bow!":GOTO
60
785 IF OB(4,0)<>-1 THEN ? "You have no arrows!":GO
TO 60
790 ZZ=ZZ-1:OB(4,0)=CP:? "Ok.":GOTO 60
830 ? "You need a tool.":GOTO 60
840 IF NO=16 OR NO=30 OR NO=31 THEN 870
850 IF OB(NO,0)<>-1 THEN 830
860 ? "I don't know how to close such a thing.":GO
TO 60
870 IF NO=16 AND OB(16,0)<>CP THEN ? "What vault?"
:GOTO 60
880 IF NO=16 THEN ? "The vault is closed and locke
d.":VF=0:GOTO 60
890 IF NO=31 THEN 1110
900 IF CP<>1 THEN ? "What door?":GOTO 60
```



```
910 ? "The door is closed and locked.":DF=0:GOTO 6
0
920 IF CP<8 THEN ? "Be persistent.":GOTO 60
922 IF CP<20 THEN ? "Examine things.":GOTO 60
924 IF CP<24 THEN ? "What goes up, must come down.
";GOTO 60
925 IF CP<34 THEN ? "Value things.":GOTO 60
926 IF CP<41 THEN ? "Do what Hansel and Gretel did
.":GOTO 60
928 IF CP<45 THEN ? "Think!":GOTO 60
930 IF CP=52 OR CP=53 THEN ? "Cross the bridge.":G
OTO 60
932 ? "This adventure has a violent beginning.":GO
TO 60
940 ? "It's value is ";OB(NO,1);" points.":GOTO 60
950 IF OB(NO,0)<>-1 THEN ? "You don't seem to have
 it.":GOTO 60
952 IF NO<>14 THEN 960
954 FOR I=1 TO 19:IF OB(I,0)=-1 THEN OB(I,0)=CP
956 NEXT I:ZZ=0:CP=23:GOTO 20
960 IF NO<>8 THEN ? "Wow, what fun!":GOTO 60
962 IF OB(20,0)<>CP THEN ? "Whoosh!":GOTO 60
965 IF SH=0 THEN ? "The sword bounces off the sorc
eror and  hits you!":GOTO 1220
967 ? "The sharp sword slices the sorceror.":SH=SH
+1:IF SH<4 THEN 30
970 OB(20,0)=0:OB(14,0)=CP:? "You've vanquished hi
m!":GOTO 30
980 IF OB(NO,0)<>-1 THEN ? "You're not carrying it
.":GOTO 60
990 IF NO<>8 THEN ? S2$:GOTO 60
995 IF OB(10,0)<>-1 THEN 830
1000 ? "The sword is now razor sharp.":SH=1:GOTO 6
0
1010 IF (CP=1 AND NO=30) OR (CP=44 AND NO=31) THEN
 1022
1011 IF OB(NO,0)<>-1 AND OB(NO,0)<>CP THEN ? S1$:G
OTO 60
1012 IF NO=17 AND OB(2,0)=0 THEN OB(2,0)=CP:? "The
re's something fallen out of his     pocket!":GOTO
 60
1013 IF NO=21 AND OB(6,0)=0 THEN OB(6,0)=CP:? "The
re's something fallen out of its     stomach!":GOT
O 60
1014 IF NO=8 AND SH=0 THEN ? "It's blunt.":GOTO 60
1016 IF NO=8 THEN ? "It's sharp.":GOTO 60
1018 IF NO=7 THEN ? "On the bottom it says 'DRINK
ME'.":GOTO 60
1020 IF NO=18 THEN ? "It's very fragile.":GOTO 60
1022 IF NO=31 THEN ? "It's big enough to jump out
of.":GOTO 60
1024 IF NO=1 OR NO=30 THEN ? "It's made out of woo
```

```
d.":GOTO 60
1026 IF NO=20 THEN ? "He's preparing to cast a spe
ll on you.":GOTO 60
1028 IF NO=13 THEN ? "It's soft.":GOTO 60
1030 IF NO=10 THEN ? "It's grey and gritty.":GOTO
60
1038 ? "It's nothing special.":GOTO 60
1050 IF NO=4 AND OB(4,0)=22 THEN ? "Look for it in
 the forest.":GOTO 60
1053 IF NO=2 AND OB(2,0)=0 THEN ? "Examine things.
":GOTO 60
1055 IF OB(NO,0)=-1 THEN ? "You're holding it!":GO
TO 60
1057 IF OB(NO,0)=CP THEN ? "It's right in front of
 you.":GOTO 60
1060 IF NO<>20 THEN ? "Just try looking for it, al
l right?!":GOTO 60
1062 ? CHR$(125):? "You're in the sorceror's tortu
re chamberand he's coming towards you with a white
hot poker!"
1064 FOR Q=1 TO 3:GOSUB 390
1065 IF VB=25 AND NO=14 AND OB(14,0)=-1 THEN 950
1066 ? "He thrusts the poker at you.":NEXT Q:GOTO
1220
1070 IF (CP>19 AND CP<24) OR CP=34 THEN ? "Going d
own ....":GOTO 1220
1080 IF CP<>44 THEN ? "Boing boing boing.":GOTO 60
1090 IF WF=0 THEN CP=43:GOTO 1080
1100 ? "You land safely in the branches of the  tr
ee (phew).":CP=21:GOTO 60
1110 IF CP<>44 THEN ? "I see no windows here.":GOT
O 60
1120 IF WF=0 THEN ? "It's already closed.":GOTO 60
1130 ? "It's stuck.":GOTO 60
1140 IF CP<>44 THEN 1110
1150 IF WF=1 THEN ? "It's already open.":GOTO 60
1160 ? "It's not easy, but you manage to get thewi
ndow open.  You see a big leafy tree  about 2 mete
rs below you."
1161 WF=1:GOTO 60
1170 ? "A black knight is charging across the   br
idge towards you!":GOSUB 390
1180 IF VB<>17 OR NO<>17 THEN 1210
1190 IF OB(1,0)<>-1 THEN ? "You have no bow!":GOTO
 1210
1195 IF OB(4,0)<>-1 THEN ? "You have no arrows!":G
OTO 1210
1200 ? "The arrow finds a chink in the knight's ar
mour: he falls to his doom."
1205 KN=1:ZZ=ZZ-1:OB(4,0)=52:OB(17,0)=52:GOTO 60
1210 ? "He skewers you with his lance."
1220 ? "You're dead.":GOTO 1370
```



