# atari adventures

## a guide to playing and writing adventures

tony bridge

# atari adventures

## tony bridge

## the eye of the star warrior

gary radburn and tony bridge

from an idea by roy carnell

First published 1984 by:
Sunshine Books (an imprint of Scot Press Ltd.)
12–13 Little Newport Street,
London WC2R 3LD



ISBN 0 946408 18 1



Cover design by Graphic Design Ltd.
Illustration by Stuart Hughes.
Typeset and printed in England by The Leagrave Press Ltd.

# CONTENTS

*For Jilly, again*

*Acknowledgements:* Many thanks to Gary and John Radburn, Kikey of Ace Software, Southall, for his invaluable assistance and Silica Shop of Sidcup for supplying alternate realities — and Sascha, Kerry and Lucy for cups of tea!

# Contents in detail

# Introduction

Adventuring has become more than a pastime for bored computer programmers to play illicitly after hours, using million-dollar mainframes, and very expensive computer time, to fight dragons and snakes. It has become a big business and, with the present explosion of home computer sales, has won the hearts of millions of computer hobbyists around the world.

Over the last few months, indeed, while writing this book, I have noticed a distinct turn away from the arcade games toward adventure games.

Why is this? I believe that we have reached a plateau with arcade games. We started, back in the Pet/Nascom days, with very expensive hardware running Space Invaders. The Space Invaders were Xs and Ws, but the lack of graphics didn't detract from the game. It was enough to see even this pale imitation of the pub classic (as well as not having to find 10p each time we wanted to play — we conveniently forgot about the £800 to buy the machine!).

This point was the jungle floor, at the foot of towering cliffs. Space Invaders held sway for years, while we thrashed around, hacking away at a thick foliage of assemblers, disassemblers and machine code, and hardware prices. Eventually we were led out into the light by one or two intrepid souls such as Clive Sinclair in the UK, who finally brought the computer within the reach of teenagers. These teenagers had the necessary dedication to video games required to start that long climb up the cliff face to the dizzy heights.

The first toeholds on the way were the Space Invader clones such as Galaxians, Gorf and so on — then came *Pacman*, and a million versions and variants. Each toehold carved by the pioneers made it easier for the swelling numbers of games programmers behind them to scale the cliffs.

The Pacman, Kong and Invader routes up the cliff remain an invaluable aid to budding programmers, and indeed on the plateau we have now reached, there are little kingdoms for each of these genres, which have their temporary kings and which will increase their boundaries unhindered.

There are also highly original games, and these are like major cities — archaeologists will one day spend lifetimes digging through the many

layers, delighting in the wonders they find. Some of these cities will prosper and one or two lucky ones will, in the future, grow into kingdoms themselves and make fabulous wealth for all their inhabitants.

A plateau it must remain though, however high in the sky it may be. At the present level of computer technology, it is not possible to do more than invent new ways of zapping aliens and run round mazes, or climb ladders.

As each new machine is announced, it introduces new ways of making it easier for us to program effects into our arcade games; but those effects remain broadly the same. Several attempts have been made at new techniques — at the time of writing there is a certain interest in 3D games — but in the end, arcade games can only take a leap forward with more realism.

Realism is what players are looking for in an arcade, shoot'em up game. The danger, no matter that it is vicarious, must appear to be real, the adrenalin must flow. And the hardware, as it presently stands, has just about gone as far as possible in creating a lifelike arena in which the player can act out his or her fantasies.

Arcade games have an average life of only a few months — the smash hits, of course, survive much longer, but the game business is a hotbed of invention. Machines are with us now that use video discs to create a lifelike environment for the player, and networks are now also making an appearance, in which two or more players, at separate machines, can play against each other. In the near future, we can expect holography to be used. After that, who knows? Maybe there will one day be some kind of tactile input — smell perhaps, or some sort of touch sensation: after all, sight and sound both make a huge impact in the present arcade games — what is left?

Most of these techniques, with the possible exception of networking, which would seem to be made for microcomputers, will be far too expensive to make it worthwhile implementing them on home computers.

The next step towards realism in games for home computers, can only be some sort of mental link between the machine and the human mind.

That is obviously looking far into the future and, thrilling though it may be to contemplate, brain-to-brain contact must bring with it other things too dark to contemplate.

Is there another area of computer games in which the imagination can be allowed to run riot? There is, and you will no doubt have guessed what it is. Adventures, of course!

Adventure games have progressed in much the same way as arcade games — a long time spent wandering around the jungle until the present generation of home microcomputers dragged them into the light. Since then, because their scenarios exist largely in the collective

imaginations of the author and player, adventure programs have not yet reached that plateau on which arcade games now reside.

That is not to say you will find no mention of arcade games within this volume. Far from it! I love arcade games, but, suffering from advanced age, they usually become too frenetic for my tired old fingers. Although the elusive high-score becomes a fatal attraction, I prefer, in the end, to have my brain exercised even in an arcade game. This is why I make no apology for including such programs as *Jumpman*, which on the surface, would appear to be pure arcade games. I would rather call *Jumpman*, and several other similar games, *Puzzle Arcventures*. The relevant sections will explain further — suffice to say here that I do not want to be bound to purely traditional text adventures in this overview.

What makes a good adventure?

The original adventure, *Colossal Cave*, was written for a mainframe computer. These computers did not, in those days, have VDUs (visual display units); instead, hard copy was taken on a printer linked to the computer. So the format of the game was, text dumped to the printer.

The mythology has thus grown up, among computer adventurers, that text is best. This is not so!

Not necessarily. High-resolution graphics take up an enormous amount of memory, and this memory could be better employed, possibly, in describing more locations and more objects. I like to see a pretty picture of the scene when I stumble across a new location, but also like to be offered an opportunity to clear the display on subsequent visits. Both the American Scott Adams *Graphic Adventures*, and the English Channel 8 *Mysterious Adventures*, by Brian Howarth, offer the facility of seeing the graphics or retaining pure text, which must surely be the best compromise.

Graphics play a supreme role in arcventures, of course, and we shall be examining these later. Sound, too, is usually present in arcventures, and indeed is an indispensable part of the overall picture — in text adventures, sound does not play an important part, although some adventures include it.

How are adventures played? We shall look at each type in detail, but for now, a typical text adventure will suffice to give us the flavour of an adventure.

There are two ways in which the vast majority of adventures are conducted. The first way is for the player to instruct the computer in what to do. This is called The Puppet Method, in which the computer acts as the player's puppet. Thus, a typical exchange will be:

"I am in a Graveyard. All around me are old Tombstones, partly obscured by a low creeping mist. I can see an eerie figure in the shadows. Obvious exits are North and West. What do I do now?'

The required answer may be, in the first instance, GO NORTH, although this may usually, in any program worth its salt, be shortened to just NORTH, or even N. This is the most basic response — there are many others that will be required, depending on the circumstances prevailing at the time. The player may, for instance, have picked up a sword at some earlier stage, and may then instruct the computer to use this to attack the figure.

The other main subset of text adventures is the First Party Adventure. On the surface, this may look very much the same as the Puppet adventure, but in fact, the differences, though subtle, are major.

In this type, the 'I' of the first example is replaced by 'You' — so in the example we have just seen, the computer will say 'You are in a graveyard. All around you are eerie tombstones.' And so on, until the final question, 'What do you do now?' This form of adventure often allows for a rather more detailed response to the computer's questions, and more realistic reactions to your prompts.

Little equipment is needed to play adventures — unlike Dungeons and Dragons, which is so often a major part of adventure, no dice are needed, and no little model figures. Neither is there a need for reams upon reams of rule books and tables, although individual adventures may well include complex scenarios and rules, fantasy gamers being what they are! However, the very attraction of computer adventures is that they can be played, solo, by anyone — the only equipment needed by any adventurer is a large piece of paper and a pencil, and maybe a supply of coffee.

The paper and pencil will be used to map the area covered by the adventure. As the player moves around the game, going north, then east, then north and south again, solving problems along the way, his sense of direction will soon become confused — and the position of the adventurer should be known at all times as the objects found along the way will often have to be left until later. Most programs set a limit upon the number of items which can be carried at any one time by the player, and he will need to be able to return later to the location where he has left the object, in order to pick it up when possible.

Quantities of coffee will be needed to battle sleepiness as the dawn approaches!

The scenario of each adventure will differ. The original, *Colossal Cave*, was set underground, as was the original *Dungeons and Dragons*. As with D & D, computer adventures have, since then, taken to space, the wild west, ancient Rome, ancient Egypt, Elizabethan times, and so on. In fact, the number of scenarios available is limitless. We shall see many of these imaginary worlds as we progress through our look at Atari adventures.

Adventure is not, however, a simple matter of reading maps, or even

battling with monsters, much as that is thrilling and agreeable! Let's have a look, in Chapter 2, at a typical text adventure, and one way of playing it. A good adventure, of course, will reward several different ways of playing it — in future chapters, we'll have a look at graphic adventures, as well as arcventures. The first chapter is a brief glance at the origins of adventure.

*Atari Adventures* is, rather obviously, going to be a book about adventures for the Atari. But it will be about a lot more besides!

*Warning:* Please be careful when entering the program at the end of this book. The quality of copy from a computer printer means that some characters may look very similar — commas, in particular, may look like full stops.

# PART 1

# CHAPTER 1
# Origins

Switch the light on, quick! There, that's better, now we can see a little further ahead into the gloom. The strange noises went away when the light went on, and now all is quiet. Ahead, an opening in the cave well — let's go through and see what lies ahead for us.

Aha! What's this? A Black Rod lying on the damp floor of the cave. Go on, pick it up. Anything else around? No, that's it, so let's go further into the catacombs. In the next cave, an empty Wicker Cage lies discarded in one corner. Is it a trap? Try poking it with the Rod — wave the Rod at it — nothing happens, so we might as well pick it up.

Now the faint sound of a bird singing comes from the next cave. Quietly moving into the cave, we can see a Bird happily singing on a rock not far away. That must be what the Wicker Cage is for! Right, let's catch the little blighter. But it flies away — something is frightening it. Well, the Black Rod looks a bit menacing, so let's drop it. That's better, the Bird has settled again, and is merrily singing, oblivious to us creeping toward it with our Cage. The bird is caught!

On again, pausing to pick up the Rod (no doubt it will come in handy later), to the next cave, with Lamp held high before us. Suddenly a vast shadow rears up before us from the gloom! A huge Green Snake fixes us with its eye and sways before us. There is no other exit apparent, so we must get past this beast in order to progress on our quest. Let's stop and think for a moment.

Having the Rod might work this time — but no, the Snake isn't worried by that! How about using the Rod as a polevault? The Snake is too big to vault over. The little Bird is still singing away merrily — now, wait a moment here, maybe we can feed the Bird to the Snake. After all, the Snake is probably hungry, and this might keep it busy long enough for us to rush past and away. So let's free the Bird and see what happens. Now would you believe it? The Snake, frightened by the fluttering of wings, hisses violently and slithers away into the blackness, leaving us free to continue our quest.

This is a typical scenario from a computer-assisted adventure, in which overt violence does not play a large part, but there are variations

in which your sword-arm and fighting abilities take precedence over your reasoning powers. We'll look at some of these variations later, but for now, let's examine what we mean by adventure.

To find the origins of computer-assisted adventure, we have to look first at another pastime from the pre-home computer age. (Remember those days, back before you spent your evenings glued to the latest version of Zaxxon or Qix?) A pastime as old as chess itself, and nearly as old as another pastime.

Wargames have been fought since chieftains had more than half-a-dozen men to their armies — after all, even the most basic of fighting manoeuvres need practising, and what better way of doing that than making a game of it, thus giving some incentive to the proceedings?

Some very complex versions of wargames have been mounted, but the one that most people will recognise as such is the tabletop war. This has been a popular form of recreation and a military training technique since the 17th century.

Wargamers love complex rules, and H. G. Wells, an avid wargamer himself, published many years ago a slim book called *Little Wars*, covering certain aspects of tabletop battles. It soon became the bible for any serious devotee of the hobby.

During the 1960s, however, far more intricate books of rules to use with wargames made a commercial appearance, and became very popular. These rulebooks covered the smallest details of the period concerned, such as uniforms, weapons and logistics — the main periods were ancient, medieval, Napoleonic and modern.

There were, within these categories, many sub-categories, and one of these was medieval fantasy. Dave Arneson, of the Castle and Crusade Society in America, began a vast campaign and expanded the original rules to offer a complete environment for the players. These rules evolved. with the help of Gary Gygax, into one of the most successful games of the century — *Dungeons & Dragons,* published by Tactical Studies Rules. *D&D*, from TSR, presents the player with a highly stylised system of play, where nothing is left to chance, but everything played according to tables.

After the success of *Dungeons & Dragons*, it was only natural that many imitators sprang up, some making more impact on the scene than others.

Probably the most enduring of these has been Ken Andre's game *Tunnels & Trolls.* It simplified the rules of *D&D* to a great extent, but featured much of what had become so popular with the earlier game. Again, a Dungeon Master sets up a complex of caves (or in fact whatever sort of scenario he desires), and then the party of fellow-players is let loose to fare as they will. Combat is, similarly, moderated by the throw of dice, but *T&T* has no percentile dice. Spells are also

inherent in *T&T*'s system, progressing in hard-earned levels from the lowly 'knock-knock' (which opens locked doors) and 'take that you fiend' (which use IQ as a weapon), through such quaintly-named spells as 'Zombie Zonk' and 'Mutatum Mutandorum' to the ultimate, 'Born Again', which pretty well speaks for itself.

However, the main development of *T&T*, which has endeared it to many thousands of fantasy game-players throughout the world, and which makes it particularly interesting to us computer users, is the unique system of solo dungeons which has taken *T&T* as its game system.

These dungeons take the form of illustrated books, each containing a ready-made adventure which can by played, according to the *T&T* rules, by one person. Really a series of multiple-choice actions, in which the text acts as Dungeon Master, the books are a boon to the player who is unable to get together with other adventurers.

The great success enjoyed by these slim volumes is indicative of the great number of people who are unable to play full-blown RPGs (role-playing games) and who now look toward the computer as mediator, referee and Dungeon Master.

In the mid 1970s two enterprising chaps called Willie Crowther and Don Woods, whilst hunched over their huge mainframe computer, devised a game called *Adventures*. They were very likely *D&D* fans, as the scenario for their game included a complex of caves, peopled with assorted strange beings, and liberally scattered with treasure of all kinds.

Like *D&D*, the player makes his way slowly through unknown territory, receiving information about his surroundings — but this time from the computer. A tireless referee and Dungeon Master, the computer is the ideal medium for the fantasy game.

Other games came along in the wake of *Adventures*, probably the most successful being *Zork* which is the forerunner of many of the adventure games implemented for the home micros of the late 1970s. The PET, Apple and Tandy machines were well-served with these for several years, while the original was passed, in disk form, around the circle of computer professionals. This free market was, however, rather black—the companies who owned the big computers that were used by their employees for playing these illicit games, were, naturally, rather upset at this use of expensive computer time!

The obvious attraction of these computer-moderated adventures to the home enthusiast is that they can be played at any time, and alone if need be (it's quite often illuminating to play these games with companions, each putting in their own contribution). Whilst the game can often take weeks or months to complete, the state of play can be saved to disk or tape and resumed at a moment.

The obvious drawback for the home enthusiast, is that in its original form, *Adventures* can only be run on a mainframe computer, costing a couple of hundred thousand pounds — not the usual living-room furniture!

With the advent of the cheap microcomputer, programs of the adventure genre came within reach of the home user. Now solo play became possible to the enthusiast not lucky enough to possess an IBM mainframe.

# CHAPTER 2
# Text Adventures

At the time of writing, the end of 1983, a new microcomputer is announced every other week. Some of these have turned out to be what has been called 'vapourware', that is, they never materialise. While this is a distressingly common phenomenen in the computer business, there are still machines about, and very good machines. The initial software package that eventually arrives from the manufacturer of the computer invariably includes an adventure program.

Back in the pre-historic era of microcomputing, about five or six years ago, three machines dominated the scene. All were American, and all were very expensive — and, incidentally, still are! And all had implementations of *Adventures* and *Zork* written for them. The Apple I, the Commodore Pet and the TRS–80 are still with us, in one form or another, but the cost of these machines in those far-off days served to keep the adventure club rather exclusive.

We've seen the beginnings of computer adventuring in the Crowther/ Woods original and *Zork*, but several other popular programs were written in the States, in the mid 1970s. As computer prices dropped, and programmers became more efficient on their new toys, so these older programs were adapted.

Many of the programs taken for conversion were originally published in David Ahl's *Creative Computing*, an American magazine. The most popular games included *Hammurabi*, often mis-spelt nowadays as Hamurabi. In the original, you, the player, have to direct the eponymous administrator of Sumeria in managing the city. Given so many acres of fertile land, so many bushels of grain in storage, and so many people in the city, you have to balance all the ponderables in order to last a certain number of years. This type of game has been much adapted since its original appearance in *Creative Computing*, probably the most famous version being *Kingdoms*. A version is available for the Atari machines, from Atari itself — who also produce a program called *Energy Czar*, which puts you in charge of America's energy resources. The general term for the genre is Management Games.

Many of the adventure programs around today are actually descended

from these management games, in their careful balancing of several ponderables.

*Star Trek*, first written in the late 1960s in the flush of enthusiasm for the TV series, is a kind of Hammurabi, involving, as it does, a delicate juggling between weapons control and ammunition levels, with damage controls and repairs all taking their toll. This sort of game is ideally suited to playing on a computer, leaving the machine to take care of all the details of galaxy scanning and status reporting, leaving you to get on with zapping Klingons! In keeping with an aspect of the original TV series that I could never quite reconcile in my own mind with the programme's avowed humanity, computer *Star Trek* is extremely xenophobic, and the sole aim of the game is to rid the galaxy of Klingons, as rapidly as possible. The day may come, thought, when someone writes a Star Trek program in which the aim is to make friends with the aliens . . . The Atari owner has a particularly good version of Star Trek, the best-selling *Star Raiders*. It is so well-known that it is almost redundant to mention it here — almost, but not quite! Although this is very far removed from the original, the basic premise is the same. A large galaxy is divided into several sectors, many of which contain squadrons of Klingon battleships. Others contain starbases, at which the player's craft (the *Enterprise*?) may refuel, and re-arm. As in the original, the object of the game is to kill off as many of the Klingons as possible, but unlike the original, it is all achieved in arcade fashion. With the marvellous effects and ease of play, it is no wonder that this program has been a firm favourite now for a couple of years.

*Wumpus* was also written many years ago, and has survived, in fact flourished, to this day. The original game involved a search for the mythical Wumpus through a complex of squares. Using clues given to you by the computer, you eventually narrow down the choices, triangulating onto the final location of the beast.

We'll meet descendants of all these programs, *Adventures, Zork, Hammurabi* and *Wumpus*, in future pages. For now, let's start by explaining what a text adventure is.

Originally, adventure was played on the big mainframes, as we have seen — these machines, in those days, did not have the luxury of a monitor screen, instead printing their output on printers (the reason for several terms in home computing today, such as PRINT and TAB). Thus graphics, of course, were out. Instead, the whole game was played out using reams and reams of printout paper.

There exist nowadays, bands of reactionary adventurers, each defending their own favoured type of adventure. The devotees of the text variety believe that, without pictures making a preconceived notion of the scene in their mind, the better they can enjoy the adventure, with their imagination the only limit to that enjoyment. Adherents to the

graphic adventure believe that the artist's representation of the scene enhances the game. I love adventures of any kind — while enjoying watching a good artist at work, I find that each text adventure takes on a personality of its own as the imagination works to flesh out the bare bones of the descriptions (you'll find, however, that Infocom and one or two others are so verbose in their descriptions as to leave very little to the imagination!)

The typical text adventure begins with the description of a location. In the original, this was outside a small wooden hut, in the middle of a forest. This description may be rather terse — 'You are standing outside a small wooden hut.' If memory is not constricted, the locations may have correspondingly more description, although most authors would then decide to include more locations. If the adventure is supplied on a disc, then each location may be stored and read by the program as required, and the description may be much more detailed.

After the description, be it long or short (and see the later section dealing with Infocom adventures), the computer asks the player what he wishes to do. There are two ways in which adventures may be conducted. After describing the location, the computer may then say: 'I see nothing of interest. What do I do now?'. The adventure will then be played out from the point of view of the computer, which you, the player, will direct. The computer, in this case, becomes the adventurer's 'puppet'. Again, see the Infocom section — as befits the most original adventure software authors around at the moment, they approach this kind of third person adventure in a completely fresh way, in their program, *Suspended*.

First person adventures are subtly different, however. In this variety, you will be the protagonist, and the computer will say: 'You see nothing. What will you do now?'. For me, this makes the whole thing much more personal and immediate, and I would normally prefer to play this kind of adventure.

The difficulty in describing the text adventure is in attempting to convey the essential flavour of the game, without giving away the answer to a problem that may have been intriguing a reader for many weeks — and much of the fun in solving text adventures is in shaking a tough problem by the neck until it succumbs.

We'll have a look now at *Dungeon Adventure* by Level 9. Any reader who finds himself currently in this particular adventure should skip the rest of this chapter! Those of you who may be thinking of buying this program should not despair — nothing crucial will be given away.

*Dungeon Adventure* is part of an excellent series written by Level 9, (High Wycombe, UK) which starts with *Colossal Adventure*, based closely on Crowther/Woods original *Colossal Cave* program, but adding some 70 locations to the end game. The first program in this series,

called *The Middle Earth Adventures* by Level 9, is *Adventure Quest*. All the Middle Earth adventures inter-relate with each other and form a large fantasy would which can be explored and enjoyed.

A new title, just recently released, is *Snowball*, set on a huge, five-mile long starship. With over 7,000 locations (so Level 9 claim!), and rampant, man-eating nightingales patrolling the corridors of the spaceship, the title probably refers to your chances of surviving! *Snowball* is the first in a new series of Level 9 adventures, which promises to be the equal of the Middle Earth series.

The opening scene of *Dungeon Adventure*, and the puzzles associated with it, is a very pure example of the classic adventure program, and will serve to lay some ground rules for adventuring in general.

The booklet which accompanies the cassette of *Dungeon Adventure*, is typical of the quality of Level 9's output. A page of scene-setting, with a brief snatch of story concerning the seige of Minas Tirith, the death of the Demon Lord, and the protagonist's decision to rush off and search for the Lord's treasure in the Black Tower, prepares you for the start of the game. The rest of the booklet outlines the various commands you may give to the computer, along with several hints for the adventurer. Finally, a unique touch in an envelope in which you may, if you get dreadfully stuck, beg Level 9 for a clue.

At the start of the adventure, you, the hero, awake cold and weaponless on a mudbank beside a large packing case, open at one end. There is also a piece of driftwood lying nearby. You see a stone bridge across a river, reaching from the granite cliffs above to the flat lands of the far bank. Now, you may, of course, go charging off up on to the bridge, eager to get on with exploring the caves which you secretly know are up there. And here we get to our First Golden Rule of Adventuring:

*Look around every location in as much detail as possible.*

Adventures of the traditional kind, which we are presently engaged in, almost invariably start off with the player alone and defenceless outside some sort of building.

In this present instance, there are some 170 locations to be explored within the cavern complex that you will find over that bridge. Your chances of surviving to see more than about half a dozen of them are, however, depressingly remote! Very soon after starting off, you will find certain objects become necessary — objects which can only be obtained from the outside locations. That is, the locations which you will find on this side of the bridge.

You will find out all this after a couple of tries at the cavern complex — and now you can start a search for these vital objects.

We know that there is nothing useful — yet — to be found by going north across the bridge, so let's try going south.

But first of all, we much not forget our First Rule of Adventuring, and look around as much as possible! Now, if you remember, we are on a mudbank, and we can see some driftwood and a large packing case. Let's pick up the wood. Different programs recognise different commands for this action, and we can try PICK, TAKE or GET, the usual ones. In fact, *Dungeon Adventure* will accept TAKE, so now we are the proud possessors of a lump of wood.

There doesn't seem to be much to do with it, so let's concentrate on something else.