```
1240 ? "A pirate sneaks up on you and steals theke
y.  Har Har Har, he chortles.I'll hidethis deep in
 me maze."
1250 PI=0:OB(2,0)=34:ZZ=ZZ-1:GOTO 60
1330 ? "A giant spider drops from the ceiling.  Oh
 no! It's moving towards you.":GOSUB 390
1337 IF VB<>17 OR NO<>21 THEN 1350
1340 IF OB(1,0)<>-1 THEN ? "You have no bow!":GOTO
 1350
1342 IF OB(4,0)<>-1 THEN ? "You have no arrow.":GO
TO 1350
1345 ? "The arrow rips into the spider.":SP=1:ZZ=Z
Z-1
1347 OB(21,0)=29:OB(4,0)=0:OB(5,0)=29:GOTO 60
1350 ? "The spider ppunces on you, and sinks itsfa
ngs into your neck.":GOTO 1220
1370 ? :? "That's your lot!":GOSUB 384:END
1500 ? CHR$(125):GOSUB 1700+CP*5:? :? "You can see
:":D=0:RESTORE 2000:FOR I=1 TO 21
1502 READ OB$:IF OB(I,0)=CP THEN ? OB$:D=D+1
1520 NEXT I:IF D=0 THEN ? "Nothing special."
1525 ?
1530 IF CP=1 AND DF=0 THEN ? "The door is locked."
1540 IF CP=35 AND VF=0 AND OB(16,0)=35 THEN ? "The
 vault is locked."
1550 IF CP=17 AND CF=0 THEN ? "A narrow crack lead
s southwards."
1560 IF CP=1 AND DF=1 THEN ? "The door is open."
1570 IF CP=35 AND VF=1 AND OB(16,0)=35 THEN ? "The
 vault is open."
1580 IF CP=17 AND CF=1 THEN ? "A wide crack leads
southwards."
1590 IF CF=0 THEN P(17,1)=0
1610 IF CP=44 AND WF=1 THEN ? "The window is open.
 You can see a tree  some 2 meters below it."
1620 K=0:? :? "You can go":RESTORE 2216:FOR I=0 TO
 3:READ DD$:IF P(CP,I)=0 THEN 1650
1630 IF K=1 THEN ? ", ";
1640 PRINT DD$;:K=1
1650 NEXT I:IF K=0 THEN ? "Nowhere fast."
1660 IF K=1 THEN ?
1670 ? :P(17,1)=18:GOTO 30
1700 NP=53:LO=35:NN=31:NV=30:DIM P(NP,3),OB(LO,1),
OB$(35):GOTO 1982
1705 ? "You are outside a medieval castle. The  pa
vement bears the enscription  'LEAVE ALL TREASURE
S HERE.'":?
1706 ? "The castle looks old and imposing, and  th
e ancient,grimy walls bear testimony  to the passi
ng of the years."
1707 ? :? "There is a door set into the wall of th
ecastle.":RETURN
```

```
1708 ? :? "There is a door set in the side of the
 castle.":RETURN
1710 ? "You're at a great crossroads, with roadshe
ading off into the distance in all directions."
:RETURN
1715 ? "You're on the great eastern road,walkingal
ong deeply rutted tracks.":RETURN
1720 ? "You're on the great western road, with ey
es watching you from all sides.":RETURN
1725 GOTO 1720
1730 GOTO 1720
1735 ? "You're walking along on the great south-er
n road. The road looks old and desert-ed, as if no
ne have passed ";
1736 ? "this way for centuries.":RETURN
1740 ? "You're in a splendid chamber, 30 feet hi
gh. This was once the main dining hallfor the old
castle.":RETURN
1745 ? "You're in a cosy sitting room, the sort of
 place where Mole and Rat may well have spent a
n evening.":RETURN
1750 ? "You're in the master bedroom (where the va
mpires sleep). Fortunately, the masteris nowhere t
o be seen.":RETURN
1755 ? "You're in a vast open corridor stretch- in
g out of sight to the south. The wallslook endless
.":RETURN
1760 ? "You're in a vast open corridor stretch- in
g away to both north and south. This seems to go
on for ever.":RETURN
1765 ? "You're at one end of a vast corridor st
retching out of sight to the north.":RETURN
1770 ? "You're in an old bedroom, with a stone fl
oor. This looks disused and neglected,but you neve
r know.":RETURN
1775 ? "You're in an old bedroom, with a wooden fl
oor. It doesn't look very comfortable,and probably
 never was.":RETURN
1780 ? "You're in an old bedroom, with a dirt fl
oor. This looks to be the most dis- shevelled of
 them all.":RETURN
1785 ? "You're in a dusty old pantry, where somedu
sty old food was prepared in bygone days.":RETUR
N
1790 ? "You're in a private art gallery, with ma
ny paintings of great anthropological significance
 (eh, Viv?)":RETURN
1795 ? "You're in an old, deserted store room. Co
bwebs hang everywhere, and it's in a disgraceful
state.":RETURN
1800 ? "You're at the top of a very large tree, wi
th sweeping views all around you.":RETURN
```



```
1805 ? "You're at the top of a bushy tree, with sw
eeping views of leaves all around you.":RETURN
1810 GOTO 1800
1815 GOTO 1805
1820 ? "You're in what was once used as the ki
tchen, with battered kitchen tools lying everyw
here.":RETURN
1825 ? "You're in an old, secret dining room, th
at was used to escape from the bustle of the main
hall.":RETURN
1830 ? "You're in a secret, shadowy alcove.Thereis
 very little room in here, and it is extremely da
rk.":RETURN
1835 ? "You're in an austere office, used by vi
siting gentry long ago. Evidence of their visits
 still remains."
1836 RETURN
1840 ? "You're in the drawing room. This place is
 beginning to look like the Marie Celeste.":RE
TURN
1845 ? "You're wandering about in the old pa
rlour, once used no doubt by old parlourmaids
.":RETURN
1850 ? "You're in the main study in the castle, an
d signs of its use still remain. It's also very du
sty in here.":RETURN
1855 ? "You're in a damp, stone passage, gettingve
ry damp yourself, but luckily not verystoned.":RET
URN
1860 ? "You're in the deep, dark, gloomy old du
ngeon, with damp walls and dirty floors.":RET
URN
1865 ? "You're in an ancient conference room, on
ce used by many leading society figures, but
 now fallen into";
1866 ? " disuse and disrepair.":RETURN
1870 ? "You're in a tower which overlooks a hugeki
ngdom down a monsterous mountain. The grass in the
 valley is ";
1871 ? "greener than green itself, and reminds
 you of home (aw).":RETURN
1875 ? "Guess what? You're in a maze of twisty li
ttle passages, and they all look the same.":RETUR
N
1880 GOTO 1875
1885 GOTO 1875
1890 GOTO 1875
1895 GOTO 1875
1900 GOTO 1875
1905 ? "You're on a long flight of stairs head- in
g down into the gloom beyond.":RETURN
1910 ? "You're on a long flight of stairs head- in
```

```
g up into the gloom beyond.":RETURN
1915 ? "You're in a mile long passage, with dampwa
lls and water dripping onto the end ofyour nose.
Stagnant water";
1916 ? " reaches up toyour ankles, and it all look
s very dark and foreboding.":RETURN
1920 ? "You're at the end of the castle, and     th
rough a small window in front of you  you can see
a dense forest."
1921 RETURN
1925 ? "You're wandering about in the dense darkfo
rest.":RETURN
1930 GOTO 1925
1935 ? "You're wandering about in the dense darkfo
rest, but the path clears a little     here to reve
al a light to ";
1936 ? "the south.":RETURN
1940 GOTO 1925
1945 ? "You're on an old path, carved out of thero
ck years ago by the hooves of many     horses.":RET
URN
1950 ? "You're at the end of the path, with     fo
rest surrounding you in all            directions."
:RETURN
1955 ? "You're in the middle of a clearing in   th
e dense, dark forest.":RETURN
1960 ? "You're on the north side of an old      br
idge crossing a slow moving river.":RETURN
1965 ? "You're on the south side of the bridge, wi
th a path heading south.":RETURN
1970 DATA 0,8,4,0,53,7,3,6,0,0,3,2,0,0,2,1,0,0,2,4
,0,0,2,5,2,7,0,0,1,11,0,10,10,0,11,0,0,9,8,0,8,12,
14,9,11,13,15,25
1972 DATA 12,0,16,17,0,0,0,11,0,0,0,12,0,33,0,13,0
,18,13,24,17,26,0,19,0,0,18,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0
1974 DATA 25,0,17,0,0,24,12,0,18,28,29,0,0,0,28,0,
26,0,0,27,0,0,30,26,0,0,0,29,0,0,32,30,41,0,0,31
1976 DATA 16,0,35,0,40,0,0,0,36,36,36,33,37,35,35,
35,36,36,38,36,36,36,36,39,40,36,36,36,36,34,36,36
,42,32,0,0
1978 DATA 43,41,0,0,44,42,0,0,0,43,0,0,45,49,46,48
,45,47,46,49,50,51,46,48,45,47,49,48,45,50,46,48,4
9,47,46,48
1980 DATA 47,52,0,0,51,0,0,0,0,0,2,0,0
1982 FOR I=1 TO NP:FOR J=0 TO 3:READ P:P(I,J)=P:NE
XT J:NEXT I
1990 DATA -1,0,0,0,30,0,22,10,0,10,0,10,24,0,34,10
,18,20,19,0,25,10,27,10,44,0,0,10,0,10,0,0,0,0,9,1
0,35,0,32,0,0,0
1992 FOR I=1 TO 21:READ A,B:OB(I,0)=A:OB(I,1)=B:NE
XT I:FOR I=22 TO 35:OB(I,0)=0:OB(I,1)=0:NEXT I
```