The packing case, if you remember, was lying with one end open, so it would be a good idea to go inside, wouldn't it? No, it wouldn't! Not yet, anyway. Writers of adventure programs are a sadistic lot, and would like nothing better than for you to walk wide-eyed and innocent into a trap — maybe just like this one. So let's try another key word which you will often come across in traditional adventures such as this one: and this is EXAMINE N (where N is the object to be EXAMINEd).

So let's type in EXAMINE CASE and see what the computer has to tell us. The case is enchanted. Not only that but there is room for us to crawl in. So we know now that we will not be eaten or stepped on by some passing monster, and we can proceed unscathed, for the moment.

Let's leave the case here, though. There are secrets to be learnt inside, of course, not least being the fact that the case is where you must deposit your treasure in order to gain points and win the game. The riddles to be solved in order to gain this information are fun to work out, and I don't want to spoil your fun at such an early stage.

Let's assume, then, that we've thoroughly explored the case, and are back outside on the mudbank, with our lump of wood. We have decided, maybe after several painful attempts, that we are on a sticky wicket in trying to negotiate the dark caverns to the north of the bridge. So let's try going to the south from the bridge. I am playing the adventure as I am writing, so this is a voyage of discovery for all of us.

We are now on an east-west road south of the river across which runs the bridge. A gigantic orc's head is carved into the cliff north of the river, its tongue forming the bridge. A ruined tower stands on top of the cliff. Exits are EAST, WEST and SOUTH. This is the standard formula for text adventures — enter a location and a brief description is given, with a list of possible exits. Actually, the descriptions of the various locations are rather detailed in Level 9's adventures, which is thanks to the larger memory available.

Usually, another list is also given — that of the objects and/or monsters or other entities to be seen, and the things that you may be carrying.

There is nothing to be seen lying around at this location, however, so we have to make up our minds which direction to take now. We have no clues here, so let's toss a coin and take the east road. The next location is, again, the east-west road, but now we are further along and we can see to the south a flat grassy plain, stretching as far as the eye can see.

This is one of those phrases that you will come to treat with respect. It usually means that if you once set foot in the location to which it refers, you will be lost foreyermore — or at least until you give up in sheer despair.

So we will give that grassy plain a miss for the moment. However, we can also see, so the computer informs us, a line of stepping-stones, which leads to a small island in the water. A young girl with flowing locks sits on the island — how charming! So let's type GO NORTH. The computer informs us that we are now on the stepping-stones, and the girl is still sitting there on the island. We type GO NORTH again. Oh dear. The girl was a siren, and sings her siren song — we flounder into the treacherous waters of the river. The computer informs us that we have managed to get ourselves killed.

The program now dumps us back on the mudbank, and we have to retrace our steps back to the east–west road, just south of the bridge. This time, let's explore to the east. We type GO EAST, and the computer duly tells us that we are now further east along the road, north of a steep, treeless hill.

This treeless hill seems to be worth investigating — going south, we come to the side of the steep hill, which rises into the clouds. Rumblings emanate from above! We can go UP here, so let's do so. We are now in a circle of distorted monoliths, etched into grotesque figures by the acid rain. Lightning arcs overhead, revealing the stark horror of this haunted place, while thunder echoes from every crag. Sounds very cosy, doesn't it? If we instruct the computer to LOOK around, a host of fierce flames will spring forth: they are Rakshasa! Their leader floats forward and challenges us to a game — if you win the throw of the dice he'll make our spiritual strength greater in some way. Of course, we may lose . . .

We have to play, and after all, it sounds quite a bargain, doesn't it? Unfortunately, we need a way of loading the dice (these Rakshasa are pretty cunning), and we haven't found it yet, although I might say that it is to be found somewhere in this opening sequence.

As you may expect, we lose this particular game, and once again find ourselves, after being asked if we want to play again, back on the mudbank — we're learning each time, though, aren't we?

Let's get back to that stretch of road just north of the steep hill where we recently met the Rakshasa. We won't go back there again until we find the method of winning the dice game. So we continue west to see what awaits us there.

And now we see a vast field of poppies which stretches west as far as the eye can see — and where have we seen that phrase before? We must tread very carefully here! A somnolent perfume hangs heavily on the breeze. Nearby we can see a dry poppy seed lying on the ground. Let's pick it up, by typing TAKE SEED. And here we come to our Second Golden Rule of Adventuring:

*Everything has a purpose*

The typical adventure program is so packed full that the programmer cannot afford to leave too many useless items lying around. Like all rules, however, this one can be broken, and you will come across the occasional red herring, which will have you rushing about trying to find some way to use it. And of course, there will sometimes be a really horrid author who will delight in handing the unwary adventurer a bomb in disguise!

So, although we can act on this Golden Rule, we must add a rider:

*Exercise extreme caution at all times*

Now, let's retrace our steps — I have a strange feeling about that field of poppies!

We can work our way back along the road, past the steep hill, and the bridge, back to those stepping-stones. As we have a poppy seed with us now, we might be able to use it to fight that siren (we don't know how yet, but it'll be worth a try!). At the stepping-stones, we type GO NORTH, which gets us on to the stones. Last time we were here, we got unceremoniously dumped in the water when we tried to proceed further north past the siren, so let's be a little more cautious this time.

We might as well try the poppy seeds now — we'll type THROW SEEDS and see what that does (I'm condensing a lot of heartache here, and a lot of effort in finding the correct solution!). The seeds, in falling, burst with loud explosions. Did they frighten the siren off?

To find this out, we have to give a command to the computer. Type LOOK, and the computer will obligingly inform you of what is at the location. Quite often, of course, the answer you get is 'I see nothing special', but the question is, like EXAMINE, well worth asking. LOOK should not be confused with REDESCRIBE (REDES or R for short), which will instruct the computer to do just that with the present location.

Back to our present problems. Having tried LOOK, we find that the siren is still there, sitting by the river. Now we'll introduce yet another useful word—LISTEN. Unpleasant things have a nasty habit of hiding behind trees or rocks, out of sight of the computer, but can't help

making slimy, squidgy noises, which will give them away!

Ah! Now we're getting somewhere! The computer tells us that we have been temporarily deafened, and can't hear anything. Although she must still be singing away as a good siren should, we won't hear her, so it's probably safe now to creep past her . . . but she's got us again! The temporary effect of deafness must be very short-lived.

Here we are back again at the mudbank. Now we can waste no time in getting back to that pesky siren, not forgetting to collect the poppy seed!

Another word worth remembering, and which would help us in the future is SAVE. Most good adventure programs will allow this word which means that you can save your present stage of play to tape or disc. You should, if possible, do this before taking any risky initiative. We should have been more prudent and done this ourselves before attempting to get past the siren — we would now simply be able to type RESTORE and, once our old position was re-established, make a fresh attempt to cross the stepping-stones. As it is, we have to do a lot of typing to return, poppy seed at the ready, to that siren.

So, let's make this our Third Golden Rule of Adventuring:

*Always save your position if in doubt*

But here we are again, and we've dropped the seed, which duly explodes. We'll SAVE our position in case we are discombobulated yet again, and this time let's get on with it and rush past her. After all, we know now what effect the explosions will have. And — it works! The siren flees in panic, seeing that her song has no effect on us. In fact, the program will at this point, allow you up to four commands before the deafness wears off.

And this is another ploy that the better adventure program will use. Approach a problem in one way, and the result may be completely different from the outcome of another attempt.

We're now at the southern end of a small island — the far end is occupied by a vicious-looking willow tree with six long rubbery branches. A silver mirror lies nearby. We can TAKE the mirror, but if we try and get past the tree, we will be killed. Why *six* branches, there must be a clue there!

Let's leave *Dungeon Adventure* now. This particular game is extremely rich — we've only covered a few locations and I haven't mentioned the resurrection procedure which you will need before playing the game for real, and several other important details which can be found in the opening scenario.

Actually, this whole sequence, which consists of some thirty rooms is only an introduction to the main adventure, which takes place in the cave complex that we saw at the start. The original *Adventures* program

started with the player standing outside a small building in a forest. This building contained several articles required for the adventure, such as keys, food and water — any treasure had to be brought back to the building. The very large network of locations at the start of *Dungeon Adventure* serves the same purpose as that small building in the original — but now in a far more complex way, which in fact is an intriguing mini-adventure in its own right.

After our first, fumbling attempts at *Dungeon Adventure*, you should have gained an idea of what it feels like to play a text adventure (and feel motivated to play Level 9's adventure yourself).

Let's summarise what we've learnt in this chapter so far:

*The Golden Rules:*
I've mentioned four, but of course, you will be able to think of others!

*Words*
As the argument over text or graphic adventures rages, so there is a similar argument over vocabulary. To reveal or not to reveal, this is the question. Some adventures disclose all in the documentation, in which case all the words recognised by the computer will be set down in black and white — there will be no confusion when it comes to playing. Most adventures, however, deem it necessary to add the Great Word Hunt to the puzzles inherent in the game, in which case only a smattering of the recognised words will be divulged in the documentation, or possibly none at all. In the former case, the player is given just enough to start him off, while the latter case assumes that the player is a hardened adventurer and will know the conventions.

I'm personally in two minds about all this — if the game were a real-life situation, the player would of course be on his or her own, and would have to react in whichever way he saw fit, in which case his lack of vocabulary may very well influence the outcome of the game. But then many games are so tough that the player will certainly not need to be confounded from the very start, and have to work out the vocabulary before even getting to the nitty-gritty of problem-solving.

So, if the authors are kind enough to give me guidance with vocabulary, I look upon it as a bonus and accept it gratefully — if nothing is given away, then I shrug and get on with it. It can be particularly galling, however, when this little bit of bad programming is encountered (and it is encountered in almost every program, unfortunately, with a few notable exceptions). You come across a location which is described thus: 'YOU ARE IN A CLEARING. BEFORE YOU IS A HUT' — and then the program refuses to recognize the word HUT when you answer: ENTER HUT, saying instead: I SEE NO HUT HERE. This happens quite often,

unfortunately, and all that is happening is that the object is included, rather unfairly, in the location description as mere colour.

## A potted vocabulary

This is a short list of those words that you will most often meet in adventures. The list isn't exhaustive, so if these don't do the trick for you, try a Thesaurus!

**GO:** This may be used when in a location with only one exit. However, most adventures are waiting for at least a two-word input, and you will then be asked: GO where?, or: Try a direction!. If an open window is included in the initial description of the room, you can usually obtain a view from the window by typing: GO window.

**DIRECTION:** You can, of course, type GO NORTH, GO WEST, GO UP and so on, but most programs will allow just SOUTH, EAST — or even U, N, W, which saves a lot of typing and makes for a much faster game.

Lateral thinking will often work wonders when you are deciding which way to go. Many adventures describe the location with 'SOME of the exits are . . .', and you may then try directions that are not mentioned. And, of course, the cardinal compass points are not the only ones you can use — try the odd SW or NE! If a ledge is at the location, you may often jump up, or climb up, and you'll usually be well rewarded for doing so (although you could easily find yourself in a man-eating tiger's den).

Whichever way you find yourself going, make a map! If you get lost, you will be to retrace your steps, or start the whole adventure again, and quickly get back to where you were (see SAVE, later).

**QUIT:** This is the word to use if you're ready to give up. Very occasionally, an author will regard this as the worst Californian-ese, and require you to type STOP, but such pedantry is mercifully rare. Most adventures will then go on to ask if you really want to QUIT, so giving you a chance to change your mind.

Then you are given your score, if a score feature is included. Scott Adams' *Adventures* will inform you of the number of treasures you have so far collected, and give you a rating based on this number, compared with the total treasures available.

Channel 8 Software, in their *Mysterious Adventures*, seem content to stop the game as soon as you type QUIT, and you'll get no second chance! You'll not find a SCORE routine, either, in *Mysterious Adventures* — if you ask for your score, you'll be told: THIS IS NOT A FOOTBALL MATCH!

**HELP:** You'll get a lot of funny answers to this question — be prepared to be well and truly insulted for admitting to the computer that you haven't the faintest idea of how to progress. Many programs will merely say: LOOK AROUND AND TRY REPHRASING YOUR COMMAND, which means that you will get absolutely no help whatsoever throughout the course of the game. Others will sometimes give a cryptic clue, based on what is currently at the location: but no adventure can give you a specific clue.

**INVENTORY:** Typing this, at any time, will give you a list of items that you are currently carrying. The word can be abbreviated to IN (don't be surprised if the computer misunderstands you, and takes you into a room full of bloodthirsty Orcs!), or I, which is probably safer.

**LIST:** Often accomplishes the same as INVENTORY.

**LOOK:** Will give you another chance to see the room or location that you are in.

**REDESCRIBE:** Has the same effect as LOOK.

**EXAMINE:** When you come across the body of a dead tramp, or a pile of leaves — in other words, some fairly inanimate object — then EXAMINE it! Don't forget our Golden Rule: *Everything has a purpose.* Adventure programs haven't the memory space to include non-essential items, so assume that anything you see will have some purpose in the overall scheme. Thus, a pile of leaves will often conceal a gleaming sword, or a dead body yield up a diamond bracelet.

**SAVE/RESTORE:** Judicious use of this command will save you a lot of time. Before embarking on a hazardous mission, like going into a dark cave, or opening a large wooden door, first of all SAVE your current position. Then, if you are killed off, as you usually will be, you won't have to retrace your steps — steps which have no doubt been very hard-won! RESTORE is the command usually required in order to reload your position from disc or tape.

**PICK/TAKE/GET:** One of these will be your program's required command to pick up an object. To find if you have indeed acquired the object, type INVENTORY, when you should see the object added to the list of things carried by you.

**DROP:** Conversely, you may need to leave an object — most programs have a weight limit, which, realistically, you cannot exceed. So a careful

compromise is required, between an object's usefulness, and the player's acquisitiveness.

**OPEN:** This command is most often used when the player is confronted by a door, or receptacle. Most programs are very pedantic in this regard — you will be required, usually, to unlock the door or chest before being able to open it. Sometimes, the player will have to PICK or TAKE the chest before being able to open it.

**KILL:** This is one for the old *D&D*ers! Most adventures will recognise the word, or indeed any synonym, like STAB, KICK, THROTTLE, BREAK (that can also be used against the stubbornly-closed door) and so on. Finding the one that will elicit the response you are hoping for is really left to your imagination — don't forget to SAVE your position first.

**D———:** Along with all the other words that will be thought of at 3 am, when you are about to throw the d———ed computer out of the window, some form of swear word is almost always understood by the program. If they are not, it does not say very much for the programmer's sense of humour — on the other hand, you can't really expect the program to recognise every variation that you might dredge up! And don't expect to see a long list in our program *Eye of the Star Warrior*, this is a family book!

This is necessarily a very short list of the words you will come across — you may be expected to DIG, SWIM, FLY, SHOUT, WHISPER, CRAWL, MOVE, SPIT, DRAW and COMMIT SUICIDE among many other possibilities. The only way to find out which words a particular program accepts, is to try every thing that comes to mind.

There is no British-written software for the Atari to match Level 9's special blend of humour, complexity, literacy, puzzle-setting and sheer bravado. Their programs will have you alternately shaking with frustration and curled up with uncontrollable laughter — sample, for instance, the moment in one of their adventures when the player is confronted with a tiny plant which whispers in a meek, pitiful voice: 'Water . . . Water'. Taking pity on the poor thing, the player waters it, and then can but watch as the plant rears up to eight feet, and menaces the player, demanding: 'WATER, WATER !!'.

One British software firm which challenges Level 9 is Channel 8 Software, of Preston. Channel 8 has some ten Adventures for the Atari, as well as the Spectrum, BBC, Commodore 64, Dragon 32 and other machines. The catalogue has a long and honourable pedigree, originally being written, by Brian Howarth, for the TRS-80 and Nascom machines way back in 1980, and marketed through Molimerx. A list of the intriguingly-named adventures follows:

*1 The Golden Baton:* Ever since the Golden Baton was stolen from the kingdom of King Ferrenuil, brave Warriors and hard Knights were sent far and wide through the world in search of this artefact . . . none ever returned . . . Now it is up to you!

*2 The Time Machine:* Alone in the fog on the moors, you, the news reporter for the Tulkingham & Dunsby Gazette, have set out to investigate the strange goings-on around the old house . . .

*3 Arrow of Death:* You have at last regained The Golden Baton, and returned it to King Ferrenuil. Recently, however, the Golden Baton has become a symbol of hatred and evil. With a sense of foreboding, you are now travelling to the palace, full of dread. Surely a mere mortal, such as yourself, cannot stand up against the evil power that threatens the future of your land . . .

*4 Arrow of Death Pt.2:* Having successfully completed Part One, you will now need the Arrow in order to defeat Xerdon the Evil!

*5 Escape from Pulsar 7:* Onboard the PULSAR 7, you are alone and being hunted by a savage creature, which has killed and eaten the rest of the crew . . .

*6 Circus:* Out of petrol on a dark, deserted country lane, you come upon a circus in the middle of a field. The raucous sounds of laughter and merriment stop as soon as you approach . . . darkness is falling, and it's going to be a long, long night!

*7 Feasibility Experiment:* You awake from a troubled sleep to find yourself . . . the subject of an experiment by strange, disembodied aliens from an artificial world far beyond the rim of the Galaxy (where else?). They wish to test the human race, to find a heroic strain for their gene pool — they will test your heroism in an arena . . .

*8 The Wizard Akryz:* The evil Sorcerer has had enough of the pesky mortal who has foiled him at every move in his attempt to snaffle the Golden Baton. He has now dreamed up the ultimate plan — you are that mortal, and this looks like Golden Baton Pt 3!

*9 Perseus and Andromeda:* You are Perseus, and your task is to bring back the head of Medusa the Gorgon, whose very glance will turn grown men to stone!

*10 Ten Little Indians:* The treasure of Major Johnston-Smythe has been much-publicised in the press, and it is hidden somewhere in the creepy Manor of Major J-S — you cannot resist the challenge of finding it, and thus find yourself on a train, rattling through the countryside towards your destiny . . .

By the time you read this, one or two more will have been released, although too late for me to have seen copies. Again, they have intriguing titles like *Midwinter* and *After the Fire* (a post-nuclear adventure).

The versions for the Commodore and Spectrum contain high-resolution graphics, to support the location descriptions, but as yet the Atari owner will have to make do with text-only displays.

In the main, the programs follow the usual pattern of the two-word input, and thus hold no surprises for the adventurer weaned on Scott Adams. Howarth even follows Adams in dedicating each adventure (sometimes obscurely, true, but that is what dedications are for, surely?).

The plots and scenarios, as we've seen, are unusual and make a welcome change from the troll-bashing that is present in many other adventures. Many of the puzzles are also extremely hard to crack (of course, once unravelled, that's it — there is no other way to solve these adventures but one), and will keep you guessing for a long while. The adventures themselves, and the attractive packaging, which would look good on any shelf, make for an interesting series that will reward a closer look.

And that is about it, as far as home-grown, nationally-available adventures for the Atari are concerned. Both Level 9 and Channel 8 programs sell for about £9–10 ($14), which makes them great value for money. It is worth looking in a British-produced magazine called *Page 6* (an Atari programmer's in-joke!), for news of home-grown software. Not as expensive as American products, there are the occasional programs that will interest the adventurer. American software is more expensive — from the £16 ($23) for the Scott Adams adventures, which is also pretty good value — on up to £30 ($44) and beyond.

# CHAPTER 3
# Scott Adams and Infocom

Two of the American software houses stand far taller than the others producing text adventures — Scott Adams's Adventure International, and Infocom.

The first name most people hear of when they start computer adventuring in earnest is Scott Adams. This is especially true if they own an American machine, such as one of the Commodores, the Apple, or the Atari. Scott Adams, and his company, Adventure International, have had almost a monopoly in American adventures, where the British machines, like the Spectrum, the Dragon and the BBC are very much in the minority. At the time of writing, late 1983, versions for all these British machines are appearing written, incidentally, by Brian Howarth, of Channel 8 fame (and one suspects that a great deal of prodding went on to persuade Adventure International that there was indeed a market for games on these machines!).

It is fitting that Howarth should have a hand in implementing these adventures for home computers, for Channel 8 programs have a great deal in common with Adams's adventures, even down to the dedications at the start of each game.

Scott Adams started his career as a computer programmer in Florida, and, like many before and since, found Crowther and Woods's original *Colossal Cave* on the company's mainframe. Entranced at the fantasy world conjured up, he wrote a version on his TRS-80, before releasing a commercial program, *Adventureland* in 1978. The project very nearly died in the making, when Mrs Adams put the disk in the oven — luckily for Scott, and a grateful world, the oven was not switched on!

Adams went on to found Adventure International, which nowadays has several arcade success stories, including the great *Preppie*, as well as many business programs and utilities. The company is also promoting another adventure-writer, Jyym Pearson, who has written four excellent adventures.

The programs written by Scott (and Mrs) Adams are recognised as classic text adventures.

There are twelve:

*Adventureland*: this is the one that started it all. There are thirteen

treasures in all to find, and some 29 locations to be mapped. All the other adventures follow the same pattern as this first one, and indeed, Adams wrote an adventure compiler, which acts as a framework on which any storyline can be hung.

*Pirate's Cove*: and this shows the technique in action — Mrs Alexis Adams actually wrote this one, using the compiler. It is probably the easiest, but do not believe for one moment that any Scott Adams adventure is not frustrating and devilishly difficult! A lot of humour is present in *Pirate's Cove*, and it has 25 locations, though only two treasures.

*Mission Impossible*: is a race against time, with the player attempting to stop a saboteur from blowing up the nuclear reactor.

*Voodoo Castle*: set in a dark castle with 24 rooms, this one leads on naturally to

*The Count*: Dracula, in this case — although the adventure only has 19 locations, 'things change' throughout the game, as night succeeds day.

*Strange Odyssey*: set in space, on a damaged spaceship, crash-landed on an alien planet, this adventure has 22 locations, and only five treasures — but the last one will prove extremely difficult to find!

*Mystery Fun House, Pyramid of Doom,* and *Ghost Town*: a trio of adventures, the scenarios of which should be fairly obvious from the titles. *Fun House* has some 37 locations, and is many people's favourite of the twelve. The second can be rather tricky, while the third is extremely difficult. It has nearly 40 locations, and can be solved in more than one way. It also has a couple of objects carried over from *Pyramid*.

*Savage Island Parts I & II*: these two are parts of a whole — the first has to be solved before the second can be tackled. A password to the second is the goal of the first. They are the toughest of the Scott Adams adventures, and not for the faint-hearted.

*Golden Voyage*: less difficult than the *Islands*, this concerns your race to bring back the elixir of youth for the king.

Like an old music hall joke, these adventures may appear, at first glance, to be just like all the other programs around, but they were the first, and in fact, most of the other text adventures merely imitate the

Scott Adams series. No news, yet, of more adventures in the series, but the existing programs will keep many players happy for hours and hours, in their attempts to solve the Adams conundrums.

A fairly recent development from Adventure International is the *Saga* series. This stands for the Scott Adams Graphic Adventures. AI has taken the first half-dozen adventures (although there may well be more by now) and, without changing the plot, have added graphic displays.

We'll quickly run through a couple of the early locations and see what difference the graphics make.

The adventure starts in a flat in London. And the program tells you:

> *"I'M IN A FLAT IN LONDON. VISIBLE ITEMS: FLIGHT OF STAIRS. SIGN SAYS: "BRING * TREASURES * HERE, SAY SCORE" BOTTLE OF RUM. RUG. SAFETY SNEAKERS. SACK OF CRACKERS."*

And sure enough, you can see all those items scattered around the room. If you now type:

TAKE SNEAKERS

The program will reply:

*"OK"*

and, lo and behold, the Sneakers will have disappeared from the room. You can carry on and do this with the Rum and the Crackers — try and TAKE the Rug, though, and you will be told that it is nailed to the floor. Serves you right, you vandal!

To see what you are now carrying, type INVENTORY. A picture will then be called up from the disc, showing you (rather ugly, too!) holding up a sack, out of which has tumbled all your possessions, which lie scattered around you. You are also told in text what you carry.