```
2000 DATA A long bow,A bronze key,A leather-bound
book,A silver arrow,A broken arrow,A gigantic sapp
hire
2001 DATA A vial of amber liquid,A golden sword,A
large Rembrandt painting,A whetstone,A set of silv
erware,A platinum pen
2002 DATA A velvet pillow,The Sorceror's sceptre,A
 gold bar,A vault in the wall,A dead knight,A ming
 vase
2003 DATA A sign saying: Diabolical Maze,A wicked
sorceror,A dead spider
2210 DATA BOW,KEY,BOO,ARR,BRO,SAP,LIQ,SWO,PAI,WHE,
SIL,PEN,PIL,SCE,BAR,VAU,KNI,VAS,SIG,SOR,SPI,NOR,SO
U,EAS,WES,N,S
2211 DATA E,W,DOO,WIN
2214 DATA GO,GET,LOO,INV,SCO,DRO,HEL,CLI,DIG,QUI,S
AY,DRI,OPE,TAK,KIL,REA,SHO,ATT,CLO,LOC,UNL,GIV,USE
,VAL,SWI
2215 DATA SHA,EXA,THR,FIN,JUM
2216 DATA North,South,East,West
2217 RETURN
20000 ? CHR$(125):SETCOLOR 4,1,6:SETCOLOR 2,0,0:SE
TCOLOR 1,1,14
20001 POKE 82,0:POKE 752,1
20002 ? "          CASTLEMAZE ADVENTURE"
20004 RETURN
30000 OPEN #1,4,1,"K:A.FRED":I=1
30010 CM$="":A$=""
30012 ? "WHAT NOW ? ";
30015 ? "*";CHR$(30);
30020 GET #1,X:Z=X
30021 IF Z>127 AND Z<>155 AND Z<>156 THEN Z=Z-128:
POKE 694,0
30022 IF Z<32 OR Z=127 THEN 30020
30024 IF Z>95 AND Z<126 THEN Z=Z-32:POKE 702,64
30025 IF Z=156 THEN CM$="":I=1:? CHR$(156);:GOTO 3
0012
30026 A$=CHR$(Z)
30028 ZL=LEN(CM$):IF ZL>26 THEN 30032
30030 IF Z<>155 AND Z<>126 THEN CM$(I,I)=A$:? A$;:
I=I+1:GOTO 30015
30032 IF Z=155 AND ZL>0 THEN ? " ":I=1:CLOSE #1:RE
TURN
30033 IF Z=126 AND ZL>1 THEN ? " ";A$;A$;:I=I-1:CM
$=CM$(1,I-1):GOTO 30015
30034 IF Z=126 AND ZL=1 THEN CM$="":I=1:? CHR$(156
);:GOTO 30012
30040 GOTO 30015
```

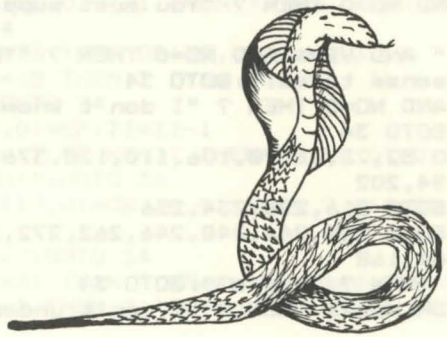
8

Tunnel Adventure

Another full-blown adventure, and again written in the same style as Castlemaze Adventure and Underground Adventure. This should serve to illustrate how easy it is to produce a large number of different games from the same basic rules.

This again is challenging, although it doesn't have the glossy edges of Underground. However it should keep you very busy trying to solve the many problems presented along the way.

Watch out for the vicious cat, and the evil hooded cobra, and the affectionate turtle encrusted with diamonds isn't all he seems either, in the ancient city of Kez!



```
0 GOSUB 700
1 DIM N1$(10),V1$(10),NT$(3),VT$(3)
2 DIM W1$(37),S4$(22),W2$(23),DR$(25),GB$(62),I1$(
25),W5$(64),WA$(64),SP$(30),KN$(32),IM$(21),WD$(16
)
3 DIM JA$(64),S1$(20),S2$(20),S3$(23),DD$(5),A$(1)
4 W1$="The panther sees the snake and flees.":S4$=
"You're not holding it.":W2$="You're out of matche
s."
5 DR$="It's very drafty in here.":GB$="A bird swoo
ps down out of the sky and  lands in front of you
."
6 I1$="You need a direct object.":W5$="The turtle
eats the carrot and rubs yourleg affectionately."
7 WA$="No way! I'm not paying for anything you mig
ht break.":SP$="You discover a secret passage."
8 KN$="There's something in his pocket.":IM$="That
's not possible.":WD$="It's pitch dark."
9 JA$="The javelin glides through the air as ifpul
led by magic.":S1$="I don't see it here."
10 S2$="Don't be ridiculous.":S3$="I don't know th
at word."
11 CP=39
24 GOTO 446
26 GOSUB 414
28 IF TG=1 THEN OB(29)=CP:IF CP=36 THEN TG=0
30 IF TG=1 THEN ? :? "The turtle is following you.
"
32 IF TG=1 AND CP=11 THEN GOSUB 148:? CHR$(125):?
"CAVE-IN!!":? :? :P(13,3)=0:P(9,1)=0:CP=13
34 GOSUB 390
35 IF BI>0 THEN BI=BI+1
36 IF MF=1 AND M2=0 THEN M2=1
38 IF VB>9 AND VB<>20 AND NO$="" THEN ? I1$:GOTO 3
4
39 IF VB=0 AND NO>0 THEN ? "You must supply a verb
.":GOTO 34
40 IF NO$<>"" AND VB=1 AND NO=0 THEN ? "That doesn
't make any sense to me.":GOTO 34
44 IF VB>10 AND NO=0 THEN ? "I don't know what a "
;N1$;" is!":GOTO 34
46 ON VB GOTO 52,72,26,98,106,110,128,376,136,72,1
54,172,174,194,202
48 ON VB-15 GOTO 216,226,234,236
50 ON VB-19 GOTO 238,246,248,246,262,272,274,296,3
04,226,356,364,168
52 IF NO$="" THEN ? "Where?":GOTO 34
54 IF NO>28 OR NO<21 THEN ? "I don't understand.":
GOTO 34
56 IF NO>24 THEN NO=NO-4
58 NO=NO-21
60 IF NO>-1 AND PD=1 THEN ? "You've fallen into a
```



```
pit.":GOTO 612
62 IF NO>-1 AND OB(30,0)=CP THEN 142
64 IF GF=0 AND CP=18 AND NO=1 THEN ? "You can't. T
he gate's locked.":GOTO 34
66 IF P(CP,NO)=0 AND CP<>1 THEN ? IM$:GOTO 34
68 IF P(CP,NO)=0 THEN ? "You can't go that way.":G
OTO 34
70 CP=P(CP,NO):GOTO 26
72 IF NO$="" THEN ? I1$:GOTO 34
74 IF OB(NO,0)=-1 THEN ? "You've already got it.":
GOTO 34
76 IF NO=0 THEN ? S3$:GOTO 34
78 IF CP=18 AND NO=31 THEN 88
80 IF NO<>37 OR CP<>29 THEN 86
82 IF OB(17,0)<>-1 THEN ? "You need a container.":
GOTO 34
84 OB(17,0)=0:GOTO 96
86 IF OB(NO,0)<>CP THEN ? S1$:GOTO 34
88 IF (NO>18 AND NO<32) OR NO>49 THEN ? "It's too
heavy.":GOTO 34
90 IF NO=12 THEN GOSUB 140
92 IF ZZ>3 THEN ? "Your hands are full.":GOTO 34
94 ZZ=ZZ+1
96 PRINT "Ok.":OB(NO,0)=-1:GOTO 34
98 ? "You are carrying:":ZZ=0
100 RESTORE 570:FOR I=1 TO LO:READ OB$:IF OB(I,0)=
-1 THEN ? OB$:ZZ=ZZ+1
102 NEXT I:IF ZZ=0 THEN ? "Nothing much."
104 GOTO 34
106 ? "Points are scored by leaving valuables  at
the mouth of the tunnel."
108 GOSUB 378:GOTO 34
110 IF NO=0 AND NO$<>"" THEN ? "What's a ";N1$;"?"
:GOTO 34
112 IF NO=0 THEN ? "Huh?":GOTO 34
114 IF OB(NO,0)<>-1 THEN ? "You aren't carrying it
.":GOTO 34
116 IF NO=35 THEN OB(35,0)=0:NO=32
118 IF NO=15 THEN ? "You can't.  It's stuck to you
r hand.":GOTO 34
120 OB(NO,0)=CP:ZZ=ZZ-1
122 IF NO=17 OR NO=37 THEN OB(NO,0)=0:OB(38)=CP:PR
INT "Crash!":GOTO 34
124 IF OB(12,0)=OB(30,0) THEN ? W1$:OB(30,0)=0:GOT
O 34
126 ? "Ok.":GOTO 34
128 IF CP=31 THEN ? "Read the medallion.":GOTO 34
130 IF CP=7 THEN ? "Try using prime numbers.":GOTO
 34
132 IF S=90 AND OB(41,0)=0 THEN ? "Some music woul
d be nice.":GOTO 34
134 ? "Just be patient and keep on examining  thi
```

```
ngs.":GOTO 34
136 IF CP=43 OR CP=44 THEN CP=87-CP:GOTO 26
138 PRINT IM$:GOTO 34
140 ? "The snake bites you.":BI=BI+8:RETURN
142 ? "The panther pounces on you.":GOTO 612
144 IF OB(32,0)>0 THEN RETURN
146 OB(32,0)=OB(35,0):OB(35,0)=0:RETURN
148 FOR I=1 TO OB(31,1):NEXT I:GOSUB 144:OB(31,1)=
100:RETURN
150 FOR I=1 TO 54:IF OB(I,0)=13 THEN OB(I,0)=55
152 NEXT I:RETURN
154 IF NO=31 AND CP=18 THEN 162
156 IF NO=31 THEN ? "I see no gate here.":GOTO 34
158 IF OB(NO,0)<>-1 THEN ? S4$:GOTO 34
160 ? "That's not necessary.":GOTO 34
162 IF GF=1 THEN ? "It's already open.":GOTO 34
164 IF OB(4,0)=-1 THEN GF=1:? "The gate swings slo
wly open.":GOTO 34
166 ? "You need a key to open the locked gate.":GO
TO 34
168 IF NO<>46 OR OB(46,0)<>-1 THEN 110
170 OB(46,0)=17:ZZ=ZZ-1:? JA$:GOTO 34
172 ? "Try 'PUSH'":GOTO 34
174 IF OB(NO,0)<>-1 THEN ? S4$:GOTO 34
176 IF NO<>2 AND NO<>16 AND NO<>18 AND NO<>5 THEN
? "There's nothing written on it.":GOTO 34
178 ? "It says:":IF NO<>2 THEN 186
180 ? "Will Haycraft, dazzling 'man about town'was
 heavily fined by magistrates today, following a s
ensational incident";
182 ? "          involving ...."
184 ? :? "The rest of the paper is torn, and you
can read no more.":GOTO 34
186 IF NO=18 THEN ? "Even  FELINES have enemies.":
GOTO 34
188 IF NO=16 THEN ? "Fermented juice can be very h
ealing.":GOTO 34
190 ? "Take the first six letters, throw away  the
 left half, double the middle, and   turn it aroun
d.":GOTO 34
194 IF OB(NO,0)<>-1 THEN ? S4$:GOTO 34
196 IF NO<>13 THEN ? S2$:GOTO 34
198 IF OB(13,0)<>-1 THEN 210
200 ZZ=ZZ-1:? "Yuk! It tastes horrible.":OB(13,0)=
0:GOTO 34
202 IF OB(NO,0)<>-1 AND OB(NO,0)<>CP AND NO<>29 TH
EN ? S1$:GOTO 34
204 IF NO<>12 AND NO<>29 AND NO<>30 AND NO<>41 THE
N 218
206 IF NO=30 THEN 142
208 IF NO=12 THEN GOSUB 140:GOTO 34
210 IF OB(29,0)<>CP AND NO=29 THEN ? "What turtle?
```



```
":GOTO 34
212 IF OB(13,0)<>-1 THEN ? "You have no food.":GOT
O 34
214 ZZ=ZZ-1:? W5$:OB(13,0)=0:TG=1:GOTO 34
216 IF OB(NO,0)<>-1 AND OB(NO,0)<>CP THEN ? S1$:GO
TO 34
218 IF NO<>12 AND NO<>41 AND NO<>29 AND NO<>30 THE
N ? "It isn't alive.":GOTO 34
220 IF NO=12 THEN GOSUB 140:GOTO 34
222 IF NO=30 THEN 142
224 ? "It's immortal.":GOTO 34
226 IF NO<>22 AND NO<>11 THEN 230
228 IF CP=21 AND P(21,1)=0 THEN P(21,1)=9:? "You've
 broken through!":GOTO 34
230 IF OB(12,0)=CP THEN GOSUB 140:GOTO 34
232 ? "Nothing happens.":GOTO 34
234 ? "You don't have enough charisma.":GOTO 34
236 ? "Try 'OPEN'":GOTO 34
238 IF NO<>51 THEN ? "It has no effect.":GOTO 34
240 IF CP<>22 THEN ? "What mirror?":GOTO 34
242 IF MI=1 THEN P(22,1)=4-P(22,1):? "It rolls eas
ily.":GOTO 34
244 ? "It's stuck.":GOTO 34
246 ? "Try 'USE'":GOTO 34
248 IF NO<>39 AND NO<>37 THEN ? "Try expressing th
at another way.":GOTO 34
250 IF OB(NO,0)<>-1 THEN ? S4$:GOTO 34
252 IF NO=39 THEN 258
254 IF CP<>22 THEN ? "There's no use for oil here.
":GOTO 34
256 MI=1:? "The rollers are now oiled.":GOTO 34
258 IF OB(15,0)+1 THEN ? "Your nails are nice and
clean now.":GOTO 34
260 OB(15,0)=CP:ZZ=ZZ-1:? "The statuette slips fro
m your grasp.":GOTO 34
262 IF NO<>36 THEN ? S2$:GOTO 34
264 IF OB(36,0)<>-1 THEN ? "You have no wine.":GOT
O 34
266 ? "Glug-glug-glug.":OB(36,0)=0:OB(17,0)=-1
268 IF BI THEN ? "Aaah... it cures the snakebite."
:BI=0
270 GOTO 34
272 ? WA$:GOTO 34
274 IF NO<32 OR NO>35 THEN ? IM$:GOTO 34
276 IF OB(33,0)<>-1 THEN ? "You don't have a match
.":GOTO 34
278 IF NO=33 THEN ? "The matches burn brightly.":Z
Z=ZZ-1:OB(33,0)=0:GOTO 34
280 IF NO<>34 THEN 290
282 IF OB(34,0)=-1 THEN GOSUB 148:? "You are blown
 to bits.":GOTO 612
284 IF OB(34,0)<>CP THEN ? S1$:GOTO 34
```