Now, the only way out of the flat appears to be via the stairs, so type:

GO STAIRS

and the program obliges. Unfortunately, the light has gone out, and two big eyes stare out of the blackness, with the legend:

*IT'S TOO DARK TO SEE!*

At the moment, you have nothing to light, so just grit your teeth, hope there is not a time limit set on movement here, and carry on upwards.

Eventually you will come across a boat, accompanied by a picture of the same. Also around here somewhere is a quaint picture of Never-Never Land, complete with mushrooms and a rainbow (or could they be Munchkins?). Most of this early part is conducted in pitch black, although you will eventually find a Musty Attic in which there is a source of light, as well as other goodies.

Once you have light, of course, you can wander around to your heart's content, and you will find that there is nothing outrageously difficult about *Pirate's Adventure*, although, like all Scott Adams adventures, they are not the easiest.

If Scott Adams is the American Channel 8, then Infocom is certainly the American Level 9. Infocom displays a lot of panache — in both their programs and the packaging of those programs. Just look at the paraphernalia that is supplied with *Deadline*. This adventure is a sort of sophisticated *Cluedo*, with you as the Inspector, attempting to solve the murder mystery. You are given 12 hours to accomplish the task, and a lot of paper. This paper includes lab reports and so on.

You'll also be given a page from a newspaper, a scrawled note, and a book of matches with a telephone number on the back. It all adds to the atmosphere surrounding the adventure, and it is typical of the verve with which Infocom endows its programs.

But to start at the beginning — Infocom is the first of a new breed of companies revolutionising the adventure game market. It is a young company, made up of young computer professionals who cut their teeth working within the hallowed walls of the MIT (Massachusetts Institute of Technology).

Since going it alone, Infocom has played a large part in the development of adventure games. *Zork*, their first program, has become a classic unequalled by any other adventure but the original *Adventure* and the Scott Adams series. If this had been their only program, they would have a secure place in the Adventure Hall of Fame, but they have since released *Zork II* and *III*, each of which is more than a match for the first.

*Zork I* was originally written, in a LISP-inspired language MDL on mainframe computers. In the early 1980s, Infocom used its expertise to implement the program on microcomputers. It borrows heavily from the Crowther and Woods original *Colossal Cave*.

The games are now available for a very wide range of computers, including Apple, Commodore, CP/M systems, Tandy, Osborne and DEC.

One of the reasons for Infocom's success is the languge used in the programs. The descriptions are no mere one-liners, as in other adventures — instead, they are extremely witty and rich in detail.

Whereas the Scott Adams adventures are rather spare in their scene-setting and recognise only short commands from the player, being content to give the player more and more difficult puzzles, Infocom has developed an extremely sophisticated 'parser', a routine that allows the player to communicate with the computer in a natural language.

Thus a Scott Adams adventure will say: *I'M IN A FLAT IN LONDON. VISIBLE ITEMS ARE . . .* , and the player must reply with one or two words, for example: TAKE SNEAKERS. An Infocom adventure, on the other hand, will say (and notice the shift in emphasis here to the first person): *YOU ARE STANDING BEFORE A CHASM . . .* (the description may well go on to fill up a whole screen!), to which the player can reply, for example: THROW THE NEWSPAPER, THE RED BOOK, AND THE MAGAZINE IN THE CHASM, or similarly: TAKE BOOK. GO NORTH. DROP BELL, BOOK AND CANDLE. Other characters may be asked to carry out various actions, or divulge certain information — for instance, the whereabouts of the Gold. The program will even try to figure out what you mean when you don't give enough information.

While LOOK is recognised by every adventure game, although the word can take on several different meanings, think about the possibilities in *Zork*, where LOOK INSIDE and LOOK UNDER are equally valid!

Several commands can be used when travelling about the world of *Zork*. VERBOSE, for example, will ensure that you are given a very detailed description of each room, every time you enter. BRIEF can be typed, in which case, only newly-encountered rooms will be described, while SUPERBRIEF will give only the name of each room.

Although each of the three Zorks may be played and enjoyed separately, there are glimpses of others in each one, and the storylines do in fact follow on.

In *Zork I*, the Great Underground Empire confronts the player with predicaments ranging from the mystical to the macabre, with 20 treasures to be found before escape has to be made. *Zork II* takes the player to new depths of this subterranean realm where the Wizard of Frobozz will be met. And in the final game of the trilogy, players encounter the Dungeon Master himself, who holds their destiny in the balance.

The *Zork* adventures are, though, only the tip of the Infocom iceberg — unrivalled though they are.

*Deadline* we have seen, and this is complemented by *The Witness*, which is a whodunnit rooted in the popular crime novels of the 1930s and 40s. It too boasts a dossier stuffed with clues in the form of matchbooks, newspapers and so on.

The science fiction adventure *Starcross* takes place in the year 2186.

The player has to rendezvous with a starship from the outer reaches of the galaxy and enter its mysterious interior. Once inside, other-worldly beings are met, some helpful, and some harmful. Another science fiction adventure is *Planetfall*, though this one is rather more light-hearted than the others. It takes place on a distant planet, where the player has been ship-wrecked, armed with only a patrol-issue, multi-purpose scrub-brush, and aided by an impish robot companion. The player is challenged with saving the doomed and plague-ridden world while trying to keep a straight face.

The most original of the Infocom Adventures is *Suspended*, subtitled *A Cryogenic Nightmare*. Written by staff writer Michael Burlyn, an established writer of SF novels, the adventure concerns cryogenic suspension — bodies in deep freeze! Having woken from a long, frozen sleep, the player finds the planet in crisis, and remembers that he has won, 500 years before, a lottery to serve as Contra's Central Mentality. Using a game board and game pieces, the player must manoeuvre six robots in an attempt to end the crisis.

Each of the robots has one sense, and thus the overall picture has to be built up from the input received piecemeal from each robot. Poet, as his name hints, is given to long-winded, obscure passages, while others are more straightforward in their information. The player also has control of Waldo, which is able to manipulate objects, and Whiz, which can interface with the Central Computer's memory banks. Thus, it is not enough to say GET OBJECT — in *Suspended* the player will have to first ensure that Waldo is already at the location — after making sure that Iris, the Eye robot, has already seen the object. And, of course, robots can go wrong!

There is no need to map the 58 locations, as in the majority of other Adventures — this is taken care of by the board and rubber pieces already mentioned. There are four levels of play, one of which will allow the player to configure the adventure to his own tastes — the robots may be placed in any starting position, and the various systems set.

The most original and challenging adventures that I know of!

# CHAPTER 4
# Arcventures

The American software houses have a love of arcade games in which each level (or sheet, as the arcadists have it), contains a puzzle — solve this, and the player can move on to the next, usually harder, level with its attendant problems. At least, they love these games this year, for that is where the demand lies! Next year, or next month, there will no doubt be a different kind of arcade game that will hold the public's attention.

Although the term 'Adventure' is often applied to arcade games in a rather cavalier fashion by the software houses themselves — thus aiming at two sectors of the market at once — there are several major arcade games that do indeed qualify for the name, as far as the complexity of the puzzles contained therein are concerned.

From JV Software comes *Journey to the Planets*. This program has been around now for some while, but still holds a certain charm.

The Journey starts on your home planet, where your little man stands in the middle of a bleak landscape. The odd tree is to be seen, and a few Lego-like buildings. Nothing moves. Shift the joystick, and your man can explore — take him to the edge of the screen, and the present display is replaced by the next section of the landscape. Unfortunately, this is where the program first shows its age — the Atari is very well able to support fine scrolling, and many, more recent, programs implement this in some way or other. It would have been a plus point if the author of *Journey to the Planets* had thought to include this in his program — but no matter, it is a small inconvenience.

On your home planet, there is a grand total of five such screens — in one of these, you will find a revolver. Remembering our First Golden Rule of Adventuring, which we encountered in the section on Text Adventures, we know that everything has a purpose. So we pick up the gun, and take it with us. Another part of the planet has a strange-looking building which will act as an energy-reviver, when you return to home base. The most interesting thing you'll find, though, is your spaceship.

This is a small H-shaped craft, and you take off by moving on to it and pressing the fire button on the joystick. Up it goes into the air, and when

it reaches top-screen, the display is replaced by the view of the top of buildings as your spacecraft heads for the stratosphere. The incessant music dims as you pull away from the surface, and the next screen is deep space.

Press the fire button now, and a Galaxy Map is shown. This is very similar to the one in the venerable *Star Raiders*, and shows your current position, and information about surrounding sectors of space, with planets marked. In order to move, just push the joystick in the required direction — little jets of flame appear, and the longer the joystick is pushed, the faster the spacecraft moves. It obeys the physical laws of the universe, in that, to stop, reverse thrust must be applied. Slowing down can thus take some while, and all the time the engines are engaged, precious energy units are being expended.

Having successfully negotiated space and found a planet (avoiding the occasional meteorite along the way), the landing must be accomplished. Again, two screens are traversed on the way down, with that good old music getting louder as you get near the surface. A bit of arcade action here, as you manoeuvre your craft through tight crevasses and channels, to finally come to rest on the landing site.

The screen display contains, as it does on the Home Planet, a status strip along the top: ENERGY (which starts at 9999, and, as we've seen, is decremented as you use your rockets), the NUMBER of PLANETS left to explore and plunder, the SCORE of course, and finally, the CLOCK — ever present and ticking away. When on a planet, this display also contains the name of the treasure to be found. Bag of Gold is a fairly uninspiring one (although the puzzle associated with it is certainly not!). But there is also a Snake Plant, as well as a Magic Accordion and a Diamond among others.

So now we must explore the planet looking for the treasure that we know must be here.

To take the Bag of Gold as an example, we land after negotiating a complicated network of caverns. Heading off to the left, we find an unslope on the next section of the planet. Moving up this slope, we come to the next section of the planet, which is a bridge across a river — the middle of the bridge has fallen away, and there is no way over, yet! So we retrace our steps, back past the spaceship. To the right of the spaceship, we find two coloured blocks of light, moving from left to right in the sky, of all things. At that moment, they are different colours. We still have a gun, and in the absence of anything else to do, we may as well have a few potshots at them. And, would you believe it, but they change colour as we hit them! Now there is a clue.

So, let's try to make them the same colour. If we have a good eye, and a true aim, we will eventually accomplish this — but nothing spectacular happens. Wait a few moments, and still nothing happens — the blocks

just move serenely back and forth in the sky.

Our little ship is sitting there, and seems to be unchanged, so we'll carry on to the up-slope. Again, nothing has changed here — wait a moment, what's this? A great key floats gently down from the sky, to settle at our feet. Now we're getting somewhere. We won't pick it up just yet, but instead, go back to the coloured blocks, and see if some other colour combination will bring forth other goodies from the sky. To cut a long story short, we will find ourselves with a chest—locked, of course, but we have a key too, haven't we? Eventually, we will be able to bridge that gap, and get on to find the Bag of Gold.

This is actually one of the easier problems to solve — the planet that contains the Candelabra holds a very complicated puzzle consisting of a lift, a detonator and a crab, as well as rings and cages. It'll take some time to crack the secrets of this particular planet.

The graphics are rather blocky, but the conundra present in *Journey to the Planets* make it good value for money. It will take some time to solve all the puzzles, though these are rather more important than the arcade action, and, once they're all solved, the program loses much of its interest.

However, the game is a very concentrated form of adventuring, in which quick reactions and a sharp eye lend a new dimension to the usual puzzles.

Two more games, from the same stable as *Journey to the Planets,* JV Software (and they seem to have done nothing else since writing these three classics), combine arcade action and puzzles in a mind (and finger) bending exercise.

*Ghost Encounters* and *Action Quest* both take a similar format. You, the player, control a small ghost that looks, sometimes, like a little white tadpole complete with waving tail.

At the start of *Action Quest* this tadpole is placed in the middle of an innocent-looking suburban lounge, complete with table and chairs, and a little light-bulb hanging from the ceiling. At the top of the screen is the number of lives (ghosts/tadpoles) remaining — there are ten to start with — along with the score, which is reckoned from the number of treasures you have found, and finally, a clock that starts ticking away as soon as the display is drawn. Four doors beckon invitingly from the corners of the room.

These are like red rags to a bull, for adventurers! To open one, we simply move onto it, and we are immediately in the room beyond. This is where the fun starts. To take one of the rooms at random, we find a room with one or two walls, and a box in one corner, containing a lovely bit of treasure. As we move into the room, we notice three things. At the bottom of the screen display, where LEVEL 1 was a moment before, we can now see another legend. This will change with each new

room, and will give us, we hope, some small clue to the puzzle that must be unlocked. This particular room is entitled: *Boxed In*. Well, the treasure is indeed boxed in — the clues are not always so obvious! The next thing we notice is that the clock at top right is still ticking away relentlessly, and if we stand still long enough, about a minute, our little surrogate ghost will rapidly fade away to nothing, and we will have lost a ghost-life (a contradiction in terms?). So we must get moving and get that treasure. And this is when we notice the next, and most unnerving thing. We are not left in peace to just walk in and nab the treasure — that would make for a pretty boring game. The spice, as in most arcade games, is an alien — in this room, just one, but other rooms may contain a couple of beasties. They all look suitably horrific! This particular nasty is fairly slow-moving, as this is the first level — how do we get rid of him?

Well, we could just work our way round him, but of course, he is not going to let us do that, and doggedly pursues us wherever we go, getting even closer until. if he touches us, we lose one of our precious lives. So, we must find some way of stopping him. Luckily, we have our trusty Atari joystick with us, complete with fire button. Although the aiming procedure is rather awkward and seems sometimes to be deliberately difficult, this particular monster, being a bit of a lumberer, is fairly easy to get a bead on. Pressing the trigger will turn us for a second into a little revolver! Having zapped the alien, turning him into a frozen shadow, we are at last free to investigate the treasure.

It is still boxed in a seemingly impenetrable receptacle. No other objects are apparent in the room, so we are on our own. Keeping an eye on the clock, and our ghost's luminosity, we must do the obvious thing, and shoot at the box. Lo and behold, a small gap appears where the bullet has passed through the wall of the box. Enlarging this, we can finally squeeze through and nab the loot. Quickly, we then exit the room, and find ourselves back in the cosy little lounge, with now three doors.

Stepping through door No. 2, the legend we see is: *The Magic Touch.* What we see is the treasure lying between two luminous blocks. No aliens here, thank goodness! It looks like a piece of cake, doesn't it? Moving our ghost up the screen, we pick up the treasure — to be crushed as the two blocks hurtle together, with our ghost trapped between them. Another ghost-life lost, but we have learnt our lesson, haven't we? Again, there are no other objects around for us to use, so once more, we use the fire trigger. Aiming at the blocks makes no difference — they are encased in bullet-proof channels. So we must try the treasure. And, yes, hitting the treasure does indeed push it back — out of the path of those two blocks. Firing rapidly, we can eventually

move the treasure far enough so that we can pick it up without being squashed.

Back in the lounge, two doors remain. Investigating these, we find one called: *Looks Too Easy*, while the other is called: *Bountiful Bullets.* Both descriptions are apt.

The first certainly does appear to be a walkover, with the treasure sitting quietly on a table waiting to be picked up. But as soon as we do so, a bird appears right next to us, and comes after us. You may be sure that its very touch means goodbye to yet another life. And at the same time, another alien appears near the door, and he too starts after our ghost. The walls in this particular room are rather twisty, making it hard for us to negotiate the route back to the door, and the aliens are making a pincer attack.

The final room, *Bountiful Bullets*, comprises several parallel corridors, with, this time, no treasure, but instead, another door at the end of the corridor network. Making our progress a little harder are the bountiful bullets. There is just enough room to squeeze between their trajectory and the corridor walls, and if we're careful we'll eventually get down to that door. Once there, we are through to another little room, called: *Disappearing Doors.* The treasure is in one corner, with no alien guards, and ready for the taking. As soon as that is accomplished, a small door appears, and we make our way toward it. As we get near, it disappears. Well, we should have known, from the title. And luckily, as that door goes, another pops into existence on the other side of the room. So, a cat-and-mouse game! Eventually, we'll be quick enough to get a door before it disappears, and we are finally back in the starting lounge, with no doors and a score based on the treasures that we have found.

That is Level One over with, and we can now move on to Level Two. The problems now get harder, and the aliens faster and meaner. Each room is unique, and the puzzles contained in each one, completely unlike the others. And the factor that keeps the player returning to the program again and again is that, although the puzzles may be solved, the arcade element ensures lasting interest.

*Action Quest*'s sister program is *Ghost Encounters.* The format is familiar now, but with a few little twists.

Our starting point is a square, featureless room — gone are the chairs and tables. Again, there are four doors, but this time, two are sealed with a huge padlock. Just as well, then, that this time we have a few extra aids available to us via the keyboard. We can use the first of these straight away, in order to open the padlock. Pressing the K key turns our little ghost into a key, which of course, will open the two locked doors. Returning to ghostly form, we can enter the unlocked door — other metamorphoses we can undergo with the relevant keypress are:

T for TORCH
H for HAMMER
S for SPADE
and
M for MAGNET

All of these manifestations are useful in any one way or another as we progress through the game.

*Ghost Encounters* is a lot harder to crack than the earlier *Action Quest*, and more abstract in design, and there are rooms in which it is difficult to know where to begin getting the room to yield up its secrets.

The two programs together, although they have been around for some while now, both contain puzzles which would grace the best of text adventures, while the arcade action should satisfy the itchiest fingers.

Although all these games from JV Software reward repeated playings, and will not be consigned like many other games to the back of a deep drawer, the arcade flavour of *Ghost Encounters* and *Action Quest* is rather sedate, and that of *Journey to the Planets* even more so. Are there any games that combine a good strategic game with a frenzied arcade action?

Fortunately, there are. Before we get on to these, though, let's look at some games that offer a lot of action, with at least a hint of adventure-type puzzles.

Many adventurers would not agree with me about my next choice — their credulity may have been strained, in fact, by the last three games, and by my belief that they are basically adventures, but let's see if I can justify my beliefs.

All arcade games have some element of stategy in them: they all require a special technique in order to be completed properly. Many books, for instance, have been published, showing the frustrated *Pacman* player the correct way to manoeuvre round the maze, and monthly magazines are devoted to letting mere 10,000 point-scorers in on the arcane secrets of the million-plus scorers.

These games, however, the Pacman, Defender, Gorf, Scramble and all the shoot-as-many-alien-monsters-as-possible-in-ten-minutes-then-find-your-way-through-the-maze/complex-of-caverns derivatives rely on a speeding-up of the process, or a gradually more complex screen layout, or a simple increase in alien numbers and speed, in order to give the player a more difficult task on each successive screen.

Arcventures set more and more complex tasks, which may or may not be combined with the usual alien hordes, for the player to solve. Once solved, he can go on to the next screen, and another, different puzzle.

I believe it is no coincidence that the top games, on both sides of the Atlantic at the time of writing, are what I would call arcventures. Space

zap games, will not, of course, lose their popularity — there will always be a big market for them. But now, after the first flush of enthusiasm for an arcade in your front room is over, computer users are looking for games to tax their mental reflexes as well as physical reflexes.

Games like *Miner 2049'er* and *Jumpman* in the States, and *Manic Miner* (for the Spectrum) in Britain, have become such big sellers for precisely this reason. And I think that we will see many more programs like these — and I hope they will not all be imitative, but innovative in some way.

Some way in the future, as hardware prices fall, we will certainly see network adventures, in which several players, not necessarily in the same room or building, will combine forces or battle against each other.

Back to *Jumpman* and its like. Epyx is the company that has created this masterpiece, and they have enjoyed a huge success with it.

A distant relative of *Donkey Kong,* one of the classic arcade games, which has bred countless versions, both official, and not so official, *Jumpman*'s basic scenario, like those of most of the arcade games, has a rather tenuous link with the actual game. The thirty levels of Jupiter Headquarters have been sabotaged by the Alienators, who have overrun the building, planting bombs wherever they have been. You, trained as the government's top secret agent, have the necessary speed and skill to manoeuvre through the girders and ropes of the Headquarters, find the bombs and defuse them. And, of course, you have been granted immortality (well, up to the seventh life, anyway!).

The game has several options — you may choose from the Beginner level, the Intermediate or Advanced, the Grand Loop or the Randomizer. Beginners only need to cover levels 1–8, and Intermediate, the next eight levels, while choosing Advanced will condemn you to roaming the hardest levels, from 19 to 30. The Grand Loop option will take you through all thirty levels in sequence, with the Randomizer making a random choice for you.

Having selected the option of playing as a Beginner, the first level is retrieved from disc. The display scrolls down to show a network of ladders and girders, some with ropes hanging. Your little Jumpman is ready and waiting for your instructions. Along the bottom of the display is the usual status report of lives left, the score and Hi-score and a readout of your bonus points. The bonus ticks away as time passes, so speed is of the essence if you wish to collect the additional points. Some levels are so hard that bonus points are not awarded at all.

In addition, the name of the room is displayed. Taking a leaf out of the *Ghost Encounters/Action Quest* book, these names sometimes offer clues to the problems to come, but are more often straightforward descriptions. Level 1, which is where the beginner starts, is called 'Easy does it', and as befits an introductory passage, is indeed rather easy.

Dotted around the layout are several little footballs. These are the bombs, and the Jumpman defuses them by simply moving on to them, when they will disappear. The object is, usually, to clear them completely, after which you will go on to the next screen.

The Jumpman has the unique property of being able to leap great distances, with merely a flick of the joystick, but you must be careful how you manipulate him — the slightest miscalculation in take-off, or the wrong choice of landing-spot, and down he will go, base over apex, toward the hard ground below, bouncing off girders, all the while accompanied by a Keystone whistle.

This first level is a simple race against the clock, and the puzzle here is to find the quickest way of defusing all the bombs and getting on to Level 2.

Later levels, of course, get harder and the puzzles much more complex. And the Alienators (remember them?), are not content to sit there watching you neutralise all their hard work — they will come after you with everything they've got! The first of the defences you'll meet is the automatic bullets, and these will quickly become a nuisance, appearing on most of the levels. They move slowly across the screen, until they draw level with your Jumpman, when they will suddenly speed up and head straight toward him. There are often two or three of these misguided missiles around the screen at once, and the poor Jumpman has to watch them all. That's easy you cry: well, it might be for a seasoned arcader, but of course, that isn't all that there is to it. As well as the bullets, which seem to have a canny intelligence and trap your Jumpman where he cannot avoid them, there are the other hazards. I didn't mention them? The guardians of the girders include mean robots, blood-sucking vampires, falling masonry and Space Invaders, as well as, would you believe, dragons! There are methods of tackling each of these hazards, but these problems are basically of the arcade type — shoot 'em as quick and as often as you can, in order to rack up the points.

The adventuring slant comes in working out the puzzles of each room — and most of the rooms have a unique puzzle associated with it, a certain sequence of girders to be climbed, or a certain order of defusing to be adhered to, in order to be successful. Sometimes, a section of girder will act as a trap, which causes another part of the structure to fall away. This will not necessarily make that girder impassable, but it will almost always become so! Defusing a bomb out of sequence will sometimes have the same effect, and sometimes will cause an impenetrable girder to appear around other bombs. There are many puzzles like this — my favourite being the mystery mazes, of which there are two kinds.

The whole screen of a mystery maze, once loaded from disc, is a

uniform light blue. Your Jumpman is at the bottom left, and that, and the section of girder he is standing on, is all that you can see. Move him, and a small section of the surrounding girder-work becomes visible. The whole screen must be explored and mapped in this way. Easy enough, while your man is edging carefully along girders, with solid steel beneath his feet — soon, however, he will have defused all the bombs thus reached, and will have to go further afield. This means jumping off those safe girders, into void, and making an educated guess as to what may lie at the end of the jump — a welcoming girder, or another slow plummet? There may be a rope, of course. These come in two varieties, up and down, distinguished by different colours. Your man will automatically climb or descend — the only way off a rope, once on it, is to jump (making sure there is a safe landing place), otherwise, stay on until the bitter end which, again, may be a safe girder or a long drop to the ground far below.