```
285 IF CP<>13 THEN ? "BOOOMMM!!!!":OB(34,0)=0
286 IF CP=13 THEN P(13,2)=24:CP=11:? "BOOOM!! You'
re through!":GOSUB 150
288 OB(34,0)=0:GOSUB 148:GOTO 34
290 IF OB(35,0) THEN ? "It's already lit.":GOTO 34
292 IF OB(32,0)=-1 THEN OB(32,0)=0:OB(35,0)=-1:PD=
0:GOTO 26
294 ? "You haven't got a torch.":GOTO 34
296 IF NO<>48 THEN ? "What?":GOTO 34
298 IF OB(48,0)<>-1 THEN ? S4$:GOTO 34
299 IF CP>35 AND (OB(41,0)>0 OR OB(41,0)=-1) THEN
? "Nothing happens.":GOTO 34
300 IF CP>35 THEN ? GB$:OB(41,0)=CP:GOTO 34
302 GOSUB 148:? CHR$(125):? "CAVE-IN!!":GOTO 612
304 IF NO<>11 THEN 314
306 IF CP=21 THEN ? "The south wall is badly erode
d.":GOTO 34
308 IF CP=17 AND P(17,3)=0 THEN P(17,3)=34:? SP$:G
OTO 34
310 IF CP=34 AND P(34,3)=0 THEN P(34,3)=35:P(35,2)
=34:? SP$:GOTO 34
312 ? "You find nothing special.":GOTO 34
314 IF NO=31 AND CP=18 THEN 320
316 IF NO=37 AND CP=29 THEN ? "It's just some oil.
":GOTO 34
318 IF OB(NO,0)<>CP AND OB(NO,0)<>-1 THEN ? S1$:GO
TO 34
320 IF NO=2 OR NO=16 OR NO=18 THEN 174
322 IF NO=13 THEN ? "It's not fit for human consum
ption.":GOTO 34
324 IF NO=40 THEN ? "It's topaz.":GOTO 34
326 IF NO=9 THEN ? "It's malachite.":GOTO 34
328 IF NO=41 THEN ? "It's made of gold!":GOTO 34
330 IF NO=10 THEN ? "It's lapis lazuli.":GOTO 34
332 IF NO=42 THEN ? "It's pyrite.":GOTO 34
334 IF NO=12 THEN GOSUB 140:GOTO 34
336 IF NO=30 THEN 142
338 IF NO=1 THEN ? "It's embroidered with gold thr
ead.":GOTO 34
340 IF NO=15 THEN ? "It glistens.":GOTO 34
342 IF NO=46 AND OB(46,0)=17 THEN ? "It's pointing
 towards the west.":GOTO 34
344 IF NO=50 THEN ? "It contains sacred oil.":GOTO
 34
346 IF NO=51 THEN ? "It's on rollers.":GOTO 34
348 IF NO=52 THEN 366
350 IF NO=20 AND OB(33,0)=0 THEN ? KN$:OB(33,0)=43
:GOTO 34
352 IF NO=20 AND OB(4,0)=0 THEN ? KN$:OB(4,0)=43:G
OTO 34
354 ? "You see nothing special.":GOTO 34
356 IF NO<>34 THEN ? "I don't know how to do that.
```



```
":GOTO 34
358 IF OB(3,0)=-1 AND OB(6,0)=-1 AND OB(14,0)=-1 T
HEN 362
360 ? "You're not holding all the ingredients.":GO
TO 34
362 OB(3,0)=0:OB(6,0)=0:OB(14,0)=0:OB(34,0)=-1:ZZ=
ZZ-2:? "Consider it done.":GOTO 34
364 ? "Try 'MAKE'.":GOTO 34
366 CO=1:CO=1:GOSUB 30000:CO=0:NO=VAL(CM$):IF NO=0
 OR NO>100 THEN 34
368 IF NO=13 AND OB(49,0)=0 THEN OB(49,0)=7:GOTO 3
74
370 IF NO=71 AND OB(8,0)=0 THEN OB(8,0)=7:GOTO 374
372 ? :? "That compartment is empty.":GOTO 34
374 ? :? "Something fell out.":GOTO 34
376 GOSUB 378:GOTO 387
378 S=0:FOR I=1 TO LO:IF OB(I,0)=36 THEN S=S+OB(I,
1)
382 NEXT I
384 ? "You have scored ";S;" points out of 100.":I
F S<99 THEN RETURN
386 PRINT :? "WELL DONE!":END
387 END
390 ? :GOSUB 30000:N1$="":NO$="":VB$="":VB=0:NO=0:
H=0:CM=LEN(CM$):IF CM=1 THEN VB=1:NO$=CM$:GOTO 410
391 INC=0:FOR I=1 TO CM:IF CM$(I,I)=" " THEN INC=I
NC+1:IF INC=1 THEN H=I-1
392 NEXT I:IF H=1 THEN VB=1:RETURN
393 IF H=2 AND CM>3 THEN NO$=CM$(4,CM):VB=1:GOTO 4
10
396 IF H=0 THEN H=CM
397 IF H=2 THEN VB=1:RETURN
398 V1$=CM$(1,H)
399 VB$=V1$(1,3)
400 RESTORE 602:FOR J=1 TO NV:READ VT$:IF VT$=VB$
THEN VB=J:IF J<>NV THEN J=NV
401 NEXT J:IF VB>0 THEN 406
404 VB=1:N1$=V1$:GOTO 409
406 IF H+2>CM THEN RETURN
408 N1$=CM$(H+2,CM):IF LEN(N1$)<3 THEN NO=0:? :RET
URN
409 NO$=N1$(1,3)
410 RESTORE 596:FOR I=1 TO NN:READ NT$:IF NT$=NO$
THEN NO=I
411 NEXT I:IF NO=35 AND (OB(32,0)=-1 OR OB(32,0)=C
P) THEN NO=32
412 ? :RETURN
414 ? CHR$(125):IF CP=16 THEN T=T+1:IF T>2 THEN ?
DR$:IF T>3 AND RND(1)<T*0.1 THEN GOSUB 144:T=0
416 IF OB(35,0)<>-1 AND CP<35 THEN ? WD$:PD=1:RETU
RN
417 IF P(13,2)=24 AND CP=13 THEN GOSUB 464:PD=0:GO
```

```
TO 420
418 GOSUB 446+CP*2:PD=0
420 ? :? "You can see:"
422 OB=0:RESTORE 570:FOR I=1 TO LO:READ OB$:IF OB(
I,0)=CP THEN ? OB$:OB=1
424 NEXT I:IF OB=0 THEN ? "Nothing much."
426 FL=0
428 ? :? "You can go:":RESTORE 608:FOR I=0 TO 3:RE
AD DD$:IF P(CP,I)<>0 THEN ? DD$;" ":FL=1
430 NEXT I:IF FL=0 THEN ? "Nowhere fast."
432 ? :?
434 IF BI>12 THEN ? ">The bite's throbbing<"
436 IF BI>23 THEN ? ">You're getting dizzy<"
438 IF BI>34 THEN ? ">It's hard to breathe<":IF BI
>42 THEN 612
440 IF CP<>18 THEN RETURN
442 IF GF=1 THEN ? :? "The gate is open.":RETURN
444 ? :? "The gate in the grill is locked.":RETURN

446 NN=54:NV=32:P=51:LO=54:DIM P(P,3),OB(LO,1),OB$
(36),VB$(3),NO$(3),CM$(27):GOTO 558
448 ? "You're in a storeroom, whose walls are  mad
e out of heavy concrete.":RETURN
450 ? "You're walking along a dusty old passageway
 that obviously hasn't been cleaned  for years.":R
ETURN
452 ? "You're in the sacred quarters of       Pri
ncess Anka. Tread very carefully on  this hallowed
 ground.":RETURN
454 ? "You're in the king's harem, but alas andala
ck (or maybe not), none of the harem are present a
t the moment.":RETURN
456 ? "You're in a storeroom with concrete     wal
ls.  You experience a strange sense  of deja vu.":
RETURN
458 ? "You're in a twisty little tunnel, and   ves
 it's one of THOSE twisty little     tunnels.":RET
URN
460 ? "You're wandering about a small niche in the
 room that was once used to store    jewelry.":RET
URN
462 ? "You're in a studio once used by the     kin
g's artists, to produce glowing (and false) portra
its of the king."
463 RETURN
464 ? "You're crawling over a jumble of broken roc
k.  The ceiling is very low here.":RETURN
466 GOTO 458
468 ? "You're crawling along a low tunnel, withhem
med in walls.":RETURN
470 ? "You're in the middle of an ancient      lib
rary, with ancient books dotted aboutthe walls.":R
```



```
ETURN
472 ? "You're in a low tunnel, but the path    ahe
ad is blocked by a thick brick wall.":RETURN
474 GOTO 458
476 GOTO 458
478 GOTO 458
480 ? "You're inside an old closet used to     sto
re the royal wine centuries ago.":RETURN
482 ? "You're in a long low tunnel, but the wayahe
ad is barred by a metal grill.":RETURN
484 GOTO 458
486 GOTO 458
488 ? "You're at one end of an old storeroom,  now
 disused and covered with tatty old  cobwebs.":RET
URN
490 ? "You're in King Kaleb's bedroom, but     luc
kily for you King Kaleb himself      isn't.":RETUR
N
492 ? "You're in the old slaves' quarters. The cra
mped conditions and sorry state of   the room tell
s you a lot about";
493 ? " the old  King.":RETURN
494 ? "You're at the west end of a temple, and the
re's an ugly hole in the west wall   (naughty old
you).":RETURN
496 ? "You're at the east end of the temple,   whi
ch has an air of faded magnificence  to it.":RETUR
N
498 ? "You're in the old warrior's quarters,   whi
ch look much better than those that  the poor slav
es had to suffer."
499 RETURN
500 ? "You're in a stable, and although your   eye
s may have told you otherwise, your  nose tells th
e truth.":RETURN
502 ? "You're in the high priest's vestry,     whe
re presumably he used to keep his    high priests
vests.":RETURN
504 ? "You're at the sacred shrine of ISIS.    The
 walls tell a sorry tale of disrepairand lost sple
ndour.":RETURN
506 ? "You're in what was once used as the     kit
chen, although the state it is now inmakes you won
der.":RETURN
508 ? "You're in a royal antechamber in the    dep
ths of the palace. It looks old and  desolate.":RE
TURN
510 ? "You're in the throne room (no, not that sor
t of throne).":RETURN
512 ? "You're walking along a long dusty       pas
sageway, brushing aside cobwebs,     muck and grim
e.":RETURN
```

```
514 ? "You're in a secret compartment, which   has
n't seen the light of day for many   a long year."
:RETURN
516 ? "You're on a long, low tunnel, with a   hin
t of daylight to the north.":RETURN
518 ? "You're at the mouth of the tunnel. The   pat
h leading out opens up here,to reveala road to the
 north.":RETURN
520 ? "You're at the end of the road, and it   loo
ks as if the road goes on forever.":RETURN
522 ? "You're deep in a dense, dark forest.":RETUR
N
524 ? "You're on a familiar old path made by   hor
ses.  You seem to have seen this   somewhere bef
ore.":RETURN
526 ? "You're at the end of a path, with forestsur
rounding you in all directions.":RETURN
528 ? "You're in the deep, dark forest,althoughthe
re does appear to be some light to   the south.":R
ETURN
530 ? "You're in the middle of a familiar   cle
aring, with a bridge to the south.":RETURN
532 ? "You're on the north side of the old   bri
dge.":RETURN
534 ? "You're on the southern side of the   bri
dge, with a path heading south.":RETURN
536 ? "You're at the great crossroads, with   roa
ds heading off in all directions.":RETURN
538 ? "You're on the great western road, and itsur
e looks long and dusty.":RETURN
540 ? "You're on the great eastern road, and   thi
s seems to head off into the distancefor ever.":RE
TURN
542 ? "You're on the great southern road,   get
ting nowhere fast.":RETURN
544 GOTO 542
546 GOTO 542
548 GOTO 542
549 DATA 18,0,5,0,25,33,8,12,0,7,31,0,22,0,0,0,0,0
,21,1,6,15,6,19,3,0,0,0,0,0,0,2,21,10,0,0,9,6,6,6,
0,20,13,0,0,0,2,0
550 DATA 0,0,0,11,19,19,20,19,6,6,6,10,14,6,6,6,0,
0,30,0,35,1,0,0,6,16,6,6,11,16,16,16,0,0,0,5,0,0,0
,31,26,27,0,0
552 DATA 28,0,25,13,29,2,0,24,0,23,0,33,23,0,0,0,0
,24,0,0,0,25,0,0,0,0,33,17,33,32,22,3,31,0,0,0,2,3
1,26,30,0,0,17,0
554 DATA 36,18,0,0,37,35,0,0,51,36,0,0,38,39,38,38
,38,40,38,38,39,41,38,38,40,42,38,38,41,43,0,0,42,
0,0,0,0,45,0,0
556 DATA 44,48,47,46,0,0,45,46,0,0,47,45,45,49,0,0
,45,50,0,0,45,51,0,0,45,37,0,0
```