As the levels get higher, so the problems get harder — as in most arcventures, once each problem has been solved, we are still left with the perennial arcade problem: how to beat the machine and score more points (and *Jumpman* has a high-score, save-to-disc routine). Actually, the number of levels, and the number of problems contained within *Jumpman* are sufficiently numerous for us to need an awful lot of practice before becoming adept at progressing through the levels easily.

*Jumpman* is merely my favourite of a growing list of maze/climbing games, which includes *Miner 2049'er,* from Big Five Software, which features Bounty Bob working his way through a mine — along with *Jumpman*, probably the most successful of all the post-Kong games.

As I said before, the reader may disagree with my insistence upon applying the name adventure to these games, which may seem to be obviously arcade games. But that is what they are essentially — animated and rather light-hearted versions of a text adventure, which keep the player amused and confounded and returning time and again for : *JUST ONE MORE GO!*

# CHAPTER 5
# Dungeons and Dragons

We have met Dungeons and Dragons before, near the start of the book. There is a vast range, now, of role-playing games. These are so called, because the player (or more often, players in the plural), take on, temporarily we hope, the role of the character they are controlling in the game.

The microcomputer is ideally suited as a housekeeper in this sort of game — rolling the dice, keeping track of positions as well as holding all the information about the various hazards that the players might stumble across. How much easier for the Dungeon Master to merely tap a couple of keys and see the information on his monitor, rather than be surrounded by countless scraps of paper!

As a housekeeper, then, the computer is extremely efficient — but this use of its powers is not so efficient. It is in the solo D&D game that the computer really comes into its own.

There are many programs that combine adventure with Dungeons & Dragons — the mix being in various proportions.

A new program, released in the States as I write, is *Zombies* from Bram Inc. This would appear to offer a real D&D scenario, with a 'party' of adventurers (well, two) taking part in exploring a weird landscape, brilliantly executed in 3D scrolling.

This is one of the few games available for the traditional D&D 'party' of two or more adventurers — probably the most well-known and established is *Ali Baba and the Forty Thieves*, from Quality Software. *Ali Baba* allows up to four players to participate — in glorious high resolution graphics. Playing is possible either from the keyboard or by joystick — the joystick option is probably easier, with commands being input entirely by moving the joystick one way or the other in response to on-screen prompts. The action takes place in a matrix of rooms, and as each room is entered, a moving cursor picks out each exit. Although this slows play down rather considerably at the start of the game, later on, it is taken for granted that you will remember the possibilities in each room.

Moriana and Abdulla are both held captive somewhere within the dungeon complex — if you rescue them, they will help you in your fight

with all the monsters and other assorted baddies. And this is what this RPG is all about — monster-bashing! There seems to be some sort of baddy at every turn; in fact it becomes quite tedious to get knocked off regularly at first, until a bit of experience is built up, and combat becomes more evenly-matched.

You are not always on your own, however, as baddies may be commanded to launch into each other, and may in fact do so anyway, if the player leaves them alone for a little while. Outside help is sometimes available in the guise of The Unicorn, who occasionally appears to help you against your foes. The game may be played by up to four people, who then do battle with each other, as well as the monsters (there are more than 100 of them, apparently) programmed in to the game.

Although the graphics are not stunning, Quality Software, the authors, have re-defined the character set to a beautiful ancient Arabic script, which is beautiful to look at, but rather difficult to read. The music, too, is rather nice, being a tuneful rendition of Rimsky-Korsakov's Scheherazade.

Avalon Hill, the great board wargame producer, has now been in the business of computer software for some years. Their programs are mostly computerised versions of the favourite board games — again, here, the computer is an ideal companion who keeps scores and moves around all those little counters. Any wargamer will be pleased to be relieved of the minutiae of play, not to mention the agony of watching helplessly as a large stack of cardboard counters is knocked over, thus ruining a three-hour build-up along the eastern front.

Avalon Hill has, however, one or two programs that will interest us. *Lords of Karma* and *Empire of the Overmind* are straightforward text adventures, and have been dealt with elsewhere — *Telengard* is a program for D&Ders.

Designed by Orion Software, *Telengard* is a real-time fantasy role-playing game. It comes impressively packaged in a large colourful box, which belongs to Avalon Hill's uniform range of bookshelf containers. Inside is the diskette/cassette and a detailed 20-page manual. Lift the tray as requested, and you'll find a pile of bumph about AH's complete range of software. The manual is very detailed, and complete with illustrations of the various monsters that you will meet. Things get a bit confusing when the manual tries to cover all the different versions for the Apple and TRS-80 as well as the Atari.

If you are entering *Telengard* for the first time, you will start, like all novice adventurers, at Level 1. If you have elected to start with a new character (you may, if you like, LOAD in a previously SAVEd character), the computer will slowly flip through a series of six attributes. Anyone familiar with D&Ding will recognize these — they are:

STRENGTH
INTELLIGENCE
WISDOM
CONSTITUTION
DEXTERITY
CHARISMA
HIT POINTS

These are standard throughout fantasy role-playing games, and mean basically the same thing from game to game.

STRENGTH is the measure of a character's strength, and will determine his chance of success in combat

INTELLIGENCE is important in casting spells

WISDOM is important in casting healing spells, and also spells to deal with the Undead creatures

CONSTITUTION is used to determine a character's Hit Point total

DEXTERITY is used to determine a character's chances of running from unwanted combat

CHARISMA is used to determine the reaction to a character, of certain monsters

HIT POINTS are related to CONSTITUTION, and are decremented in combat — if they reach zero the character is dead.

As the list of attributes flip through, the player has 3 or 4 seconds to decide whether or not he wants this particular set — by pressing RETURN, those attributes currently being displayed become those of the character.

After these preliminaries, the display is drawn. The entire program is in BASIC, and the drawing is very slow, and done section by section as your character-token moves around the dungeon complex. Movement is accomplished via the keyboard, using the keys A (for west), W (for north), D (for east), and X (for south) — S means STAY, and the character marks time where he is. The character, if he moves, does so one square at a time, and after each movement, there is the possibility of an encounter. If a monster is indeed met, the player has three options open to him: if he is fairly strong, he may decide to fight, and then of course, it is down to a straightforward matching of the monster's abilities with those of the character.

Monsters come in two basic varieties — the living and the undead. There are twenty monsters to deal with in *Telengard*'s dungeons, from the whimpish Gnoll, who is 'a sawed-off freak, not too tough but nasty nonetheless. Lowlife of the dungeon', to quote the manual, through the standard things like the Zombie, the Mummy, the Specter and the good old Vampire, all the way to the most powerful of all — the Dragon, 'King of the nasties. Like to barbeque young warriors with a small gust of their disagreeable breath'.

All monsters, like the player-character himself, have experience points. The character gains his from bashing monsters: depending on the monster's experience level, he gains a certain number of points when he defeats, or 'turns' the monster.

Should the character be more dexterous than strong, he may elect to evade the monster. If he is wise and clever, he may elect to cast a spell.

For me, this is the most interesting aspect of *Telengard*. The spells are many and varied — some will work well with living monsters, while others have more effect on the undead. The spells are divided into six levels, each of which levels contain six spells. A character at Level 1 or 2 (of experience), only has available Level 1 spells, and as he gains more experience points, so he is given the chance of employing more spells.

It's a pity that the manual only goes into detail about the twelve spells of the first two levels — leaving the adventurer to find out for himself the effects of the others. There is a list, in brief, of the other four levels.

To cast a spell, the player must have the requisite amount of spell units, the amount required being one SU for first level spells, two SUs for the second level spells, and so on.

The names of the spells make for good reading! Starting at Magic Missile, on Level 1, we go through Protection from Evil, Cause Light Wounds, Cause Serious Wounds, Finger of Death, Wall of Fire, Time Stop, and Word of Recall all the way up to Restoration and Prismatic Wall on the highest level. These last few, along with others like Power Word Kill, Phantasmal Force and Raise Dead, sound rather interesting, but I haven't as yet gained enough experience points to be able to use them!

As you can see from the names, not all the spells are used for zapping monsters — they can be used to heal wounds, set up impenetrable barriers, and so on. Other spells enable the player to 'see' into places that would otherwise remain dark to him. And, when being used to zap, the effect of the various spells on different monsters is not always the same. The broadest and most obvious difference is between the effect of a certain spell on a living monster, and the effect of the same spell on an undead monster.

Combat, which includes spell-casting, is carried out in real-time. The computer waits a very short time (the manual says five seconds, but it

seems like two at the most), and, if you haven't responded by then, will carry on — the monster will then attack you anyway. You may be unlucky in that the monster will attack you first, anyway.

Although all this combat is the most exciting part of the proceedings, there are many other things that the adventurer can get up to while prowling around the dungeon.

There is a lot of treasure, for instance, just waiting to be picked up. But you have to be quick! *Telengard* is a real-time game, and picking up treasure, like combat, is achieved by relying on your instincts and reactions. The computer gives you, as in combat, a mere three or four seconds to find the right key on the keyboard. Pressing RETURN will add the treasure to your hoard. But be careful, the most innocent bauble may turn out to have an adverse effect.

Apart from the straightforward treasure, like silver, gold and gemstones, which all serve to increase your purse (which may be exchanged at any time at the Inn) there are magical treasures also. Some of these are weapons. The Sword, Armour, Shield and so on, have various, fairly obvious properties which will be found during the course of the game, and there are several other items, like the Ring of Regeneration, the Potion of Healing and the Scroll of Rescue, which can be used for one purpose only. The Scroll, for example, when used by the player, will deposit him, free of attacking monsters, at the base of the stairs leading back up to the Inn on the first level.

The program is in BASIC, as I've said before, so lacks a certain flair in the graphics department, but the reaction-testing real-time combat compensates somewhat for this.

# CHAPTER 6
# Software Stars

Over the past year, software authors have begun to receive the personal acclaim that they deserve for the programs that they have written. To know that an adventure is written by Scott Adams ensures that it will at least be interesting and, in the past, has meant that you will have a pretty good idea of the format. Now there are several other authors whose names endow a stamp of approval upon their products. The package is at least worth looking at, and will probably turn out to be worth obtaining.

One of these new software stars is Bill Budge, whose claim to fame, among other programs, is *The Pinball Construction Set*. This fascinating program allows the user to build his very own pinball table, on-screen, and play it!

Budge eschews the anonymity of Scott Adams, who rarely appears in the publicity for his products, and instead can be seen gazing moodily from the double-page spreads in American computing magazines, looking for all the world like a rock star from the pages of *Rolling Stone*. Only recently has Electronic Arts, the company he helped set up, gone into colour advertising — at the start, all their copy was in black and white, which actually enhanced the message. The early EA ads were long on hyperbole about 'a universal language of ideas, something like a smile . . .', and 'Toward the language of dreams . . .'. The software? Oh yes, the details of that were tucked away down in one corner.

*The Standing Stones*, by Peter Schmuckal and Dan Sommers, is a Dungeons and Dragons-oriented game with 15 levels of dungeons and over 200 different kinds of monsters to fight — unfortunately, at the time of writing, it is only available for the Apple, so is beyond the brief of this book. *Archon*, by Anne Westfall, John Freeman and Paul Reiche III, of Free Fall Associates, is, however, written for the Atari, and so we may include a detailed look here.

*Archon*, like all the other packages in the Electronic Arts catalogue, has a superbly designed package, in this case executed entirely in black and white (nostalgia for those misty, moody early ads?). The diskette nestles in its co-ordinated sleeve among a flurry of words, most of which

seem to be about the authors (a kamikaze, a physicist and a writer of fugues, as it says somewhere in the blurb), the thinking behind the game, and its inception. The object of the game is to 'use every Troll, Basilisk and Dragon you have, and all the other slimy, conniving underlings you can find, to fight the Phoenixes and Djinnis of the great and evil Archon, whose Empire stretches as far as the firmament.'

Although the packaging is superb, with its triple fold-out sleeve, beautifully produced manual and reference card, here is what you are told about the practicalities: 'The first step is easy. Insert the disk. Then start playing the game.' That's it! It took me an hour or so to figure out that, in order to play the Dark side (or Player Two), I needed to insert my joystick in Port 2! Obvious, of course, but a few more lines of such mundane instructions would not have been far amiss.

Style, and there is a lot of that present in any EA package, must however be linked with content and, luckily, Electronic Arts scores 100% (and more), for the content of their games. *M.U.L.E.*, another of their new releases which is unfortunately outside the scope of this book, is a version of *Kingdom* — but only just. It is one of the most stylish games I have seen, and is a must. EA also have the obligatory arcade games in *Hard Hat Mack* and *Axis Assassin*. All of them have style in every respect of packaging and programming.

*Archon* is included here because it has a magical blending of chess and fantasy.

The first phase of play takes place on a 9×9 chess board. The player can choose a human opponent, or square up against the computer — and can also choose to play the Dark Side, or the Light Side. This is no mere translation of Black and White, as we shall see later.

Immediately, the player notices dark spots within five squares — one occupying the middle square, and four others at the middle of each outer rank of squares. The object of the game is simple — either to occupy all five squares thus marked, or annihilate the opponent.

Anyone who has played chess once or twice will be familiar with the basic play mechanics of *Archon*. A square, unless occupied, may be passed through, or landed upon, by any piece. The pieces are arranged, at the start, in two ranks on each side, facing each other, and each piece has a certain way of moving, and certain powers. These are fully covered in the Rule Book, and also on the separate Reference Card.

There are two important and major differences between chess and *Archon*, however. The first, which lifts *Archon* above any other chess-like strategy games of modern times, is that the balance between light and dark, the opposing sides, shifts throughout the game. This is manifested primarily in the shading of certain squares as the game progresses. This is the effect of the luminosity cycle, and the squares are arranged in a symmetrical pattern around the board. On these squares,

the light pieces, or icons, are stronger when the cycle is in the light phase, and weaker when the cycle has reached the dark phase — and vice versa in the case of the dark icons.

And this leads in turn to the second major difference — to capture a square, a piece does not, as in chess, simply land on it, removing any piece of the opponent's that happens to be there. In *Archon*, this leads instead to the second phase of the game, in which combat takes place, on a special field, between the two pieces; the victor, who may not necessarily be the aggressor, then occupies the square.

Electronic Arts has an extremely attractive way with attract and demo modes. If the player leaves the keyboard or joystick alone, the program will go into demo mode, in which the icons walk on and introduce themselves.

And here is another difference — the pieces on each side are not the same. The *Archon* equivalent to pawns (only because they are more numerous than the other icons, and correspondingly rather weaker and more vulnerable), is, on the light side, the Knight. There are seven of these, and they move along the ground, and up to three squares in range. Because they move along the ground, three squares is only the theoretical maximum — if they come up against an occupied square, they must stop short, if the piece is from his own side, or decide whether to fight the occupying piece, if it is of the dark side. The Knights are ranged on the second rank of the board, flanked on either side by the two Archers. These are 'fearless Amazon warriors of legendary skill', and have magical quivers that are never emptied.

Behind these relatively expendable (but nevertheless valuable!) icons, are ranged the heavy mob. As in chess, the outer three pieces are duplicated on both sides of the board on the first rank. The outer pair, on the light side, are the Valkyries. These two are flying pieces, and again can cover up to three squares — in contrast with the ground-moving icons, the fliers can move over occupied squares. They carry magical spears, which, when thrown, return to their hands.

The next pair in are the Golem. These are pretty slow (what do you expect from an artificial being shaped from stone and gleaming metal?), and their weapons are huge boulders. Inside them are the two Unicorns. These, pretty obviously, use their horns as weapons.

Now we come to the innermost three icons, the major one of which, and the light side's most valuable piece, is the Wizard. He is flanked by two very powerful pieces; on his right stands the Phoenix. He is a flaming bird of immense size. Until now, all the characters we have looked at have used projectile weapons of one sort or another — the Phoenix surrounds himself in combat with a seething mass of fire, scorching anyone on the perimeter of the blaze, and immolating any opponent foolhardy enough to be caught near the incandescent core. He

has a long range of five squares, and can fly. On the Wizard's left is the Djinni, who is a flier with the slightly shorter range of four squares. He is a magical being from another planet, that of tempest and storm. Like the Phoenix, he does not use a hurled weapon, but rather summons up a tempest to confound the enemy who gets too close.

The dark side has similar forces. Instead of Archers, you will find seven Goblins in the front rank — they wield gnarled clubs, and can be very effective if used properly. On either side of this column of Goblins, is a pair of Manticore. They resemble a large golden Lion, with a human face and a scorpion's tail. The creatures can hurl the three spikes from this tail over their heads at the enemy. The outer squares of the backward rank contain a pair of Banshees, undead souls that drain the life from an opponent with a keening wail. The Banshee is a flier of three square range.

Next in toward the middle is a pair of Basilisks. These are relatively short-lived, but have a piercing stare that will lay low any opponent who comes within range.

The innermost icon on the dark side is the Sorceress, and to her left is the Dragon. This is an extremely powerful creature, as befits the most well-known assailant of adventures! It is a flier, naturally, although not of such a long range (only three squares), but it is so terrible on the combat field that much thought is necessary before deciding to tackle it. Its mode of attack, of course, is its fiery breath. To the right of the Sorceress is one of my favourite characters, the Shapeshifter. In combat, this will assume the shape and characteristics of the piece ranged against it.

Now to combat!

As soon as any piece, in its turn, moves onto a square occupied by one of the opponent's icons, battle is the only way to resolve the conflict and decide which piece should remain in possession of that square. The checkerboard disappears, to be replaced by the battlefield, which is studded with a random placing of several obstacles. In much the same way as the main playing board cycles through from light to dark and back again, these obstacles also go through their own cycle — of solidity. That is, they go from being totally impenetrable, so that the battling icons may shelter behind them, and gradually become more transparent, in which case the icons may move or shoot through them, but will be slowed down, or the missile slowed down, in its passage. Finally, the obstacle will become totally transparent, at which time it will become effectively invisible. Then it will start cycling through to solid again.

On either side of the combat arena is a Lifebar. These show the energy remaining, for each icon. As wounds are sustained, so the energy is shown, gradually ebbing away. Once the Lifebar disappears, of

course, the icon is dead, and the victor takes possession of the square. In *Archon*, there is no possibility of retiring for a while to lick wounds — it is a fight to the death!

Now to the two main opponents — these two are like the Queen and the King of Chess, combined in one piece. On the light side is the Wizard, an ancient man of supernatural power. In battle, he casts devastating balls of fire. To risk him in combat, however, is rather irresponsible, as he is no more powerful in this respect than many of the icons which will be used against him (and a Shapeshifter, of course, will effectively become another Wizard). His great strength, and this puts him above all the other icons on his side, is in his use of spells.

There are seven spells:

*Teleport* which moves any one of the Wizard's icons any distance from one square to another.

*Heal* which will heal any wounds sustained by any one of the Wizard's icons.

*Shift Time* will reverse the flow of time — in other words, will cause the luminosity cycle in the Wizard's favour.

*Exchange* will cause any two icons (of the Wizard, or the Sorceress) to change squares.

*Summon Elemental* will allow the Wizard to bring on, for one move only, a temporary icon representing one of the four elementals — earth, fire, air and water. Once combat is over, the elemental disappears for ever.

*Revive* will resurrect an icon which has previously been killed in combat.

*Imprison* will imprison, on its square, an icon of the dark side — the effect will last until the luminosity cycle returns to its dark phase. The icon can still take part in combat.

All these spells may be used only once, so, as each one should be used at the most propitious time, great care must be taken in employing them. The Sorceress has exactly the same powers as the Wizard, and the spells have exactly the same effect.

What we have in *Archon*, then, is a unique blend of strategy and arcade game that does justice to Electronic Arts' reputation as a rising star in the software firmament.

Why have I included *Archon* in this book of Atari Adventures? After

all, it is pretty far removed from the graphic adventures we have looked at, and even further from the text adventures! But *Archon* points the way toward the sort of Adventure that we will be seeing in the future — based on fantasy figures, and with a grounding in a Dungeon and Dragon style of combat, at least in the capabilities of the individual characters. *Archon* is one of the first, and best, of this new type of adventure, which will exist very happily alongside the more traditional programs.

# CHAPTER 7
# Meet the Cast

We can start having some fun now, and look at the monsters, treasures and locations that might be met in the typical adventure.

## Monsters

The term 'monsters', for our present purpose, can be applied to any character in the program that is out to do us harm — and they are not always recognisable as such!

Like the programs themselves, we can see distinct categories here. The classic adventure, descended from the original mainframe *Adventures*, contains fairly passive monsters which tend to sit there, waiting for some brilliant strategem from the player to scare him away, as the snake did in our introductory scene at the start of the book. Or we may have to avoid the monster by finding a way around him.

The evil dwarves are a notable exception in the *Colossal Cavern* adventures. They appear occasionally to throw a weapon at the adventurer. The first dwarf throws an axe. This must be picked up by the player and then thrown at the succeeding dwarves. They are all hurling knives, after the initial axe, but if you remember to keep retrieving the axe, you should have no trouble in surviving their attacks.

In Level 9's version of the classic, an endgame is included, that extends the original in some 70 new locations. And here you can really get your own back on those little dwarves. By dropping dynamite near a crowd of them, you can score many points. Incidentally, during this finishing sequence, you can also score points by saving from death a number of elves.

As played originally, on mainframe computers, after hours, the printer was often the only means of seeing what was happening — so blow by blow combat in D&D style was not really feasible.

This leads us to the next monster category, which we find in the action or arcade games. In this category, the monsters are extremely active, and definitely out to get you. As we have seen, the combat system from the role-playing games like *Dungeons & Dragons* is often used in these

games, as exemplified in Avalon Hill's *Telengard*. And the monsters themselves, as befits the ancestry of this type of game, are of the leg-ripping, skull-crushing, heart-stopping type commonly met in RPGs. There follows a list of some of these lovely things, along with brief details of their attributes, and origins. I've put them in a very subjective order of fiercesomeness — so, if you meet a Balrog, you would usually treat it with rather more respect than, say, an Orc.

*Dogs*
Very lowly but also very vicious monsters which usually attack in pairs. Only the most rudimentary of weapons is needed to fight them.

*Wolves*
Even more vicious than dogs, and often more cunning.

*Dwarves*
Appear in the original *Colossal Cave,* as well as all the programs that have since based themselves on this classic. They usually jump out of the shadows, throw an axe or dagger, and disappear again — usually of nuisance value only!

*Orcs*
The Jack-of-all-trades monster — appears in most Tolkienesque adventures. Extremely unlikeable creatures, very spiteful and ugly. They hunt in groups and wield spears or scimitars. There is an Orcish Archer in one of Level 9's games.

*Waug*
This lumpish creature appears in the book *The Hobbit,* and also in the adventure of the same name from Melbourne House. Maybe one day this super game will be implemented for the Atari.

*Snake*
Most programs featuring a snake do so to create a puzzle rather than to pose any physical danger — see the first chapter.

*Birds*
Often to be found at the top of mountains, sitting on Golden Eggs. Find a way to frighten them off before attempting to purloin the treasure. Birds also come in handy against snakes — see Chapter 1.

*Elementals*
Come in four types: Air, Fire, Earth and Water. You will need magic of a sort relating to the elemental before attempting to fight. Electronic Arts, in their program *Archon*, use elementals to great effect.

*Vampires*
No need to tell you how to fight these! Before coming across them, you should have picked up any requisites at other locations. If you come across some garlic, don't turn your nose up at it — you can be certain that there will be a vampire not too far away!

*Mind Vampires*
A special sort — they're not after your blood!

*Ghosts*
You won't need a sword or spear to battle with these!

*Werewolf*
Only magical weapons will be of any use against these, and preferably made of silver.

*Goblin*
Small, ugly creatures, which delight in tormenting their victims by prodding their legs with sharp sticks.

*Hobgoblin*
Larger, more dangerous and cunning than their animal-like half-brothers.

*Harpy*
Winged creature of amazing strength and agility.

*Siren*
A sea-faring harpy — usually found basking on rocks near the sea, singing. Rock music was never like this.

*Troll*
Devious, greedy, oafish creatures. Can absorb a lot of punishment — but they quite often don't take too kindly to light.

*Barrow Wight*
Ghostly apparitions that populate the Middle Earth wilderness.

*Centaur*
Half horse and half man, armed with a bow and arrow, of which they are masters.

*Fire Imp*
Little, agile flames of nuisance value.

*Fire Giant*
Large dangerous flame.

*Thunder Lizard*
One of the most potent of monsters. Only tackle if you are well armed with conventional weapons, and have a fair amount of dexterity.

*Sand Worm/Purple Worm*
Extremely dangerous! The sand worm is basically a mouth with a 60-foot stomach behind it. The Purple worm is similar, but with eyes, and not confined to the sand.

*Minotaur*
The well-known bull-like creature. As dangerous as you would expect a highly-intelligent bull on the rampage to be.

*Wyvern*
Another winged creature, this one comes equipped with vicious fangs and razor sharp claws.

*Balrog*
One of the most dangerous of monsters, much beloved of adventure writers.

*Dragon*
Probably *the* most dangerous of all monsters — only the strongest and bravest of adventurers should approach this monster; or the most devious — Dragons don't always want to roast you.

*Lich*
You'll have occasion to meet one of these later in the book, so be warned — they are extremely powerful. They are former wizards who have died and brought themselves back to life to wreak havoc.

This is, of course, only a partial list of the monsters you may expect to meet. Most adventure programs will feature some of these, and more of their own. In writing your own adventure, your imagination can be the only restriction. Just about anything can be pressed into service in a game, and I have seen everything from snowmen to London double-decker buses being used as death-dealing enemies.

Most programs that rely on a *Dungeon & Dragon*-style combat system will keep you informed of your current physical status. This may take the form of physical points, combat points, or food points. You may, indeed, be given a combination of all these, but it will be clear to

you when playing, that a decision on whether or not to fight with a particular monster must be made by you, taking into account your own strength, using whatever system the program adheres to, together with your own knowledge of the monster's own rating.

Of course, you may be given no forewarning of a monster's presence, and thus have to fight whether you want to or not.

This sort of program would be a very unfair one, and they're not common — most games will give you both a certain pre-knowledge of the monster's capabilities and a method of escape should you wish to decline the challenge. You may not, though, make good your escape, if the monster is a particularly speedy one!

## Spells

The sort of combat we've spoken about so far in this chapter, with the system of strength or combat points, is, of course, based on the physical side of combat — that is, a few well-aimed chops at the monster's head with your broad-sword or morningstar. Many programs, however, give you the option of employing magical powers, and casting simple spells at a monster. This idea is very well-entrenched in *D&D, Tunnel and Trolls* and other RPGs, where many pages of the rulebooks are devoted to complex spells, which become ever more complex the longer you survive as a wizard.

The basic idea of a spell remains the same, however — to ZAP the monster with a well-aimed spell. This can take the form of a simple SLEEP spell which lays out the monster for a certain amount of time, to the more complex spells seen in *Telengard.*

## Weapons

At the start of the classic adventure, you are weaponless. You have to find your own armoury during the course of the game, and it will probably consist of not much more than a short sword, or dagger, or axe and these are usually thrown at you by the occasional passing dwarf.

These weapons come in handy when dealing with the dwarves, and you'll get points for killing them, but you will probably not get much of a chance to use them against dragons, or sirens, or other monsters. No, they are to be beaten by guile — and you'll find that the Big Bad Pirate who, every so often, rushes in to steal all your hard-won treasure and rushes out again to hide it, disappears long before you can throw any axe you may also be carrying!

So we have to look elsewhere if we are to use a lot of lovely, blood-dripping swords!

Although the Dungeon and Dragons system, with its swords, maces and so on, is a fertile ground, well-used by games-writers, there are several adventure programs available for the machine which make use of more conventional weapons.

The weapons you will meet, and be able to use, in adventures are varied in the extreme — weapons can be made of anything, and are really only limited by the writer's imagination.

Fighting monsters with deadly weapons, however, is not really the ultimate aim of the traditional adventure game — sorry Jason, Fred and Tom of 2C! The monsters are usually protecting something, and that is usually

## **Treasure** (Sometimes heavily disguised.)

We've finally arrived at what is without doubt the real reason for putting up with the frustrations of playing these wonderful games.

Earlier in this book, we looked at *Colossal Adventure*, from Level 9. This adventure, based, in the first part, on the Crowther/Woods mainframe original, contains many treasures to be collected during the course of the game. Each of these is worth a certain number of points, and in the text, or text/graphic game the score is the thing!

Take a look at this situation. You are in a long corridor, in a complex of caves. You have arrived here after collecting treasures and other objects. To find out what you are presently carrying, type INVENTORY (or usually, just INV.). The computer replies:

YOU ARE CARRYING:
The Magic Helm
The Golden Ring
The Brass Shield
The Comic
The Fire Opal
The Brown Gloves
YOU ARE IN A LONG CORRIDOR. TO THE WEST YOU SEE A DARK CAVERN.
WHAT NOW?

Well, we might as well go in — this can often prove dangerous, as the author will probably be testing us, and will set all sorts of traps about his complex of caves. The next message is often:

YOU HAVE SLIPPED IN THE DARK AND BROKEN YOUR SKULL. DO YOU WANT ANOTHER GAME Y/N?

But we have been warned in advance of the lack of light in the cave, and we have not been told of any light source in the corridor, so it is a reasonable assumption that there will be light of some sort in the cave itself. Having first SAVEd our position (any reasonable program will allow you to do this before any risky steps are taken, thus making it easy, if killed, to quickly resume), we decide to take the risk of looking into the cave. We type:

W (est)

and the computer answers:

YOU ARE NOW IN THE DARK CAVE. A FAINT LIGHT GLOWS FROM THE ENTRANCE BEHIND YOU. BETTER GET SOME LIGHT QUICK.
WHAT NOW?

Try this:

READ COMIC (for some light relief, geddit?)

Ingenious, but no good, I'm afraid! The computer merely repeats its message:

ITS DARK IN HERE. BETTER GET SOME LIGHT

This is not too good! Adventure programs often let you stumble about in the dark for only a short time, or a couple of turns, before deciding that you haven't the faintest idea of how to proceed and dumping you unceremoniously in the nearest ice-cold underground stream, or causing you to bang into an overhanging rock. Whatever method it chooses, the program will often draw the game to a sudden conclusion.

But what's this — something else comes up on the screen:

YOU FEEL SLIMY TENTACLES WRAP AROUND YOUR LEGS. IT IS A SMALL OCTOPUS.

Now the octopus may come in handy here. So:

GET OCTOPUS

And the computer comes back with:

UGH — NO THANKS — ITS ALL SLIMY
WHAT NOW?

WEAR GLOVES

O.K. I AM NOW WEARING THE BROWN GLOVES
WHAT DO I etc . . .

GET OCTOPUS

I HAVE THE OCTOPUS. MANY HANDS MAKE LIGHT WORK. A SOFT LIGHT EMANATES FROM AN INVISIBLE SOURCE. YOU ARE IN A SMALL CAVE. THERE IS A SMALL WOODEN BOX HERE. TO THE NORTH YOU SEE AN UNDERGROUND STREAM FLOWING THROUGH A SMALL CRACK IN THE ROCK. THROUGH THE CRACK YOU CAN JUST MAKE OUT SOMETHING GLITTERING IN THE DARK BEYOND.
WHAT NOW?

He! He! Some more treasure! Must get it! Flushed with justifiable pride at our success at using the octopus, we attack the next problem with gusto. How are we going to get at that gold in the next cave?
We can try the direct route, by typing:

N (for NORTH)

YOU CAN'T GO THAT WAY
WHAT NOW?

That's a bit of a blow. The computer, darn it, doesn't seem very impressed with our previous brilliant deductions, and is already putting more problems our way! There must be a way through the crack. Maybe we are carrying too many goodies, and we can't squeeze through.

DROP ALL

O.K.
WHAT NOW?

N

YOU CAN'T GO THAT WAY
WHAT NOW?

How about jumping in the stream, we might be able to swim or float through.

JUMP IN STREAM

YOU MUST BE JOKING — I CAN'T BREATHE UNDERWATER!
WHAT NOW?

Hmm, this is getting silly. Is there anything to help us in the cave? There's a little pile of treasure on the floor which we dropped a few moments ago, but none of that is useful to us now. Wait there's a box, isn't there. How could we have forgotten that?

EXAMINE BOX

YOU SEE A SMALL WOODEN BOX.
THERE IS NOTHING UNUSUAL.

Computers can be very pedantic!

OPEN BOX

O.K. THE BOX IS OPEN. INSIDE YOU SEE AN AQUALUNG.

Aqualung? Underground stream?

GET AQUALUNG

O.K. YOU HAVE THE AQUALUNG
WHAT NOW?

Well, what now? We have the aqualung, and it's a fair chance that we will now be able to negotiate the stream.

But we want to take all our lovely treasure with us, don't we, and water has an annoying habit of flowing one way only. You can bet your last golden ring that we won't be able to get back once in that stream. So, we'll have to go through the long chore of retrieving everything we want to take with us. The computer, though, has other ideas!

YOU ARE CARRYING THE AQUALUNG. YOU CAN ONLY CARRY ONE MORE THING.

Obviously, the aqualung is too bulky or heavy. Let's just recap on what is currently in the cave.

LOOK

YOU SEE:
THE MAGIC HELM
THE GOLDEN RING
THE BRASS SHIELD
THE COMIC
THE FIRE OPAL
THE OCTOPUS WHICH IS WATCHING YOU
YOU ARE CARRYING:
THE AQUALUNG
THE BROWN GLOVES WHICH YOU ARE WEARING
THERE IS ALSO A SMALL BOX WHICH IS OPEN

That's a lot of stuff to carry! Wait a moment, though, the box might be useful. We're told it is small but maybe it will carry everything we want to take with us, and maybe it will float!

Now we can try:

GET RING

O.K. YOU HAVE THE RING
WHAT NOW?

PUT RING IN BOX

O.K.
WHAT NOW?

LOOK

and we get the same list as before, but with a difference at the end:

AND A SMALL BOX WHICH IS OPEN. IT CONTAINS THE GOLDEN RING

It works! Now we'll have to repeat the process with the rest of the treasure. After several moments typing, another LOOK informs us that the box now holds all our hard-won treasure. The octopus, we keep for the moment. Now let's try:

THROW BOX

WHERE?

IN STREAM

O.K. THE BOX IS CARRIED SWIFTLY FROM VIEW INTO THE CRACK IN THE ROCKS

Now let's do the same with the octopus. The computer tells us that the octopus has also disappeared from view. Now we'll type:

JUMP IN STREAM

and the computer tells us:

YOU MUST BE JOKING — I CAN'T BREATHE UNDERWATER!

So boring! We'll have to go through every motion for the computer's benefit.

WEAR AQUALUNG

O.K. YOU ARE WEARING THE AQUALUNG.

I AM SUFFOCATING. QUICK DO SOMETHING!

Whoops! Forgotten something!

TURN ON AIR VALVE

O.K. THE AIR VALVE IS NOW ON.
WHAT NOW?

JUMP IN STREAM

O.K. YOU ARE CARRIED ON THE SWIFT CURRENT OF THE STREAM THROUGH THE CRACK IN THE ROCK, AND INTO A CAVE. THE OCTOPUS HAS GONE BEFORE. MANY HANDS HAVE MADE THE LIGHT WORK. ABOVE YOU IS THE BANK OF THE STREAM.

We breathe a sigh of relief. Now we must get up onto the bank of the stream.

U (for UP)

O.K.
YOU ARE IN A LARGE CAVE.
YOU SEE A SMALL BOX. IT IS OPEN AND CONTAINS MUCH TREASURE.

(but wait a moment before going to retrieve the treasure!)

YOU ALSO SEE:

A DRAGON ASLEEP. HIS SCALES GLISTEN IN THE LIGHT.
ALL THAT GLISTENS IS NOT GOLD! HE IS STIRRING.
WHAT NOW?

What a question! Well, what do we do now?

This is an example of a typical problem to be met in a traditional adventure (I won't tell you which one, in case you are currently playing it) — a series of interlocking problems that eventually yield a result. That result may, as in our example, lead you into a deal of trouble, but of course the possibility is that you will end up with treasure.

Getting the treasure is not, however, the final story, as you have to return to a certain location to deposit the treasure and collect your points.

While this may seem a bit of a chore, as you will have to renegotiate all the perils on your return, there are, in most good adventures, certain routines to bypass this necessity.

Crowther and Woods wrote a certain magic word on a cave wall in their adventure — utter this at the right time (that is, when you have an armful of treasure you want to get rid of), and you'll find yourself, treasure included, back at your original starting point. You may then obtain your score. Other adventure programs actually allow you to carry the treasure-repository along with you! You may have to drop it at some point to allow you to collect treasure (or squeeze through narrow cracks in rocks!) — so don't forget where you left it!

Let's have a look at some typical treasures; and as with my list of monsters, there will be a rather subjective progression of potency.

| | |
|---|---|
| **Helm** | The boring iron sort of helmet |
| **Silver Helm** | A bit better |
| **Golden Helm** | Quite nice |
| **Magic Helm** | This is more like it, although there is an even better one later. |

**Gemstones** Opals, diamonds of course, emeralds, topaz, lapus lazuli — you name them and you'll find them somewhere in an adventure.

**Ring** Again, the boring sort

**Silver Ring** Not bad

**Golden Ring** A bit common nowadays — every adventure seems to have one! The one found in *The Hobbit* is famous for having no perceivable purpose!

**Invisible Ring** Doesn't look much (how could it!) but if it makes the wearer invisible too, then it is obviously useful.

**Invisible Cloak** This is even better than the ring

These last two lead us to a secondary list — of 'found' weapons. That is, weapons that the explorer will find in his travels, scattered about the caverns, rather than the sword that most adventures provide their player with at the start of the game.

**Sword** The standard weapon of almost all adventures, although some allow you to stumble upon swords in your wanderings.

**Torch** A very basic weapon, useful for clearing an area of ants, frightening wolves and so on.

**Club** A bit of a brutal weapon — not very subtle

**Mace** Another basic weapon

**Dagger** A secretive variation of sword

**Kris** Just one example of exotica. The typical adventure author is extremely imaginative when it comes to dreaming up new weapons, and you can meet all sorts of devices you never knew existed. Who said this game wasn't educational?

**Scimitar** Another bit of exotica on the face of it, but actually one of the favourite weapons of the Orcish tribe, so often met in adventures.

**Staff** A piece of wood!

**Silver Staff** A piece of silver, which has magical connotations

**Stake** You can probably work out what this is for (if not just wait for a vampire!)

**Silver Sword** Like the silver staff, this is a bit more magical than its more mundane cousin!

**Silver Bullet** Useful if you have a gun!

**Fire Whip** This is just one example of a weapon that can be used against an elemental. These are creatures formed from the very elements (Earth, Fire, Air and Water) — and as such, they can only be fought with appropriate weapons.

Meanwhile back at the treasure . . .

**Book** Generally just worth points, as an artefact, but they are understood to be repositories of ancient lore, so, on opening, may increase your intelligence — but beware, they have been known to explode on opening!

**Gold Coins** The standard currency in all the best adventures, but not as valuable as

**Gold** which is usually found in 'hoards', and is often the objective of the adventure

**Crystal Orb** Representative of many magical items to be found in adventuring, the orb allows the gazer to see into the future, or another location, thus seeing his death

**Helm of Immortality** This one is v-e-r-y potent, and should not need explaining — actually, if you find this, you have probably won the game (not to mention the universe)

That concludes our very short list of the treasures you may expect to meet in the typical adventure — by no means exhaustive, it nevertheless should give a rough idea of the sort of object that will crop up time and time again.

The better authors, of course, employ a good deal of imagination in creating treasure, weapons and all the other paraphernalia in their adventures.

# CHAPTER 8
# Choosing an Adventure

Let's have a quick resume of what goes to make a good program.

## Graphic or Text?

The answer to this question will determine the final shape of your own adventure. A purely text game needs to have a fairly complicated scenario, one which will hold the attention of the player for the duration of the game. It will, by its very nature, probably contain many puzzles — adventure fans delight in solving complex problems, and the longer they take in the solving, the better. So you can make the puzzles as hard as the situation demands — but never illogical! Your amusing little oddity will have someone else throwing the game away in frustration.

Of course, for a children's adventure, you may decide to pose rather simpler problems — but never talk down to children in your programs.

You may also find numerical, or decoding problems worth trying. Or how about anagrams? These have all been written into programs, with varying degrees of success.

Remember, however, to always pitch the level of conundra at the final user — neither too hard, nor too easy.

A text adventure, as we've seen with the *Wumpus* lookalikes, need not always contain the sort of puzzles we've been discussing. Battles with monsters carried on in text form very soon become boring, though, so some form of problem-posing is almost essential in the long run, if only to stave off the tedium.

Monster-bashing gets very much better when supported by graphics. You will find that given a graphic scenario, the program will often move naturally toward a Dungeons and Dragons form of combat, as this makes for good, heroic confrontations!

There is a danger, here, of getting too far from the original concept of adventures. The board game went through several years of progression from the original Snakes-and-Ladder sort of game, found its fortune in *Monopoly*, and then foundered as writers tried, in vain, to find the next success. Some pretty strange games were seen, as the boardgame mutated into things like *Chart-topper, Libido*, and the rest.

Ironically, the boardgame as we knew it from years ago, is undergoing a renaissance, this time implemented on the home computer —

*Monopoly*, against the wishes of Parker Bros, the original copyright holders, has seen two or three versions for the Sinclair machine, and Waddington's *Cluedo* has also been adapted for the machine, along with several other old favourites. New games are also being written, which owe an obvious debt to those board games of a couple of decades ago.

Comedy is very difficult to pull off with an adventure game; the author must make sure that his humour is appreciated by the player on the twentieth and thirtieth playing as much as the first. APX have released an adventure called *Galahad and the Holy Grail*, which is loosely based on *Monty Python and the . . .* — and although it is a tough, real-time adventure, with many traps, the jokes wear a bit thin after a while. The author has included a Holy Hand Grenade, of all things, as well as monsters like a Great Rabbit.

Nevertheless, it takes a large amount of *brio* for an author to successfully pull off programs of this type. It is true to say the best programs are those that can be returned to again and again, and that is not usually the case with comic adventures. The joke that seems funny once or twice, very quickly goes flat after repeated airings, especially in BASIC!

Heavy-handed humour like this does not always come off well; I prefer the more subtle wit of the Infocom games like *Zork* and *Suspended.*

As we've seen from our quick look at available software for the machine, graphics can take one of several forms. As a mere support for the text, static line pictures can greatly enhance most programs. Having said this, it is important that a facility be included for those graphics to be switched off when required. Thus, the player may draw on his own imagination, or rely on the artist's impression.

Graphics eat up the memory, though, so they're usually only met in programs for the 48K machine. Pictures using block graphics, together with a small number of defined graphics, are frequently used in adventures for the smaller memory, and they can be successful — however, owners of the 48K memory must, by now, outnumber those with the smaller memory. So if you intend to market your program commercially, you will not lose too much in catering for the larger memory.

Adventurers, after all, are probably looking for more and more complex situations, and adverts worded 'fills the whole 48K' just make them drool!

If graphics are to become more than just something to look at while the player gets his brain in gear, or to evoke a mood, then the scenario you write around your adventure need not be so complex. Include puzzles by all means, but they will, of course be more visual than those you write into a text program.

Whatever you choose for your adventure, text or graphics, never forget that the game is all-important. It is no good having stunning graphics, with a boring scenario — the game will be run a couple of times, then consigned to a deep, dark drawer marked YAWN.

How to make your program as addictive? Again, it depends upon your initial answer to that question, graphics or text?

Like a good story or film, text adventures must have a beginning, a middle, and inevitably an end. Level 9's *Dungeon Adventure*, which we looked at in Chapter 2, has a beginning (roaming around outside the cavern complex, collecting various necessities for the adventure ahead, and which we were involved in), a middle (the exploration of the caves themselves), and an end (bringing all the treasure back to base, and thus gaining a final score).

You must, when writing your adventure, allow your imagination free rein! Players are becoming ever more sophisticated in their choice of ideal game. If you memory is large enough, try to include more than one area to be explored, and give the player tantalising glimpses of these additional locations. Thus, when the adventure is finally solved, the player can, in future, return to a previously unsolved location, and possibly have a whole new game.

And don't forget *Zork* and other programs, which give a hint of sequels yet to come, and glimpses of past programs.

# PART 2

# CHAPTER 9
# The Legend

Now at last we're getting down to some serious programming!

The following section of the book will describe a graphic adventure, and its development. Many of the modules described can be taken and used in your own programs, and the techniques discussed, adapted as you wish.

Before starting, let's consider the question — why did we choose a graphic program? After all, the original classic adventure is text-based, as we've seen.

There are two reasons — the more important, probably, is that in this way we can present the maximum number of techniques. As I've said, you may wish to adapt these to your own programs (or, indeed, add your own touches to the present program). The second reason is simple — a text adventure, while being a lot of fun to play, is not much of a surprise after being typed in from a listing!

Having said that, though, you may wish to add to your own text adventure, and the techniques used in *The Eye of the Star Warrior*, may be used just as easily in this situation. For instance, the section on generating the room complex will be just as valid in a text game, as will the movement routines, the monster generating routines, the treasure pick-up and so on.

So let's get on with it. First of all, a bit of background, explaining what you're doing in this hellish dungeon, and what your objectives are in playing the game. Have fun!

## The Legend

There is a dimly remembered legend of the third continent that tells of a treasure with such power that it threatened to consume all life on earth. The Eye of the Star Warrior contained at its heart the fire of a thousand suns!

The legend dates back to early in man's civilization, before the Great Flood. The Demon, Agor, escaped from the negative zone and took a mortal form on Earth. Disguised as the Wizard Domire, Agor held the

lands of the third continent in tyrannical rule for over two million years. No power on Earth could match Agor, and all who opposed him, perished. The wisest of the wizards of the third continent met in secret council, and the Brotherhood of the Star Warrior was formed. They plotted to use mystic arts forbidden by the Creator. The Brotherhood sought to breathe life into a clay giant, and to give this artificial man power greater than the Demon Agor. Only in the depth of space did such power exist! Combining their psychic strength, the Brotherhood set a thousand suns on a collision course that would last two million years.

Future generations protected the giant's body and nurtured the growing gem that would eventually sit in the Star Warrior's single eye socket. As the time grew close, many members of the Great Council questioned the plan to give the Star Warrior so much power. Aldous, last leader of the Council, gave orders for the eye socket of the Star Warrior to be lined with explosives.

When the meeting of the suns took place, the energy of their collision was channelled into the magic stone. The stone was then ceremoniously placed in the eye socket of the Star Warrior, and it gave life to the clay giant.

The Star Warrior was terrible to behold! It strode across the countryside, burning the land and destroying everything in its path. Soon, the Star Warrior met Agor in battle, and the Demon was incinerated. The Star Warrior placed himself on Agor's throne, and a new reign of terror spread across the third continent. Under orders from the Star Warrior, members of the Brotherhood were hunted down and slain. Aldous was captured and brought to the Temple, there to be dismembered by Trolls, under the terrible gaze of the Star Warrior.

As Aldous' right hand was severed from his arm, it rose from the floor, and fired a lightning bolt that struck the Star Warrior in the eye. The explosives were detonated and the eye fell from its socket.

Without the power of the stone, the giant was no more than a statue, and it crashed to the ground. One of the Trolls uttered a curse and brought down his sword towards Aldous' chest. Before he died, Aldous threw a magic field of energy around the stone, to prevent it being replaced in the Star Warrior.

The stone has since fallen into the hands of evil Wizards. It is now possessed by a Lich, a dead Wizard with almost demonic power. He is close to breaking the shield around the stone.

The Eye of the Star Warrior must be destroyed before the powers of evil can use it once more!

# CHAPTER 10
# Create your own Dungeon

Our first task, then, is to create the dungeon complex.

The complex will be spread over three levels, with a maximum of 300 cells, 100 on each level. It is quite possible to have one large level only, consisting of the same 300 cells. It would get rather difficult to find your way about after a while, though, and in having different levels, we can more easily set different tasks, and varying degrees of difficulty.