```
558 FOR I=1 TO P:FOR J=0 TO 3:READ M:P(I,J)=M:NEXT
 J:NEXT I
560 DATA 28,10,37,0,1,0,0,10,28,0,5,0,4,10,0,10,13
,0,18,10,0,0,36,0,30,0,21,0,8,10,12,0,0,0,34,10,32
,0,43,0,0,0
561 DATA 0,0,0,0,0,0,0,0
562 DATA 0,0,0,0,0,0,27,10,31,0,0,20,35,0,0,0,0,0,
0,0,17,0,0,0,0,0,3,0,11,10,0,10,13,0,30,0,8,0,23,0
,26,0,27,0,32,0
564 DATA 0,0,29,0,22,0,7,0,22,0,3,0
566 FOR I=1 TO 54:READ A,B:OB(I,0)=A:OB(I,1)=B:NEX
T I
570 DATA An ephod,A scrap of newspaper,A keg of ch
arcoal,A silver key,A parchment scroll,A keg of sa
ltpetre
572 DATA A platinum chastity belt,A ruby earring,A
 green pebble,A blue stone,a,A vicious cobra,A shr
iveled carrot
574 DATA A keg of sulphur,A jade statuette,An old
medical book,An empty bottle,A gold medallion,A so
lid gold throne
576 DATA A dead knight,a,b,c,d,e,f,g,h,A giant tur
tle covered with diamonds,A hungry panther,A gate,
An old torch
578 DATA Some matches,Three kegs of gunpowder,A sh
ining torch,A bottle of wine,A bottle of oil,Some
broken glass
580 DATA A jar of nail polish remover,Some brown a
nd pink gravel,A bird,A gold nugget,A wooden spoon
,A block of marble
582 DATA A set of manacles,A rusty javelin,Straw a
nd dung,A brass clarion,A satin ribbon,A marble fo
nt,A huge mirror
584 DATA 100 small compartments,A king-sized bed,A
 triclinium
596 DATA EPH,NEW,CHA,KEY,SCR,SAL,BEL,EAR,PEB,STO,W
AL,COB,CAR,SUL,STA,BOO,BOT,MED,THR,KNI,NOR,SOU,EAS
,WES,N,S,E,W
598 DATA TUR,PAN,GAT,TOR,MAT,GUN,TOR,WIN,OIL,GLA,R
EM,GRA,BIR,NUG,SPO,BLO,MAN,JAV,STR,CLA,RIB,FON,MIR
,COM,BED,TRI
602 DATA GO,GET,LOO,INV,SCO,DRO,HEL,QUI,CRO,TAK,OP
E,MOV,REA,EAT,FEE,KIL,HIT,CHA,UNL,PUS,REM,USE,OIL,
DRI,BRE,LIG
604 DATA PLA,EXA,KIC,MAK,MIX,THR
608 DATA North,South,East,West
610 GOTO 26
612 ? :? "You are dead!!":GOTO 376
700 POKE 752,1:POKE 82,0:? CHR$(125):SETCOLOR 4,1,
6:SETCOLOR 2,1,14:SETCOLOR 1,0,0
702 ? "             TUNNEL ADVENTURE"
704 RETURN
```