Now, although we are eventually going to be moving about a system of rooms, and will then be calling them rooms, at this point in the program's development, we will be talking also of cells.

In a dungeon complex, we will naturally require a number of individual rooms, through which we can move, and which will contain monsters, and (we hope) treasures. These rooms will be inter-connected (unless you want your player to tunnel between rooms).

```
30 GRAPHICS 2:POKE 710,0:POKE 708,15
40 POSITION 7,0:? #6;"PLEASE":POSITION 8
,2:? #6;"WAIT":POSITION 7,4:? #6;"2 MINS
"
50 REM TURN ON ATTRACT MODE
60 POKE 77,128
70 GOSUB 610
```

The first few lines set up the Attract mode; that is, as the dungeon is being set up, the screen will cycle through some restful colours to let you know that while the message PLEASE WAIT 2 MINS is being displayed, there is actually something happening! If you don't happen to like this, you can substitute POKE 77,0, which the program does after the set-up (at line 600, second statement).

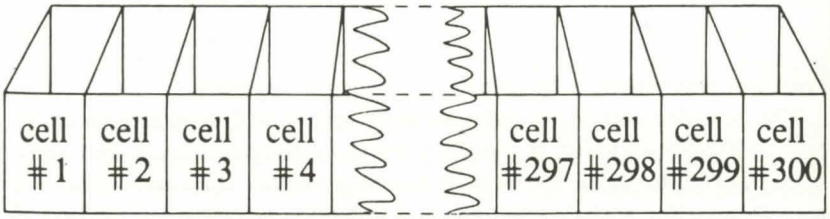
After this, line 70 sends the program to line 610, which starts our set-up and initialisation routine.

```
610 DIM C(300)
```

LEAVE THE COMPUTER SWITCHED ON, OR SAVE THIS LINE

Line 610 simply DIMensions the array C to 300. C now contains 300 locations, and can be thought of as a long length of boxes, if you like, stuck together. There are 300 of them (see **Figure 1**). This takes care of our maximum number of cells.

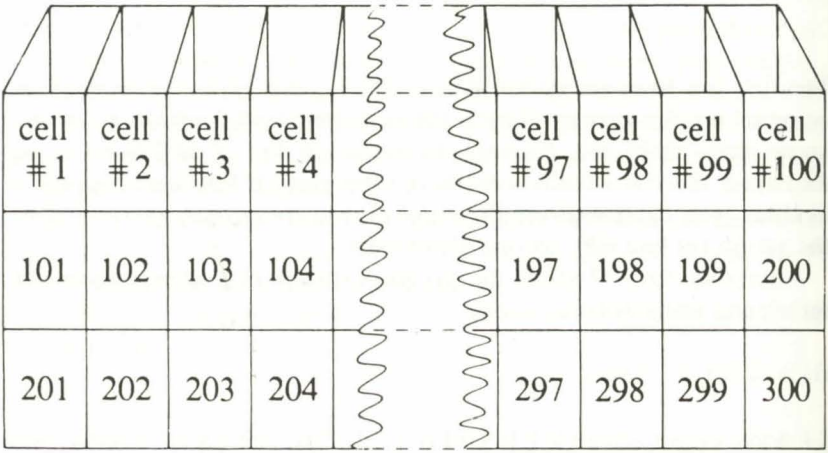
**Figure 1**




Now, we could leave the set-up there — we would be able to move from cell #1, to cell #2, and then onto #3 and #4 and so on, or indeed back to cell #1 if we wished — but what a boring game. In a text adventure, we would be moving in one direction all the time — no variety! And if a terrible monster was barring our way, we would be able to progress no further, being able to find no way round him. So, we have to provide a bit of flexibility.

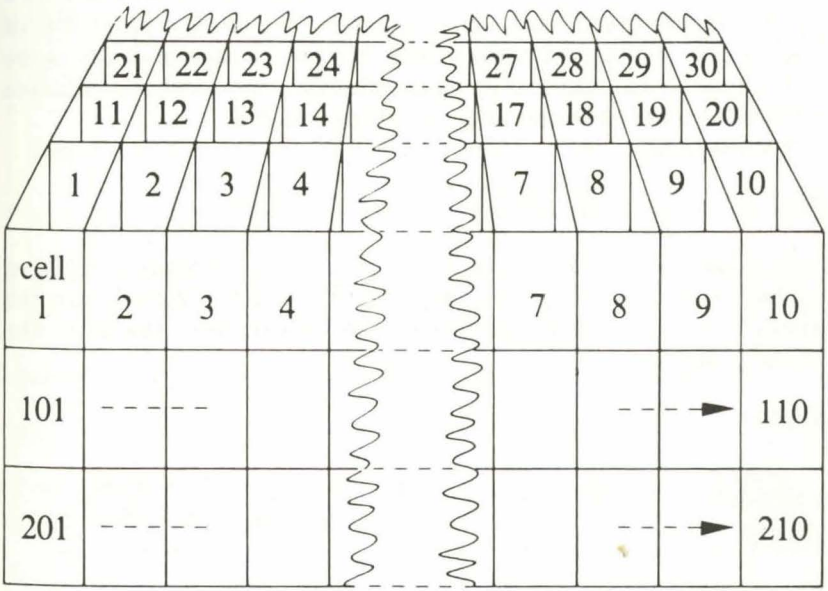
To do this, let's first imagine (and this will be entirely in our imagination, as the computer is still thinking of the array C as one long line of boxes) that we can somehow bend our 300 boxes, or cells, into three layers of 100 cells each. If we chose to stay with this format, we would at least be allowed some freedom of movement up or down from any given cell, and also left or right from a cell to an adjacent cell.

**Figure 2**



But, although this is a bit better than our first set-up, you will have noticed that we still only have two-dimensional movement. We could make our matrix, which at the moment is 3×100, into a matrix of, say, 30×10, or some such construction, which would make for a more complex set-up, but still only two-dimensional. Let's go three-dimensional. Staying with our three levels, we'll construct a box, with the dimensions 10×10×3. Our complex now looks like this!

**Figure 3**




So now we have three levels, each of 100 cells (10×10), in three dimensions, about which we may roam to our heart's content. This is the theory — but how does the computer hold this information. In fact, computers are not able to grasp concepts like three dimensions, as are we humans. Array C is still, to the computer, a long line of 300 boxes, or locations, and will remain so, however we may choose to imagine it.

How, then, does our computer know when we are on level 1, or 2, or 3?

Let's look again at **Figure 3**. You can see from this that the player, in real life, could simply go from Cell #1, directly down to Cell #101, and thus be on the second level. But the computer is still, as we said, holding the whole dungeon complex in one long array of 300 boxes. In fact, **Figures 2** and **3** exist only in our imagination. To see the reality of the computer's view, turn again to **Figure 1**. It would need some pretty

fancy coding to achieve easy movement between our three levels. The way we get our computer to distinguish between the three levels, is by putting an impassable barrier, or wall, between a cell of one level, and a cell of another level. In other words, design and create a wall.

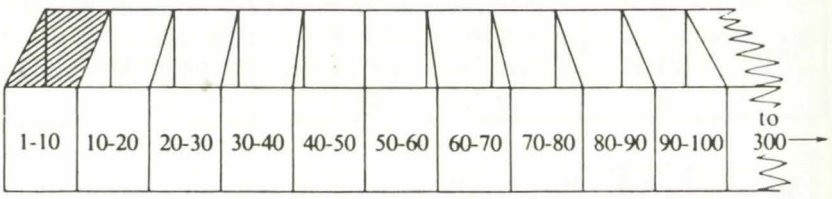
Now we can return to the computer! We have 300 cells, or locations, in array C, and we know that we can move freely around all these cells (if we had keyed in a movement module — we will later). We have now to make some kind of barrier between the levels, and the best way we can do this is to place a number in the required location in array C, which will act as a marker. As we have 300 locations, we need to have a number higher than 300, to act as a marker. We could make it 301, or 462, or 872 — it doesn't really matter, but for the sake of clarity in the listing (not to mention that in six months' time we may have forgotten what these numbers stand for!), let's settle on 999.

Let's type in:

```
630 FOR A=1 TO 10:C(A)=999:NEXT A
```

This line will create a block in cells 1 to 10, ie the array C will now contain, in its first 10 locations, the value 999, which effectively bars the player from visiting these particular cells. Let's see what this looks like to the computer.

**Figure 4**



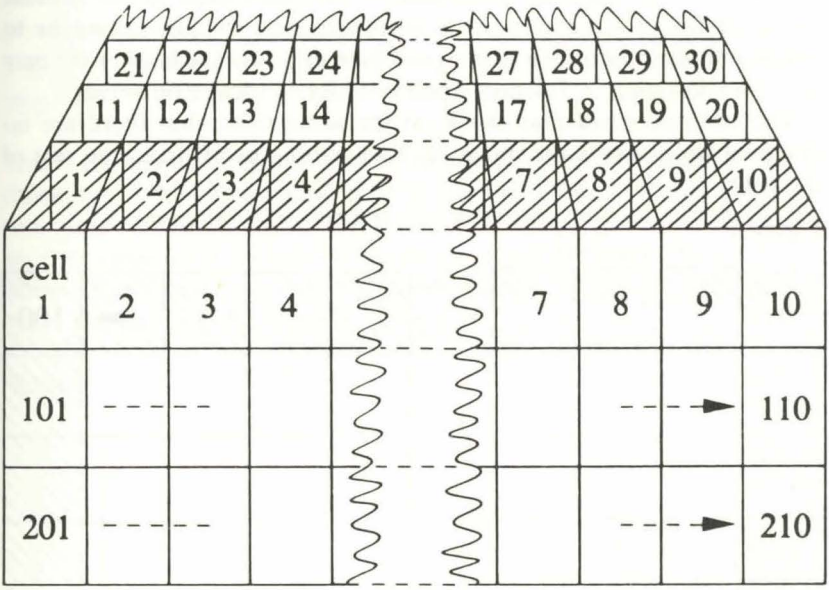

Why cells 1 to 10? Look now at our imaginary three-storey dungeon. Aha! An impenetrable wall to the outside. Cells 1 to 10 are now no-go areas, and will never contain monsters or treasure. The next three lines do the same for the other levels.

```
640 FOR A=90 TO 110:C(A)=999:NEXT A
```

Line 640 creates an impassable barrier at the last row of Level 1, and the first of the second level, while

```
650 FOR A=190 TO 210:C(A)=999:NEXT A
```

**Figure 5**




creates the wall between the second and the third levels. Finally

```
660 FOR A=290 TO 300:C(A)=999:NEXT A
```

creates the last barrier at the end of the third level. Now, we need to put in the boundaries at the edges of the levels

```
670 FOR A=11 TO 291 STEP 10:C(A)=999:NEX
T A
680 FOR A=10 TO 300 STEP 10:C(A)=999:NEX
T A
```

These two lines create blocks in steps of ten, thus creating the remaining boundaries.

Look at the next illustration, a plan view of Level 1 to see what has happened. The other two levels have been treated in the same way.

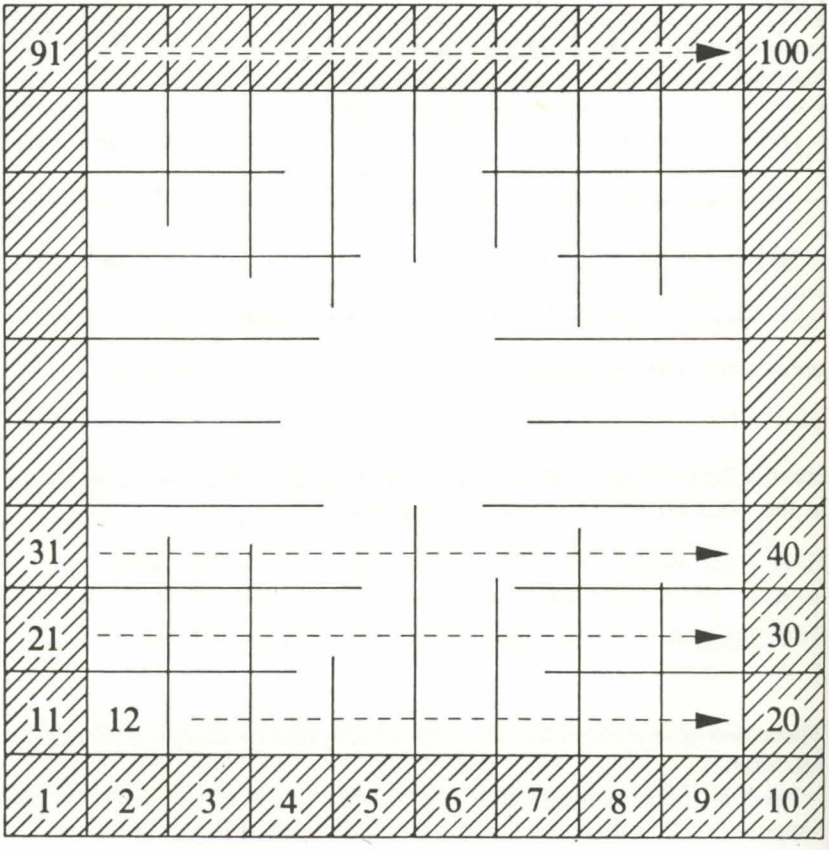
To complete the complex, we have to scatter a few walls at random throughout our dungeon, to make for a bit of variety in the game.

As the computer looks through array C, or, as we imagine it, our three levels of 100 cells each, on a 10×10 matrix, it is going to check the cells around each location, to see if any contain the value 999. As we know, this value is a marker which tells us that this particular cell, or location, is a no-go area. It may not seem important, at the moment, if

we have two of these walls together, but let a cell be surrounded by walls and we have an impenetrable room — not much good in our present game (although you may want to write a routine to allow us maybe to tempt a monster into such a room and then imprison the beast.) We may have, for instance, a Teleport Spell in order to escape ourselves.

The computer will also check, at the same time, that there are no diagonal walls across the dungeon, thus cutting us off from the rest of the cells.

**Figure 6**




```
690 B=INT(RND(0)*30)+40
```

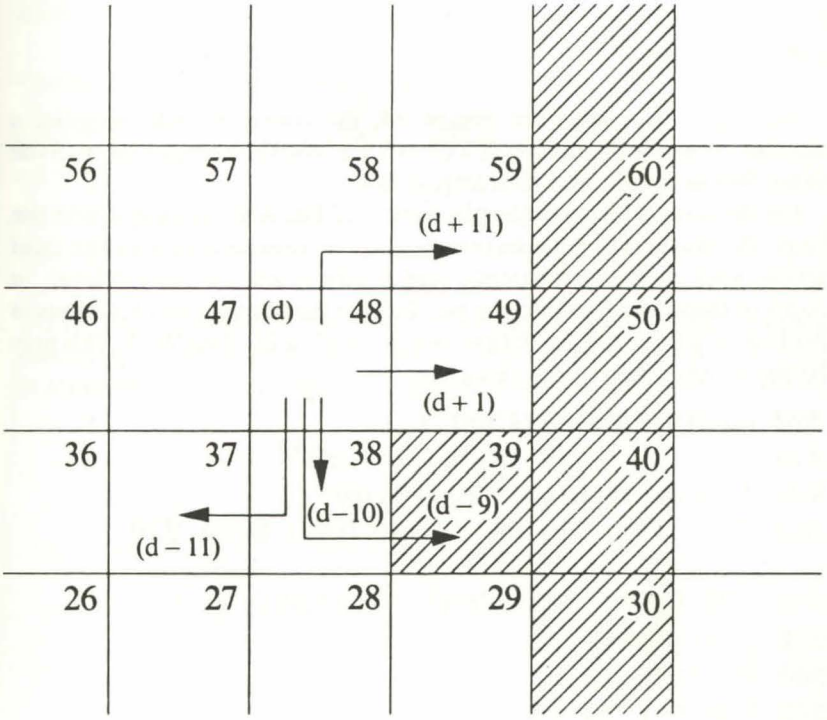
gives us a value for B, between 30 and 59. This is the range we require, of walls in each level, apart from the boundary walls.

```
700 FOR A=1 TO B
710 D=INT(RND(0)*280)+11
720 IF C(D)=999 THEN GOTO 710
```

```
730 IF C(D+11)=999 THEN GOTO 710
740 IF C(D-11)=999 OR C(D-9)=999 OR C(D+
9)=999 THEN GOTO 760
750 C(D)=999
760 NEXT A
```

This is the loop to create the walls. In lines 730 and 740, the program is checking all the surrounding cells to the one chosen by 710, to make sure none contain the value 999. If one does, the program returns to find another value for d, and repeat the check. Look at **Figure 7** to see this in action. The illustration shows just one section of Level 1 (although it could just as easily be Level 2, or 3).

**Figure 7**




Line 710 gives us a value for d of between 10 and 290 — these numbers should be familiar to us now, but to recap, we have walls between 1 and 10, and between 290 and 300, so there's no point trying to put a wall at these locations! In our example, the program is checking location 48 — that is, d has received the value of 48 from line 710.

You can see these lines at work, in **Figure 7**. The surrounding locations of each location are checked, in turn, and the illustration shows location #48 (the current value of d) being checked. A value of less than 999 is encountered until location 39 is looked at. Does d−9 (48−9=39) =999? Zounds! it does, and now the program jumps to line 760, which returns us to line 710 for a new value. If, on the other hand, no 'wall' had been found surrounding C(48), that location would have been given the value 999, thus making it an impassable wall.

Now that all our cells are designated either walls, or rooms, we can go on to number those rooms.

```
770 D=0
780 FOR A=10 TO 290
790 IF C(A)=999 THEN GOTO 820
800 D=D+1
810 C(A)=D
820 NEXT A
```

Now that our rooms are numbered, the computer will recognise a number between 10 and 290 as a clear area, and those locations with the value 999 as a wall, and thus impassable.

On the second level we can place the Fire Pit. After his battle with the Lich, the player will have to renegotiate the various rooms of the third levels, with its attendant perils, and finally reach the second level, in order to throw the Eye into the pit. To give the player a bit of a bonus in the earlier part of the adventure, we can build a Healing Well. This may be put in any room in the dungeon.

```
900 HW=INT(RND(0)*300)+1
910 IF C(HW)=999 THEN GOTO 900
920 FP=INT(RND(0)*200)+100
930 IF C(FP)=999 OR FP=HW THEN GOTO 920

940 DIM TREA$(375),OM$(15),Y(26)
950 J=1:TREA$=""
960 RESTORE 1010
970 FOR N=1 TO 26
980 READ OM$
990 TREA$(J)=OM$:Y(N)=LEN(TREA$):J=1+Y(N
)
```

One last thing needs to be done before we can leave this phase of the program. How do we travel between the floors of the dungeon? Having

gone to a lot of trouble to set up walls between the levels, we need a way of bypassing these walls as we find necessary.

Stairs have been a successful answer to this problem for several centuries, so let's put some of these in our complex.

We don't want to make it too easy for adventurers to find their way to another level, so let's put just one staircase on each level. For Level 1, we'll store the location of the staircase in variable SONE, and of course, we don't want our staircase in a wall, so we can forget locations C1–10 and C90–100, and line 840 selects a random location in which to place the stairway. Line 850 checks to make sure we are not placing it in a wall.

```
830 REM SET UP STAIRS,HEALING WELL,FIRE
PIT
840 SONE=INT(RND(0)*90)+10
850 IF C(SONE)=999 THEN 840
```

And so on for the other two levels:

```
860 STWO=INT(RND(0)*90)+110
870 IF C(STWO)=999 THEN 860
880 STHREE=INT(RND(0)*90)+210
890 IF C(STHREE)=999 THEN 880
```

This finally completes the structural set-up of our dungeon complex. And although the routines were written with our final graphic adventure in mind, the techniques may just as well be used in your text adventure.

# CHAPTER 11
# Monster, Monster

No self-respecting dungeon should be without its fair share of monsters and treasures, so let's get on and put some in ours.

```
1000 NEXT N
1010 DATA DUMMY
1020 DATA SPADE,FIRE WHIP,SWORD,SILVER S
WORD,SILVER STAFF,SAINTLY STAFF,TALISMAN
,CROSS,SHIELD,TORCH,INVISIBLE CLOAK
1030 DATA CLUB,HOLY WATER,BOW AND ARROWS
,MAGIC SHIELD,EMPTY BOTTLE,HEALING WATER
,WIZARDS HAND,TELEPORT,FORCESHIELD
1040 DATA PSYCHIC SHIELD,LIGHTNING BOLT,
STONE SPELL,LIMBO SPELL,STONE
```

After the structural set-up, the program goes on to Line 940 and dimensions TREA$ to 375, which is the number of characters in the data statements in lines 1020, 1030 and 1040. OM$ is a dummy string that the program uses, along with the dummy variable Y, to set up TREA$. OM$ is dimensioned to 15 because we have 15 weapons to select from, and Y is dimensioned to 26 as that is the maximum number of characters in the names of the weapons.

The DUMMY in line 1010 looks a bit inelegant, but, if it were not for this, we would need a much more complex algorithm in, for instance, line 2270, which will print out the weapons we hold in the menu routine.

Now that we have a string of weapons (TREA$), how about putting a bit of uncertainty into our combat sequence? After all, we can't expect a spade to be of much use against a vampire!

ATT$ will hold information on the attack capabilities of each weapon, while DEF$ will do the same for its defence properties.

```
1050 DIM ATT$(320),PM$(20),Z(17)
1060 J=1:ATT$=""
```

```
1070 RESTORE 1120
1080 FOR N=1 TO 17
1090 READ PM$
1100 ATT$(J)=PM$:Z(N)=LEN(ATT$):J=1+Z(N)
1110 NEXT N
1120 DATA DUMMY
1130 DATA 0103040506071222 24,02071022232
4,06071324,0207102324,06071324,020405061
013222324
1140 DATA 060724,020405061013222324,0304
222324,0204060710222324,030414222324,010
506071213222324,02030414222324
1150 DATA 02030410222324,03040714222324,
2324
```

The routine is much the same as that for putting the weapon names into TREA$. And this is the routine for the defence string DEF$.

```
1160 DIM DEF$(342),QM$(18),A(20)
1170 J=1:DEF$=""
1180 RESTORE 1230
1190 FOR N=1 TO 20
1200 READ QM$
1210 DEF$(J)=QM$:A(N)=LEN(DEF$):J=1+A(N)
1220 NEXT N
1230 DATA DUMMY
1240 DATA 09111519,020910111519,07081519
20,02070910111519,071319,020708091011131
519
1250 DATA 0207080910111314 19,0708151920,
09111519,020709101113151 9,020910111519
1260 DATA 091113151920,020910111519,0209
10111519,07091519,192021,1520,21,21
1270 DATA 091113151920,020910111519,0209
10111519,07091519,192021,1520,21,21
```

Before we return to line 70 (if you remember, we were sent here by that line), we can set up a place in memory for our custom character set. This is done in lines 1290–1320. As you see, line 1320 sends the program onto line 3060.

```
1290 CH=(PEEK(106)-8)*256:CHORG=57344
1300 FOR I=0 TO 1024:POKE CH+I,PEEK(CHOR
G+I):NEXT I
1310 LOC=CH/256
1320 GOSUB 3060
1330 RETURN
```

Line 3690 dimensions C$ to 13, the number of characters we are going to redefine. You'll see the characters in line 3700. The next lines POKE the data from lines 3810–4010 into the space we reserved in lines 1290–3010.

```
3690 DIM C$(13)
3700 C$="MXQZ!&#$'@()<"
3710 RESTORE 3810
3720 FOR T=1 TO LEN(C$)
3730 CHPOS=CH+(ASC(C$(T))-32)*8
3740 FOR I=0 TO 7
3750 READ QQ
3760 POKE CHPOS+I,QQ
3770 NEXT I
3780 NEXT T
```

Now, finally, we return to line 80.

```
80 DIM ROOM$(200),HOLD(5),DROP(300)
90 ST=50
100 FOR T=1 TO 5:HOLD(T)=0:NEXT T
```

Line 80 dimensions ROOM$ to 200, and is going to be used to print the graphics of the dungeon, room by room, as our player moves above. HOLD is the array holding the names of the weapons we carry (to a maximum of five), and DROP is the array holding information about the weapons we might want to drop. This is set to 300, equalling the number of possible rooms. Line 90 is dealt with in Chapter 13.