```
30000 OPEN #1,4,1,"K:A.FRED"
30010 CM$="":A$="":I=1
30011 IF CO=1 THEN ? "Which number ? ";:GOTO 30015
30012 ? "WHAT NOW ? ";
30015 ? "*";CHR$(30);
30020 GET #1,X:Z=X
30021 IF Z>127 AND Z<>155 AND Z<>156 THEN Z=Z-128:
POKE 694,0
30022 IF Z<32 OR Z=127 THEN 30020
30024 IF Z>95 AND Z<126 THEN Z=Z-32:POKE 702,64
30025 IF Z=156 THEN CM$="":I=1:? CHR$(156);:GOTO 3
0011
30026 A$=CHR$(Z)
30028 ZL=LEN(CM$):IF ZL>26 THEN 30032
30030 IF Z<>155 AND Z<>126 THEN CM$(I,I)=A$:? A$;:
I=I+1:GOTO 30015
30032 IF Z=155 AND ZL>0 THEN ? " ":I=1:CLOSE #1:RE
TURN
30033 IF Z=126 AND ZL>1 THEN ? " ";A$;A$;:I=I-1:CM
$=CM$(1,I-1):GOTO 30015
30034 IF Z=126 AND ZL=1 THEN CM$="":I=1:? CHR$(156
);:GOTO 30011
30040 GOTO 30015
```


9

Further Information

Introduction

We've presented you with information on various adventures from both the U.K. and the U.S.A. over the pages of this book, but most of the games mentioned so far have been fairly old, in that they go back as far as some of the earliest microcomputers like the Apple and the Commodore PET.

In this last section we'd like to round off by going through a few currently available adventures for various microcomputers that are relatively recent, at least at the time of writing.

Some are classics, some are obviously destined to be so, and some will probably fade over the years into a delightful obscurity and never be heard of again.

The rest of this chapter will give you some useful information on where to find out more about adventures generally, as well as listing a number of popular newstand magazines that do sometimes carry features about this sort of game.

Finally, a few useful names and addresses, and especially for those of you who own Commodore kit and want to acquire a copy of the legendary Adventure by Crowther and Woods that has featured prominently in this book, the name and address of the person to contact at the Independent Commodore Products Users' Group.

For owners of other machines, it's worth asking around to see if a copy exists for your particular machine, but if you haven't got disk

drives, forget it! This game relies almost entirely on a disk-based mode of operation, and would require an awful lot of memory before it would function on a micro that was sans disks.

That's all for now, except to say thanks to a few people. Obviously Crowther and Woods, but also Jim Butterfield, for producing the orginal PET version, and to Steve Darnold, for inadvertently getting me started on this whole adventure writing lark in the first place, and who provided the original code for Castlemaze Adventure and Tunnel Adventure.

Current Adventure Games

All the names and addresses of the companies involved can be found in most of the current popular magazines, as most of them seem to advertise quite extensively.

If not, a copy of *Personal Computer News,* the (at the moment!) 50 pence weekly, has a tri-weekly round up of software available, and covers most of the adventure games around.

So, to get the ball rolling, how about The Hobbit, which must rank as one of the classic modern games of adventure, which is available from Melbourne House for the 48K Spectrum.

A complete solving of this would take a very long time indeed, and I've yet to hear of anyone who has actually solved the entire thing. A nice style of entering your commands here as well.

PIMania seems to be the other game currently 'in vogue' as it were, although I think I'd like it a lot better if it wasn't for the inept advertising by the company who handle it, namely Automata UK. Are they really trying to produce the worst advertising in the microcomputer industry?!

Still, at least the game is good, and has the virtue of working on the Spectrum, Dragon and BBC.

Sphinx, for the BBC model B, from John Wiley and Sons is also quite a good, classical adventure, involving all the usual thud and blunder techniques beloved by writers of this particular type of adventure.

John Wiley also do a few more for the model B as well, so they're worth checking out if you're tuned into Auntie Beeb.

Microdeal have inevitably produced a series of adventures for the Dragon, including Escape, Flipper, and Mansion Adventure, or at least they call them adventure games. Personally the only one I thought was of lasting interest was the Mansion Adventure, but then we all have our different tastes.

For the Commodore 64, well, Romik have produced a couple of games, and modesty prevents me from telling you how wonderful they are, but I would like to thank Kevin Bergin for some last minute programming on those!

And the Vic 20 ? Well, there are always the cartridge versions of the Scott Adams games, and Kayde Electronics have produced the Swamp (...In the Swamp, no one can hear you scream..., runs the advertising. Yawn...), although it, not suprisingly, requires a minimum of 16K expansion.

Those are just some, but any periodical should give you details of many more.

Of course there is also now a magazine devoted to adventure games, called Micro Adventurer. Well worth a look.

More Information

Strangely enough, the general magazines don't appear to have picked up too strongly on this resurgence of interest in adventures, although *Personal Computer News* regularly carries a number of reviews for all kinds of machines, and most of the others mention them every now and again.

However, there are three classic issues of old magazines which the serious adventure freak must have.

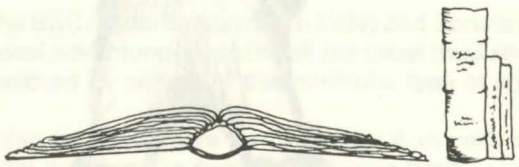
The December 1980 issue of *Byte* magazine, the one Daley Thompson does weight training with, is mainly devoted to adventuring, and features a whole host of excellent articles by many of the top authors around at the time, including Scott Adams, P. Lebling, Bob Liddell, and many more. A great issue, if you can dig it out.

The other two are different issues of the same magazine, but finding them is not going to be easy.

The magazine in question is *Creative Computing*, and the first major article appeared in August 1979, when the data structure behind the Scott Adams series of adventures was explained in full. This has inspired a number of people to begin writing their own adventures, including David Malmberg, who went on to write the very good Castle Adventure (the one with the sleepy piranha in it that I mentioned earlier!).

July 1980 was another good issue, including the article that explained the working of the program Zork, in the excellent 'How to fit a large program into a small computer'.

All required reading for the serious adventure fan, but keep your eyes on the newstands for other, newer issues of magazines.



Who to Contact

User Groups are the people to contact, and the following covers most of the popular makes of home computers.

Atari:
UK Atari Computer Owners Club
P.O. Box 3
Rayleigh
Essex

BBC:
Laserbug
Paul Barbour
10 Dawley Ride
Colnbrook
Slough
Berkshire

or
Beebug
Sheridan Williams/David Graham
P.O. Box 50
St. Albans
Hertfordshire

Commodore:
ICPUG
Jack Cohen
Riverhead
154 Chesterfield Drive
Sevenoaks
Kent

Spectrum:
Sinclair User Group
Irving Brand
Polytechnic of North London
Holloway Road
London N7

Writing to the appropriate address for your machine should produce the desired response.

EXPLORING ADVENTURES ON THE ATARI 48K

The three adventures in this book are available on a cassette at £7.95, from all good computer stores and bookshops, or in case of difficulty, direct from the publisher.

Send your cheque/ postal order to:
Gerald Duckworth & Co Ltd
The Old Piano Factory
43 Gloucester Crescent
London NW1

and they will be sent to you post-free

Index

This index usually only shows the first appearance of a subject in the book, but if a second (and subsequent) entry is important, it is also noted down.

Adams, Scott : 4, 5, 6, 28, 29, 30
Adventure : 1, 4, 21
Asc command : 63
Attack verb : 170
Bears : 100
Bottles : 105
Break verb : 194
Butterfield, Jim : 4, 9
Castlemaze adventure : 215
Chop verb : 164
Chr$ command : 63
Climb verb : 166
Close verb : 152
Contents : v
Creating adventures : 115
Crevices : 109
Cross verb : 148
Crowther, Willie : 3, 8, 10, 21
Cut verb : 164
Data command : 54
Data validation : 102
Death! : 104
Dialogue : 43, 44, 45
Dim command : 71
Drink verb : 158
Drop verb : 144
Dungeons and Dragons : 35
Eat verb : 154
Examine verb : 192
Feed verb : 156
For command : 64
Gargoyle : 106, 107
Get verb : 140
Go verb : 138
Gosub command : 67
Goto command : 67
Hassett, Greg : 6
Hazards : 83, 84
Help verb : 203
Hit verb : 174
If command : 56
Input command : 51, 53
Input subroutines : 111, 112, 113
Int command : 71
Introduction : vii
Inventories : 142
Jump verb : 194
Kill verb : 172
Left$ command : 60
Len command : 58
Light verb : 168
Load verb : 200
Logical Operators : 57
London adventures : 117
Look verb : 202
Lord of the Rings : 7, 10
Make verb : 176
Map drawing : 17, 20, 85, 87
Mazes : 93
Mid$ command : 59
Movement : 74, 76, 77
Murder adventures : 128
Next command : 64
Noun data : 212
Objects : 42
Obstacles : 83
Offer verb : 160
Oil verb : 180
On command : 69
Open verb : 150
Panthers : 109
Personal Computer News : 242
Philosopher's Quest : 14
Pirate Adventure : 13, 28
Popular Computing Weekly : 2
Problem solving : 83, 98
Push verb : 196
Quit verb : 146
Read verb : 190
Reflect verb : 178
Restore command : 55
Return command : 67
Right$ command : 61
Rnd command : 70

Rub verb : 188
Save verb : 198
Score verb : 202
Screen design : 51
Screen Responses : 95
Solving adventures : 16
Space adventures : 122
Spray verb : 184
Stab verb : 182
Storylines : 80, 81, 82
String commands : 58, 59, 60
Str$ command : 61
Subroutines : 57
Take verb : 203
Temple of Apshai : 11
Then command : 56
Throw verb : 186
Torches : 105
Traditional adventures : 131
Tunnel adventure : 227
Underground Adventure : 36
Underground variables : 74
Underground verbs : 100, 137
Underground data : 205
User Groups : 243
Val command : 61
Variables : 52
Verb data : 212
Verbs : 42, 92
Vocabulary : 34
Wave verb : 162
Western adventures : 125
Woods, Don : 3, 8, 10, 21
Zork : 3, 9, 31, 32

DUCKWORTH
HOME COMPUTING

THE ADVENTURER'S COMPANION
Mike and Peter Gerrard

How do I survive the pale bulbous eyes? How do I get past the troll? Where is the Pirate's Chest? How do I escape the Goblin's Dungeon?

In response to these and hundreds of other questions sent in to magazines by frustrated adventurers, here is a complete guide to playing four of the most popular adventures on home micros today: *The Hobbit, Colossal Cave Adventure, Adventureland* and *Pirate Adventure*. The book provides a solution to every problem you will meet, and is designed to enable you to look up the answer without giving away anything of the rest of the adventure. It also includes complete maps for all four adventures. £3.95

THE ADVENTURER'S NOTEBOOK
Mike Gerrard

This book is for both beginners and regular adventure players. It explains what an adventure game is, gives a history of adventure games, includes hints on how to play games more successfully and a list of recommended adventures. The main part of the book consists of a series of maps with space for your notes on verbs, nouns, locations, how to pass obstacles – everything the keen adventurer needs in order to keep all those scribbled sheets and notes together in one book. £3.95

Mike and Peter Gerrard are regular contributors to *Which Micro?* and *Personal Computer News*. Peter Gerrard is the author of many titles in the Duckworth Home Computing list, including the Exploring Adventures series, and contributes to *Popular Computing Weekly, Commodore Horizons* and *Micro Adventurer*.

DUCKWORTH
The Old Piano Factory, 43 Gloucester Crescent, London NW1 7DY
Tel: 01-485 3484

Duckworth Home Computing

EXPLORING ADVENTURES ON THE ATARI 48K
by Peter Gerrard

This is a complete look at the fabulous world of Adventure Games for the Atari Micro. Starting with an introduction to adventures, and their early history, it takes you gently through the basic programming necessary on the Atari before you can start writing your own games.

Inputting information, room mapping, movement, vocabulary – everything required to write an adventure game is explored in detail. There follow a number of adventure scenarios, just to get you started, and finally three complete listings written specially for the Atari, which will send you off into wonderful worlds where almost anything can happen. The three games listed in this book are available on one cassette.

Peter Gerrard, former editor of *Commodore Computing International*, is the author of two top-selling adventure games for the Commodore 64 and a regular contributor to *Personal Computer News, Which Micro? and Software Review* and *Commodore Horizons*.

ISBN 0-7156-1924-1

9 780715 619247

Duckworth
The Old Piano Factory
43 Gloucester Crescent, London NW1

ISBN 0 7156 1924 1

IN UK ONLY £6.95 NET