To set up the graphics, we first have to clear out the string ROOM$, and then we fill it in the right places with our redefined 'Q'. In the final screen display, each room will be a rectangle 20×8, and that's the reason for all those Qs! They will, of course, be something else when the program is run!

```
120 ROOM$(1)="♡":ROOM$(200)="♡":ROOM$(2)
=ROOM$
130 ROOM$(1,20)="QQQQQQQQQQQQQQQQQQQQ":R
EM "Q"S IN INVERSE!
140 FOR I=1 TO 8
150 ROOM$(20*I+1,20*I+1)="Q":ROOM$(20*I+
20,20*I+20)="Q":REM INVERSE "Q"S!
160 NEXT I
170 ROOM$(181,200)=ROOM$(1,20)
```

Please note that the Q character in quotes MUST be entered in the INVERSE mode!

## Name that monster

On without pause to the good bit — populating our new dungeon with the enemy and the loot!

We want MONS and TRES, the two arrays which will hold information about the monsters and treasure respectively, to coincide with the array C, which, as you will remember, is our cell array. Knowing that the last ten cells, or rooms, of C will have the value 999 (in other words, be solid walls), we could in fact dimension the two new arrays to 290 — but let's waste a bit of memory! There will be a maximum of one treasure (which will take the shape of a chest containing lots of lovely gold), and one monster to each room — monsters will not be able to roam around in this dungeon (because of the forcefield at each door).

```
180 DIM MONS(300),TRES(300)
190 FOR R=1 TO 300
200 M=INT(RND(0)*10)
210 IF C(R)=999 THEN 270
220 IF M<INT(R/100+1)*2 THEN MONS(R)=1:G
OTO 240
230 MONS(R)=0
240 T=INT(RND(0)*10)
250 IF T>5 THEN TRES(R)=INT((RND(0)*(R+5
0))/5)*5:GOTO 270
260 TRES(R)=0
270 NEXT R
```

Line 190 contains a dummy, or local variable, which you will meet quite often throughout the program. This one is R, and loops the little

routine through the 300 numbers of our locations. As with all the other local variables, this one will be forgotten when its job here is over. M is another local variable, and on line 200 receives a random value. Line 220 takes that value and does a little sum with it — if M is less than the answer, a 1 is put into MONS at the location R. This means that there is now a monster here. A '0' means that no monster is present at this location in the array. Because of the algorithm at 220, you can see that the chances of a monster being put into higher-numbered locations is greater than for the lower-numbered locations. Lines 240–260 do the same thing for the treasure. The weapons get a similar treatment, although here we must first of all clear the array WEAPON, as we shall want to drop weapons as we move around in the final game. Lines 280–390 achieve all this, with the array being first dimensioned to 300 and cleared (the value 0 being put in each of the 300 locations) on lines 290–310. The local variables here are T, which is the level, or floor, and I, which selects one of the 25 possible weapons.

```
280 DIM WEAPON(300)
290 FOR Q=1 TO 300
300 WEAPON(Q)=0
310 NEXT Q
320 FOR T=0 TO 2
330 FOR I=1 TO 25
340 PLACE=INT(RND(0)*100)+(T)*100+1
350 IF C(PLACE)=999 THEN GOTO 340
360 IF WEAPON(PLACE)<>0 THEN 340
370 WEAPON(PLACE)=I
380 NEXT I
390 NEXT T
```

Although we have now reserved space throughout the dungeon for the monsters, we have not yet named them. To hold the names we create another string, and call it MONSTER$. This is dimensioned, in line 420, to 160, that being the number of characters in the data statements of lines 400 and 410, and the data from those lines read into the string.

Line 460 creates a variable CHAR, which is dimensioned to the length of DUM$. This in turn takes its length, to a maximum of 10, from the data. Thus a flag is set up in MONSTER$ which signifies the end of that particular monster. The program will need to know this when printing the name of the monster.

```
400 DATA skeleton,mummy,demon,zombie,ele
mental,vampire,banshee,wraith,dragon,wer
ewolf,cyclops,sandman,harpie
410 DATA serpent,balrog,lich
420 DIM MONSTER$(160),DUM$(10)
430 RESTORE 400
440 FOR MONSTER=1 TO 16
450 READ DUM$
460 CHAR=LEN(DUM$)
470 MONSTER$((MONSTER-1)*10+1)=CHR$(CHAR
)
480 MONSTER$((MONSTER-1)*10+2)=DUM$
490 NEXT MONSTER
```

And to put our named monsters into those reserved spaces in the dungeon, we type in the next lines.

```
500 FOR T=1 TO 100
510 IF MONS(T)=1 THEN MONS(T)=INT(RND(0)
*10)+1
520 NEXT T
530 FOR T=101 TO 300
540 IF MONS(T)=1 THEN MONS(T)=INT(RND(0)
*15)+1
550 NEXT T
560 REM PLACE LICH IN FINAL LEVEL
570 PLACE=INT(RND(0)*100)+201
580 IF C(PLACE)<>999 THEN MONS(PLACE)=16
:GOTO 600
590 GOTO 570
```

Line 510 checks to see if the array MONS, at location T (only the first 100 locations at this point), contains a 1, which, you'll remember from the routine at line 220, means that there is going to be a monster here. If this is the case, then a random number from 1 to 10 is placed here. Line 530 does the same, but for the higher levels, and you'll see from line 540 that a random number from 1 to 15 is put in locations on the higher two levels. Line 570 places the Lich, the ultimate monster, in some location on Level Three. In fact, in the Lich's case, it doesn't matter if MONS contains a 1 at the chosen location, or even if there is already a monster there — as befits the Lich's status, he will just replace it.

# CHAPTER 12
# Attack and Defend

We are nearly able to print out our graphics, and the two minutes set-up time is just about over now. The rooms, or locations have been initialised, and the program has tabs on what monsters are where, and what weapons are where. It also knows which weapons can be used in attacking each monster, and which to defend against each monster. And we have set aside an area for our redefined graphics, and redefined certain characters.

A couple more things have to be done at this point. To give the player some sort of prod, we will give him an initial strength of 50 points, and allow this to be decremented each time he fights a monster. The routine to effect this will come later, but, for now, we can initialise it in line 90, with ST becoming the array.

We are about to set foot in the first room. As the first eleven locations are solid walls (ie the array C has a value of 999 in the first 11 locations), we can start off in Room 12, Level 1.

```
620 LEV=1:ROOM=12
```

Then we place our player in the middle of the room. POKEing 77 with 0 will turn off the attract mode that we initiated in line 60, and the program is sent to 4330.

```
600 X=1:Y=4:POKE 77,0:GOTO 4330
```

Line 4340 is the start of the movement and combat routine but, first of all, the program checks to see if there is a monster here. This is done in the two lines 3090 and 3100. If not, the two variables HI and DE are set to zero. HI is a flag which will eventually show whether our player's weapons have an attack capability for the monster present at the location. DE is a similar flag to show defence capabilities. After finding no monster, the program returns to our movement routine at 4350, which we shall cover in the next chapter.

```
4340 GOSUB 3090

3090 QQ=MONS(ROOM)
3100 IF QQ=0 THEN HI=0:DE=0:RETURN
```

If, however, there *is* a monster at the present location, then the program goes on to work out how the coming fight will be carried out.

```
3110 PM$=ATT$(Z(QQ)+1,Z(QQ+1))
3120 FOR Q=1 TO LEN(PM$) STEP 2
3130 IF ATT=VAL(PM$(Q,Q+1)) THEN HI=1:GO
TO 3160
3140 NEXT Q
3150 HI=0
```

These lines find what weapons the player is carrying, and compare the value of the weapons with the value of ATT$. This string has been dimensioned in the routine starting at line 1050. When the program finds a weapon that will be a good defence against the monster here (the name and number of this monster has been found in line 3090), then the flag HI is set to 1, and the program goes on to work on the weapons to find a defence capability (lines 3160–3200). If no such attack weapon is found, then the flag HI is set to 0.

```
3160 PM$=DEF$(A(QQ)+1,A(QQ+1))
3170 FOR Q=1 TO LEN(PM$) STEP 2
3180 IF DEF=VAL(PM$(Q,Q+1)) THEN DE=1:GO
TO 3210
3190 NEXT Q
3200 DE=0
```

These lines again run through the list of weapons held, and look to see if any of these will be a good defence against the monster, with the flag DE being set accordingly.

```
3210 DE=DE*(INT(RND(0)*22)+78/LEV)
3220 HI=HI*(INT(RND(0)*22)+78/LEV)
3230 RETURN
```

And now a certain percentage is worked out — initially, this percentage could be up to 99% (the monster must be allowed that final edge), although the algorithm will make sure that, at higher levels, the percentage will be less. Of course, if the two flags DE and HI are 0, then the result of the equation will be 0 too.

# CHAPTER 13
# Pretty Pictures

Now that the initial parameters have been sorted out, we can start drawing some pictures. If your taste is for text adventures, it would be possible for you to conduct the whole game in text, using what we have set up so far. But without graphics the game would be rather boring, like those programs that are still released, even now, based on the old *Wumpus* game that we touched on earlier.

In this sort of game, the player is merely told at each location: THERE IS A DRAGON/ORC/GIANT MOLE/KILLER HANDBAG HERE. WHAT NOW? Not very dramatic is it? Wouldn't it be better if we could actually see that monster coming towards us, jaws slavering, eager for our blood?

```
4350 GOSUB 3240:POKE 756,LOC:POSITION X,
Y:? #6;"m":REM INVERSE "m"!
```

The first statement is the one we're interested in at the moment — we will come back to the following statements on our return.

```
3240 IF WEAPON(ROOM)>0.9 THEN GOSUB 4030
3250 IF DROP(ROOM)>0.9 THEN GOSUB 4100
3260 IF MONS(ROOM)>0.9 THEN GOSUB 4170
```

These three lines send us on to the subroutines to draw all the characters that will change depending on what the program finds at the present location.

```
4030 RESTORE 5350+WEAPON(ROOM)*10
4040 CHPOS=CH+(ASC("(")-32)*8
4050 FOR I=0 TO 7
4060 READ QQ
4070 POKE CHPOS+I,QQ
4080 NEXT I
4090 RETURN
```

WEAPON is the variable which holds the weapon information. These lines take data from lines 5360 on, and redefine the character '(' using that data.

```
4100 RESTORE 5350+DROP(ROOM)*10
4110 CHPOS=CH+(ASC(")")-32)*8
4120 FOR I=0 TO 7
4130 READ QQ
4140 POKE CHPOS+I,QQ
4150 NEXT I
4160 RETURN
```

These lines do the same for dropped weapons (that is, weapons that we may have previously left here). In this case, the character ')' is used. Now for the monster graphics. This routine is slightly different, as the monsters are made up from four characters. The little subroutine at line 4280 is the one that places all the data in each of those four characters (!, &, ' and $).

```
4170 RESTORE 5600+MONS(ROOM)*40-30
4180 ? " ":POSITION 5,0:? MONSTER$(MONS(
ROOM)*10-8,MONS(ROOM)*10-9+ASC(MONSTER$(
MONS(ROOM)*10-9)))
4190 CHPOS=CH+(ASC("!")-32)*8
4200 GOSUB 4280
4210 CHPOS=CH+(ASC("&")-32)*8
4220 GOSUB 4280
4230 CHPOS=CH+(ASC("#")-32)*8
4240 GOSUB 4280
4250 CHPOS=CH+(ASC("$")-32)*8
4260 GOSUB 4280
4270 RETURN
4280 FOR I=0 TO 7
4290 READ QQ
4300 POKE CHPOS+I,QQ
4310 NEXT I
4320 RETURN
```

Line 4180 is here because, before entering a new room, the name of the monster present there, if indeed there is one, is printed beneath the room display. Thus the player has the option of changing his mind about entering, or selecting the appropriate weapons to use in combat.

Incidentally, I haven't included all the data statements — you'll find these at the back of the book, in the main listing, at lines 5360–6240.

The following routine will print the room graphics and all the contents. We'll go through the routine in sections.

```
3270 GRAPHICS 2:POSITION 0,0:POKE 756,LO
C:? #6;ROOM$
3280 IF C(ROOM-1)<>999 THEN POSITION 0,4
:? #6;"♡"
3290 IF C(ROOM+1)<>999 THEN POSITION 19,
4:? #6;"♡"
3300 IF C(ROOM+10)<>999 THEN POSITION 10
,9:? #6;"♡"
3310 IF C(ROOM-10)<>999 THEN POSITION 10
,0:? #6;"♡"
3320 IF TRES(ROOM)>0.9 THEN POSITION 10,
5:? #6;"<":REM INVERSE "<"!
3330 IF LEV=1 AND SONE=ROOM THEN POSITIO
N 18,1:? #6;"X":REM INVERSE "X"!
3340 IF LEV=2 AND STWO=ROOM THEN POSITIO
N 18,1:? #6;"X":REM DITO!
3350 IF LEV=3 AND STHREE=ROOM THEN POSIT
ION 18,1:? #6;"X":REM AND AGAIN!
```

Line 3270 prints, in graphic mode 2, ROOM$. The following lines, 3280–3310, look through the array C to see if there is an adjoining room, in which case a blank is inserted on that wall. Line 3320 checks the value of TRES. If the value at this location is greater than 0.9 (ie, 1 or more), then there is a weapon here, and the graphics for that printed. Don't forget to type in the character between the quotes (the '<' arrow), in INVERSE! Lines 3330–3350 check to see if there is a staircase here. If there is, the graphic is printed (and again, the character in quotes *must* be typed in *inverse*).

Every time a room is entered, this routine is run through, and the program will then come to line 3360. This sets a flag, DONE, to 0. This signifies whether the basic room graphics have been printed. On first entering a room, this flag, being at zero, points the program to line 3420, which checks to see if there is a monster here. If not, the program then jumps on to line 3470, which checks for weapons, and then (if there is no weapon), to line 3520 to check for dropped weapons. If no dropped weapons are here, we go on to line 3570, which checks for the Healing

Well, and then on to line 3620, which does a final check, this time for the Fire Pit. If none of these things are present, the flag DONE is set to 1, signifying that the graphic set-up is now complete, and the program returns to line 3360.

```
3360 IF DONE=0 THEN GOSUB 3420
```

This is the line for monster check

```
3420 IF MONS(ROOM)<0.9 THEN GOTO 3470
```

and the one for weapon check

```
3470 IF WEAPON(ROOM)<0.9 THEN GOTO 3520
```

Next the one for dropped weapon check

```
3520 IF DROP(ROOM)<0.9 THEN GOTO 3570
```

then the Healing Well check

```
3570 IF HW<>ROOM THEN GOTO 3620
```

and finally the Fire Pit check

```
3620 IF FP<>ROOM THEN GOTO 3670
```

What if one of these items is found at the location? Take the monster first. If the check at line 3420 finds that there is indeed a monster here, then the program just carries on to the next line, 3430, which gives an initial starting point for MX and MY, the monster co-ordinates. The following lines print the monster graphics at these positions, not forgetting that, in the monster's case, four characters are used and the program has to add one to each of the co-ordinates. Line 3450 sets a flag LI to 1, signifying that the monster is alive. Line 3460 works out the monster's strength, and puts that value into the variable MST. This value will get higher as the level number gets higher.

```
3430 MX=8:MY=3
3440 POSITION MX,MY:? #6;"!&":POSITION M
X,MY+1:? #6;"#$"
3450 LI=1:REM SET FLAG THAT MONSTER IS A
LIVE
3460 MST=INT(RND(0)*MONS(ROOM))+(20+MONS
(ROOM))*LEV:REM ALGORITHM FOR MONSTER ST
RENGTH
```

Now we can do the same for the other objects — the weapons, dropped weapons, Fire Pit and Healing Well. There are a few differences; the starting position is random (line 3480 is typical), and then checks, with the LOCATE statement, to make sure that no other graphic is being overprinted at that position. If it is, the program goes back to the beginning of the subroutine (for instance, line 3510 goes back to 3480) to find another position.

```
3480 WX=INT(RND(0)*17)+2:WY=INT(RND(0)*6
)+2
3490 LOCATE WX,WY,QQ
3500 IF QQ=0 THEN POSITION WX,WY:? #6;"(
":GOTO 3520
3510 GOTO 3480
3530 DX=INT(RND(0)*17)+2:DY=INT(RND(0)*6
)+2
3540 LOCATE DX,DY,QQ
3550 IF QQ=0 THEN POSITION DX,DY:? #6;")
":GOTO 3570
3560 GOTO 3530
3580 HX=INT(RND(0)*17)+2:HY=INT(RND(0)*6
)+2
3590 LOCATE HX,HY,QQ
3600 IF QQ=0 THEN POSITION HX,HY:? #6;"'
":GOTO 3620
3610 GOTO 3580
3630 FX=INT(RND(0)*17)+2:FY=INT(RND(0)*6
)+2
3640 LOCATE FX,FY,QQ
3650 IF QQ=0 THEN POSITION FX,FY:? #6;"@
":GOTO 3670
3660 GOTO 3630
3670 DONE=1:RETURN
```

Finally, line 3670 sets DONE to 1, meaning that all the graphics have now been dealt with. And now the program returns from where it came — the routine at line 4350.

There is one final graphic to be printed — that of the player. This is accomplished by the final statement of line 4350. Note that this must be entered in lower-case inverse!

# CHAPTER 14
# Let's make a Move!

Now all the graphics are printed and we have a nice picture on the screen showing the room, complete with walls, and anything else, like treasure chest, monster, fire pit and so on. And there is our little man, standing there, waiting for something to happen. Now we need to write a movement routine. Line 4370 is the start of the routine. Lines 4400–4430 check the direction in which the player is moving the joystick. DIRX and DIRY are used to update the player's co-ordinates without losing the values of X and Y. If you like, they act as a suicide squad — in other words, DIRX and DIRY range ahead of X and Y. If the positions looked at by DIRX and DIRY are clear, then X and Y are safely updated at line 4480. Line 4490 prints the player graphic at the new position. If we left it here, we would have a string of little men printed across the screen, so as we update at each move, we must blank out the old graphic. This is accomplished at line 4470. Finally, to give a bit of arcade sound, line 4460 will make a little plodding noise at each step. You can experiment with the figures in the SOUND statement to customise it.

```
4370 S=STICK(0):DIRX=0:DIRY=0
4400 IF S=7 THEN LOCATE X+1,Y,Z:DIRX=1:D
IRY=0
4410 IF S=11 THEN LOCATE X-1,Y,Z:DIRX=-1
:DIRY=0
4420 IF S=13 THEN LOCATE X,Y+1,Z:DIRX=0:
DIRY=1
4430 IF S=14 THEN LOCATE X,Y-1,Z:DIRX=0:
DIRY=-1
4460 SOUND 0,120,8,10:SOUND 0,0,0,0
4470 POSITION X,Y:? #6;"♡"
4480 X=X+DIRX:Y=Y+DIRY
4490 POSITION X,Y:? #6;"m":REM INVERSE "
m"!
```

One last thing needs to be drawn on the display. We should really give a status report somewhere, and we have room at the bottom of the display for a few lines of text. The next lines will give our player a bit of information which could be useful. It is here, too, that the name of the monster in the next room will be printed.

```
2980 GRAPHICS 2
2990 REM TURN OFF CURSOR
3000 POKE 752,1
3010 POSITION 0,0:? "STRENGTH=";ST,"HIT%
=";HI
3020 ? "WOUNDS=";WO,,"DEF%=";DE
3030 ? "GOLD=";GO,,"LEVEL=";LEV
3040 RETURN
```

The only thing that needs explaining here is GO. This variable is updated each time our player moves onto a treasure chest.

Now type in lines 4440 and 4450. Line 4440 checks to see if another character has been hit by the player. If it has, the program goes on to the collision routine. Line 4450 checks for the player moving out of a door — if he has, then the program moves to the new room subroutine. Let's look at the collision first of all.

```
4440 IF Z<>0 THEN GOSUB 4950:DIRX=0:DIRY
=0
4450 IF X+DIRX=0 OR X+DIRX=19 OR Y+DIRY=
0 OR Y+DIRY=9 THEN GOTO 5290
```

There are several types of collisions to be catered for. The most common will be that of the player with the wall.

```
4960 IF Z>32 AND Z<39 THEN POP :GOTO 483
0:REM  CLEAR GOSUB STACK THEN GOTO COLLI
SION WITH MONSTER
4970 REM HEALING WELL(DRIES UP AFTER USE
)
4980 IF Z=ASC("'") THEN WO=0:HW=0:FOR I=
110 TO 90 STEP -1:SOUND 0,I,10,10:NEXT I
:SOUND 0,0,0,0
4990 REM ERASE WELL
5000 IF Z=ASC("'") THEN POSITION HX,HY:?
 #6;"♡":RETURN
```

```
5010 IF Z=ASC("Q") THEN SOUND 0,120,8,15
:FOR DEL=1 TO 5:NEXT DEL:SOUND 0,0,0,0:R
EM COLLISION WITH WALL
5020 IF Z=ASC("Q") THEN ST=ST-1:IF ST<=0
 THEN GOTO 6250
5030 IF Z=ASC("Q") THEN GOSUB 3010:RETUR
N
```

Line 5010 checks to see if Z equals the ASCII code for Q, which you'll remember is the character used for the wall graphics. If it does, in other words, if the player graphic has collided with a wall, then a little 'bloop' sound is made, and a short delay instigated. The player can't hope to go around bashing into walls without some penalty, so line 5020 decrements his strength (held in ST) by one

There are several other objects with which we can collide, however, and most of these are dealt with in the following lines.

```
5060 IF Z=ASC("X") AND LEV<>3 THEN LEV=L
EV+1:ROOM=ROOM+100:GOSUB 3010:DONE=0:GOS
UB 3240:RETURN
5070 IF Z=ASC("X") AND LEV=3 THEN LEV=LE
V-1:ROOM=ROOM-100:GOSUB 3010:DONE=0:GOSU
B 3240:RETURN
5080 IF Z=ASC("<") THEN GO=GO+TRES(ROOM)
:POSITION 10,4:? #6;"♡":TRES(ROOM)=0:GOS
UB 3010:RETURN
5090 IF Z=ASC("(") THEN GOSUB 5110:RETUR
N
5100 IF Z=ASC(")") THEN GOSUB 5200:RETUR
N
5110 FOR QQ=1 TO 5
5120 IF HOLD(QQ)<1 THEN GOTO 5150
5130 NEXT QQ
5140 RETURN
5150 HOLD(QQ)=WEAPON(ROOM)
5160 IF WEAPON(ROOM)=18 THEN WHAND=1
5170 WEAPON(ROOM)=0
5180 POSITION WX,WY:? #6;" "
5190 RETURN
```

```
5200 FOR QQ=1 TO 5
5210 IF HOLD(QQ)<1 THEN GOTO 5240
5220 NEXT QQ
5230 RETURN
5240 HOLD(QQ)=DROP(ROOM)
5250 IF DROP(ROOM)=18 THEN WHAND=1
5260 DROP(ROOM)=0
5270 POSITION DX,DY:? #6;"♡"
5280 RETURN
```

Line 5060 checks for a staircase graphic being hit. If it is, then the ROOM (the number of the room) and LEV (the level number) are both updated and the new room graphics printed (at subroutine 3240). If we are currently on Level 3, we can only go down, and this is taken care of by line 5070.

Line 5080 checks for a collision with a treasure chest, and if the test is positive updates GO, the gold held by the player, at 3010, the status report routine. Lines 5090 and 5100 do the same for collisions with weapons and dropped weapons. In this case, the program looks at the list of held weapons in HOLD, and updates accordingly if the player has room to carry another weapon. Line 5180 prints a blank in the room display where the weapon was (if the player is allowed to pick it up). The healing well, which will reset WO, the player's wounds, to zero, is dealt with on line 4980 — as you can see, this miraculous device may only be used once. You can change that, of course, if you wish! Another special case is seen at line 5160. You'll remember from the legend that the Wizard's hand figures in the story, and we can use this in our graphic adventure. With this in his possession, the player can use a spell without sapping his strength. WHAND is set at 1 if the hand is in the list of weapons held by the player (the hand is number 18 in the treasure string).

Now for the new room subroutine:

```
5300 IF X+DIRX=0 THEN ROOM=ROOM-1:X=18
5310 IF X+DIRX=19 THEN ROOM=ROOM+1:X=1
5320 IF Y+DIRY=9 THEN ROOM=ROOM+10:Y=1
5330 IF Y+DIRY=0 THEN ROOM=ROOM-10:Y=7
5340 DONE=0:LI=0
5350 GOTO 4330
```

At line 5340, DONE is set to 0, which will force the program to set up the new room's graphics, and LI is set to 0.

## Graphic violence

So that's dealt with all the graphics and movement — let's get down to some fighting!

```
4500 IF LI=1 THEN GOSUB 4520
```

This line checks the value of LI (which is set to 1 when the graphics are printed at lines 3440 and 3450). If LI is 1, it means that the monster is alive.

```
4510 GOTO 4360
4520 IF RND(0)<0.5 THEN RETURN
```

Line 4510 sends the program to 4360 where the Fire button is checked. We'll deal with this feature later. Line 4520 gives a random number up to 1. If it is less than 0.5, then the monster stays put, and the program goes back to get the next joystick input from the player. Otherwise, we go on to the monster move routine at line 4530.

```
4530 IF MX<X THEN GOSUB 4570:RETURN
4540 IF MX>X THEN GOSUB 4640:RETURN
4550 IF MY<Y THEN GOSUB 4710:RETURN
4560 IF MY>Y THEN GOSUB 4770:RETURN
4570 LOCATE MX+2,MY,Z:IF Z<>0 THEN POP :
GOTO 4830
4580 LOCATE MX+2,MY+1,Z
4590 IF Z<>0 THEN POP :GOTO 4830
4600 POSITION MX,MY:? #6;"♡♡":POSITION M
X,MY+1:? #6;"♡♡"
4610 MX=MX+1
4620 POSITION MX,MY:? #6;"!&":POSITION M
X,MY+1:? #6;"#$"
4630 RETURN
4640 LOCATE MX-1,MY,Z:IF Z<>0 THEN POP :
GOTO 4830
4650 LOCATE MX-1,MY+1,Z
4660 IF Z<>0 THEN POP :GOTO 4830
4670 POSITION MX,MY:? #6;"♡♡":POSITION M
X,MY+1:? #6;"♡♡"
4680 MX=MX-1
4690 POSITION MX,MY:? #6;"!&":POSITION M
```

```
X,MY+1:? #6;"#$"
4700 RETURN
4710 LOCATE MX,MY+2,Z:IF Z<>0 THEN POP :
GOTO 4830
4720 LOCATE MX+1,MY+2,Z:IF Z<>0 THEN POP
 :GOTO 4830
4730 POSITION MX,MY:? #6;"ᗢᗢ":POSITION M
X,MY+1:? #6;"ᗢᗢ"
4740 MY=MY+1
4750 POSITION MX,MY:? #6;"!&":POSITION M
X,MY+1:? #6;"#$"
4760 RETURN
4770 LOCATE MX,MY-1,Z:IF Z<>0 THEN POP :
GOTO 4830
4780 LOCATE MX+1,MY-1,Z:IF Z<>0 THEN POP
 :GOTO 4830
4790 POSITION MX,MY:? #6;"ᗢᗢ":POSITION M
X,M
4800 MY=MY-1
4810 POSITION MX,MY:? #6;"!&":POSITION M
X,MY+1:? #6;"#$"
4820 RETURN
4830 IF Z<>ASC("m") THEN SOUND 0,90,8,10
:FOR T=1 TO 5:NEXT T:SOUND 0,0,0,0:GOTO
4360
```

This is a large chunk, but all these lines are needed to move our monster and update its position. The LOCATE statements (for instance, at lines 4640 and 4710), are all checking for collisions. The only collisions affecting the monster are those with the player, and if the tests are positive, line 4830 looks to see if it is the player that the monster has hit. This is the only collision that will affect the monster, and, otherwise, the program will go back to see if the player wants to move. If a player has been hit, then there is a nice little collision sound, and the program goes on to the next routine, the combat routine.

```
4850 STRIKE=INT(RND(0)*100)+1
4860 IF STRIKE>DE THEN SOUND 0,160,6,15:
SOUND 0,0,0,0:WO=INT(WO+(STRIKE-DE)/4)
4870 GOSUB 3010
```

```
4880 IF WO>=100 THEN GOTO 6250
4890 STRIKE=INT(RND(0)*100)+1
4900 IF STRIKE<HI THEN SOUND 0,120,6,15:
SOUND 0,0,0,0:MST=INT(MST-(HI-STRIKE)/2)
4910 IF MST<=0 AND MONS(ROOM)=16 THEN EY
E=1
4920 IF MST<=0 THEN SOUND 0,14,10,15:SOU
ND 0,0,0,0:MONS(ROOM)=0:LI=0:POSITION MX
,MY:? #6;"♡♡"
4930 IF LI=0 THEN POSITION MX,MY+1:? #6;
"♡♡":MK=MK+1
4940 GOTO 4360
```

If the value of STRIKE, found at line 4850, is greater than the player's defence percentage (DE), then he has been hit by the monster. There is a hitting sound, and then WO, the player's wound status, is updated. Line 4870 sends the program back to the status report display and amends it accordingly. If the player has sustained more than 100 wounds, the game is over, and the final routine, which we haven't typed in yet, is gone too. The monster always goes first, but the player gets his go at line 4890. If the monster's strength (MST) falls past zero, then he is dead, and all the variables pertaining to him are reset at line 4920, as well as the graphics being blanked out. 4910 checks for the presence of the Lich at the location and sets the flag EYE to 1 to signify that the player is now in possession of The Eye of the Star Warrior.

# CHAPTER 15
# Pass The Menu Please

To enable him to fight the monster, we must allow the player to select his defence and attack weapons. The easiest way, and the best way from the player's point of view, of accomplishing this is by way of the joystick fire button.

Before the player moves, we can write a routine to check to see if the trigger is pressed — if it is, then the program goes to the menu routine.

```
4360 IF STRIG(0)=0 THEN GOSUB 1340:GOSUB
 3270

4380 IF S=15 AND LI=1 THEN GOSUB 4520
4390 IF S=15 THEN GOTO 4360
```

Line 4360 is the button check, and we need lines 4380 and 4390 to re-check if no movement of the joystick is detected. There is a penalty, in line 4380, for hanging about with a monster around!

If the button is pressed, on we go to the menu at line 1350.

```
1350 GRAPHICS 0
1360 REM RESTORE NORMAL CHARACTER SET
1370 POKE 756,224
1380 REM INHIBIT CURSOR
1390 POKE 752,1
1400 POSITION 9,0:? "USE JOYSTICK TO SEL
ECT"
1410 POSITION 0,2:? "UP    SELECT WEAPON
"
1420 POSITION 0,4:? "DOWN SELECT DEFENCE
"
1430 POSITION 0,6:? "LEFT       DROP ITEM
"
1440 POSITION 0,8:? "RIGHT        SEE MAP
"
```

```
1450 POSITION 6,12:? "PRESS FIRE BUTTON
TO RETURN"
1460 IF STRIG(0)=0 THEN RETURN
```

First of all, the entire screen is cleared of the room display, and the program goes back to the blue of graphic mode 0 before printing the menu. If the joystick is pressed now, the program returns to line 4360 — this in turn goes to line 3270, and reprints the room. There are a couple of lines that we can include in that routine to make the job easier for the program.

```
3370 IF MONS(ROOM)>0.9 THEN POSITION MX,
MY:? #6;"!&":POSITION MX,MY+1:? #6;"#$"
3380 IF WEAPON(ROOM)>0.9 THEN POSITION W
X,WY:? #6;"("
3390 IF DROP(ROOM)>0.9 THEN POSITION DX,
DY:? #6;")"
3400 GOSUB 3010
3410 RETURN
```

Line 3360, which we have typed in already, checks DONE. As the program has printed out all the graphics before we came to the menu routine, DONE equals 1, and the program does not have to go through all the checking procedure to see which monster is here, whether there is a Fire Pit here, and so on. It already knows all this, and can get on quickly to print the relevant graphics. Finally, returning to 4350, the program either goes to the menu again, or gets on with updating all the movements, depending on whether the button is pressed or not.

Meanwhile, back at the menu:

```
1470 S=STICK(0)
1480 IF S=7 THEN S=4
1490 IF S=11 THEN S=3
1500 IF S=13 THEN S=2
1510 IF S=14 THEN S=1
1520 IF S=15 THEN GOTO 1460
1530 ON S GOTO 1600,1860,2140,2470
1540 GOSUB 1560
1550 GOTO 1460
1560 POSITION 8,14:? "INVALID OPTION.TR
Y AGAIN"
```

```
1570 FOR DEL=1 TO 100:NEXT DEL
1580 POSITION 8,14:? "
     "
1590 RETURN
```

These lines find the direction in which the stick has been moved. If a diagonal has been tried, line 1560 prints a suitable message. A delay at line 1570 is set up, before 1580 prints a string of blank characters in place of the message.

```
1610 POSITION 0,0:? "↰SELECT ATTACK WEAP
ON"
1620 DEL=0
1630 FOR QQ=1 TO 5
1640 POSITION 0,QQ*2
1650 DUM=HOLD(QQ)
1660 IF DUM<1 THEN DEL=DEL+1:GOTO 1680
1670 ? DUM;".";TREA$(Y(DUM)+1,Y(DUM+1))
1680 NEXT QQ
1690 IF DEL=5 THEN DEL=0:GOTO 1340
1700 POSITION 0,12
1710 ? "INPUT NUMBER OF WEAPON";:INPUT A
TT
1720 FOR QQ=1 TO 5
1730 IF ATT=HOLD(QQ) THEN 1770
1740 NEXT QQ
1750 GOSUB 1560
1760 GOTO 1600
1770 POSITION 0,14:? "SELECTED WEAPON IS
:";TREA$(Y(ATT)+1,Y(ATT+1))
1780 IF ATT>18 AND WHAND<>1 THEN ST=ST-5
1790 IF ST<=0 THEN GOTO 6250
1800 FOR DEL=1 TO 100:NEXT DEL
1810 REM CHECK POTENCY OF ATTACK
1820 GOSUB 3090
1830 REM CLEAR SCREEN BEFORE RETURNING
1840 ? "↰"
1850 GOTO 1400
```

Now the player has selected the first option, and the program prints out a list of the weapons held. After the player has chosen the weapon he would like to use to attack the monster, the program then prints the name of the selected weapon before checking the weapon's attack capability against this particular monster. At line 1780, the program checks to see if a spell has been used (weapons numbered greater than 18), with the following line sending the program to the final routine if strength has fallen below 0. Then there is a delay before returning to the menu once more.

If the player has decided to choose a defence weapon, the next routine is encountered, at line 1870.

```
1870 POSITION 0,0:? "↖SELECT DEFENCE WEA
PON"
1880 DEL=0
1890 FOR QQ=1 TO 5
1900 POSITION 0,QQ*2
1910 DUM=HOLD(QQ)
1920 IF DUM<1 THEN DEL=DEL+1:GOTO 1940
1930 ? DUM;".";TREA$(Y(DUM)+1,Y(DUM+1))
1940 NEXT QQ
1950 IF DEL=5 THEN DEL=0:GOTO 1340
1960 POSITION 0,12
1970 ? "INPUT NUMBER OF DEFENCE";:INPUT
DEF
1980 FOR QQ=1 TO 5
1990 IF DEF=HOLD(QQ) THEN 2030
2000 NEXT QQ
2010 GOSUB 1560
2020 GOTO 1860
2030 POSITION 0,14:? "SELECTED DEFENCE I
S:";TREA$(Y(DEF)+1,Y(DEF+1))
2040 IF ATT>18 AND WHAND<>1 THEN ST=ST-5
2050 IF ST<=0 THEN GOTO 6250
2060 IF DEF=17 THEN WO=WO-20:IF WO<0 THE
N WO=0
2070 IF DEF=17 THEN HOLD(QQ)=0
2080 FOR DEL=1 TO 150:NEXT DEL
2090 REM CHECK POTENCY OF DEFENCE
```

```
2100 GOSUB 3160
2110 REM CLEAR SCREEN
2120 ? "↰"
2130 GOTO 1400
```

This routine is exactly the same as the one for attack weapons. At line 2060, we see a little routine concerning the healing water, which is treasure number 17. If the player selects this weapon as a defence, his wounds will be decreased by 20.

The 'drop weapon' sequence follows next. The first few lines trap an attempt to drop a weapon in a room where a dropped weapon already exists. If the room is clear of dropped weapons, then a list of held weapons is printed, and the player asked to input the number of weapons to drop. Lines 2400 and 2410 update our two variables holding the attack and defence percentages, while 2420 puts a blank into HOLD where the dropped weapon once was. Line 2430 sets up a position for the dropped weapon's graphic. There is a short delay (line 2440) before DONE is set to 0, thus forcing the program, on the next room set-up, to reprint all the graphics, thus ensuring that the dropped weapon will appear.

```
2150 IF DROP(ROOM)<1 THEN GOTO 2210
2160 POSITION 0,2:? "↰↰ROOM ALREADY HAS
ONE DROPPED OBJECT!"
2170 FOR DEL=1 TO 150:NEXT DEL
2180 REM CLEAR SCREEN
2190 ? "↰"
2200 GOTO 1400
2210 POSITION 0,0:? "↰SELECT WEAPON TO D
ROP"
2220 DEL=0
2230 FOR QQ=1 TO 5
2240 POSITION 0,QQ*2
2250 DUM=HOLD(QQ)
2260 IF DUM<1 THEN DEL=DEL+1:GOTO 2280
2270 ? DUM;".";TREA$(Y(DUM)+1,Y(DUM+1))
2280 NEXT QQ
2290 IF DEL=5 THEN DEL=0:GOTO 1340
2300 POSITION 0,12
2310 ? "INPUT NUMBER OF WEAPON";:INPUT D
```

```
UM
2320 FOR QQ=1 TO 5
2330 IF DUM=HOLD(QQ) THEN 2370
2340 NEXT QQ
2350 GOSUB 1560
2360 GOTO 2140
2370 POSITION 0,14:? "DROPPED WEAPON IS:
";TREA$(Y(DUM)+1,Y(DUM+1))
2380 DROP(ROOM)=HOLD(QQ)
2390 IF DUM=18 THEN WHAND=0
2400 IF ATT=HOLD(QQ) THEN ATT=0
2410 IF DEF=HOLD(QQ) THEN DEF=0
2420 HOLD(QQ)=0
2430 DX=2:DY=2
2440 FOR DEL=1 TO 150:NEXT DEL
2450 GOSUB 4100:DONE=0
2460 GOTO 1340
```

The final selection on our menu is the map. Asking for this, the player will be presented with a map of the level he is currently on, with each room depicted. The rooms with weapons are shown, as is the room containing the staircase. The position of the walls are shown, and the current position of the player. Line 2500 reselects our alternate character set, and sets up the start position of the map graphics. Lines 2540 and 2550 print the graphics for a wall and weapon room respectively. Line 2590 sends the program to 2860 to print out the dropped weapons.

```
2480 TRAP 2470
2490 ? "↖"
2500 POKE 756,LOC
2510 FOR T=0 TO 9
2520 FOR QQ=1 TO 10
2530 POSITION QQ,T
2540 IF C((T*10)+QQ+(LEV-1)*100)=999 THE
N ? "Q":GOTO 2570:REM INVERSE "Q"!
2550 IF WEAPON((T*10)+QQ+(LEV-1)*100)>0.
9 THEN ? "Z":GOTO 2570
2560 ? " "
2570 NEXT QQ
```

```
2580 NEXT T
2590 GOSUB 2860
2860 FOR T=1 TO 100
2870 IF DROP(T*LEV)>0.9 THEN GOTO 2900
2880 NEXT T
2890 RETURN
2900 P=INT(T/100)
2910 C=T-P*100
2920 QQ=INT(C/10)
2930 P=((C/10)-QQ)*10
2940 POSITION P,QQ
2950 ? "Z"
2960 GOTO 2880
```

Lines 2600–2650 print out the position of the player, with the following lines, 2660–2740, checking for the staircase position.

```
2600 T=INT(ROOM/100)
2610 C=ROOM-T*100
2620 QQ=INT(C/10)
2630 T=((C/10)-QQ)*10
2640 POSITION T,QQ
2650 ? "M"
2660 IF LEV=1 THEN T=SONE
2670 IF LEV=2 THEN T=STWO
2680 IF LEV=3 THEN T=STHREE
2690 C=INT(T/100)
2700 P=T-C*100
2710 QQ=INT(P/10)
2720 C=((P/10)-QQ)*10
2730 POSITION C,QQ
2740 ? "X"
```

Then a little display is drawn next to the map, showing the player what all the little pictures mean. Line 2810 performs the fire button check, before the normal character set is restored for the menu's use.

```
2750 TRAP 40000
2760 POSITION 15,1:? "M=PLAYER"
2770 POSITION 15,3:? "Z=WEAPON"
```

```
2780 POSITION 15,5:? "Q=WALL":REM INVERS
E "Q"!
2790 POSITION 15,7:? "X=STAIRCASE"
2800 POSITION 6,13:? "PRESS FIRE BUTTON
TO RETURN"
2810 IF STRIG(0)=1 THEN 2810
2820 REM RESTORE CHARCTER SET BEFORE RET
URNING
2830 POKE 756,224
2840 ? "↖"
2850 GOTO 1400
```

Just one final thing needs to be done, and that is to type in the endgame. This is reached in one of several ways — the player's strength (held in ST) can fall below zero, or his wounds rise above 100 (WO), or, a point we haven't yet covered, he can succeed in the game by throwing the Eye into the fire pit. Line 5050 accomplishes this. To give the program a little more spice, we could type in line 5040, which ensures that if the player bumps into the fire pit without the Eye, he is killed.

```
5040 IF Z=ASC("@") AND EYE=0 THEN GOTO 6
250
5050 IF Z=ASC("@") AND EYE=1 THEN GOTO 6
380:REM COMPLETED MISSION
```

and this is the endgame:

```
6250 FOR QQ=15 TO 1 STEP -0.2
6260 SETCOLOR 0,3,15:SETCOLOR 0,0,0
6270 SOUND 0,120,8,QQ
6280 NEXT QQ
6290 SOUND 0,0,0,0
6300 GRAPHICS 2
6310 POSITION 0,1:? #6;"YOU APPEAR TO BE
"
6320 POSITION 0,2:? #6;"DEAD!"
6330 POSITION 0,3:? #6;"monsters killed:
";MK
6340 POSITION 0,4:? #6;"gold collected:"
;GO
6350 POSITION 0,8:? #6;"ANOTHER GO?":INP
UT DUM$
```

```
6360 IF DUM$(1,1)="Y" THEN RUN
6370 GOTO 6350
6380 FOR QQ=15 TO 1 STEP 0.2
6390 SETCOLOR 0,0,15:SETCOLOR 0,0,0
6400 SOUND 0,14,10,QQ
6410 NEXT QQ
6420 SOUND 0,0,0,0
6430 GRAPHICS 2
6440 POSITION 0,0:? "THIS ADVENTURE IS "
6450 POSITION 0,1:? "OVER."
6460 POSITION 0,2:? "CONGRATULATIONS"
6470 END
```

All this is fairly obvious — the messages have been deliberately left fairly sparse. They could do with a bit of dressing-up, in both the sound department and in the text.

# The Eye of the Star Warrior

# APPENDIX A
# Instructions

The object of the adventure is to explore the three levels of Agor's temple, find the enchanted stone and destroy it.

**Movement** is accomplished via your joystick. To move into a new room, simply walk through the door!

**Treasure** In many rooms the player will find a blue treasure chest — by moving on to this, he will collect all the gold therein. The status report in the text window will be duly updated.

**Weapons** are scattered throughout the temple complex, and may be picked up simply by moving on to them. There is a maximum of five that can be held, however. Each weapon (including spells), has its own attack and defence capability — for instance, a spade is not going to be of much use against a vampire, but will probably be of some use against another monster.

**Spells** are very potent weapons, and as such use up five strength points each time they are used in combat.

**Wizard's Hand** The hand of Aldous, the good wizard, may be found somewhere within the complex. While in the player's possession, it will allow spells to be used without the usual five-point penalty.

**Combat** Monsters will often be met. Before a new room is entered, the text window beneath the room display will give the name of the monster in the next room. Using this information, the player may select attack and defence weapons.

In the text window, you will also see the HIT percentage and attack percentage of the weapons you have chosen. Use this information to see if you have chosen wisely, and decide whether to retreat or press the attack.

**Strength and Wounds** Also in the text window, you will see an update on your strength and the number of wounds you have. Strength starts at 100 and is decremented throughout the game — each time you hit a wall, by one point, and, if you tackle a very strong monster and start losing, strength will be sapped rapidly. Wounds, of course, start at zero and are increased as your battles go badly.

**Healing Well** Having found this, the player may be healed of a certain number of his wounds. If wounds reach 100, or strength reaches 0, then the player is dead. Once the healing well has been dipped into, it dries up and cannot be used again.

**Menu** By pressing the fire button on the joystick, the player is presented with a small menu. This has several options:
1. Select Weapon: Selects attack weapon
2. Select Defence: Selects defence weapon
3. Drop Item: There is a maximum of one dropped weapon per room
4. See Map: This option allows the player to see a map of the present level. On the map will be printed each room in which there is a weapon, including dropped weapons, as well as the player's present position and the staircase.

**Pause** The menu option acts as a pause, as all action freezes while the menu is being used.

**The Eye** is the enchanted stone which is held by the Lich, somewhere on the third level. The player must defeat the Lich, when he will automatically find himself in possession of the Eye. He must then battle his way back to the second level, find the fire pit, and move on to it. This will accomplish the mission. If the fire pit is moved on to by the player without the Eye, instant death will occur!

# APPENDIX B
# Full Listing

Here is the full listing to *The Eye of the Star Warrior*. There are a couple of things to note while typing this in.

First, many of the routines have been left as a framework, upon which the reader can fashion his own program. Thus, the endgame is rather brief, and will reward a more detailed working. The menu, also, lends itself to a rather more colourful treatment.

Second, and more important, care must be taken when typing some of the lines. These are the lines that use the CTRL key — for some reason best known to themselves, Atari have not seen fit to design their printers to recognise these codes. So, when you see :"", then type "[CTRL,]". This will show up in your final listing as a heart symbol, which prints a blank space on the screen.

The "clear screen" symbol (obtained by typing "[ESC,CTRL, CLEAR]") is unfortunately not recognised either, so again does not show up. Lines 1610 (before the text), 1870 (again before the text) and 2120 are just some of the lines where this character will be necessary.

The entire program uses all but some 11K of the 48K available with the larger memory. Unfortunately, without some re-working, possibly in the room storage, disk owners will not be able to use their drives to SAVE or LOAD the program, as DOS takes up too much room.

Feel free to change the graphics in any way you like — the sound routines can also be experimented with.

One or two lines have not been included in the analysis of the program — a couple of GOSUBs, the odd RETURN! So it is essential that you check this complete listing to make sure that you have typed in every line.

```
10 REM ****ADVENTURE PROGRAM****
20 REM copyright (@) 1983 by
            Gary Radburn.
30 GRAPHICS 2:POKE 710,0:POKE 708,15
40 POSITION 7,0:? #6;"PLEASE":POSITION 8
```

```
,2:? #6;"WAIT":POSITION 7,4:? #6;"2 MINS
"
50 REM TURN ON ATTRACT MODE
60 POKE 77,128
70 GOSUB 610
80 DIM ROOM$(200),HOLD(5),DROP(300)
90 ST=50
100 FOR T=1 TO 5:HOLD(T)=0:NEXT T
110 REM CLEAR ROOM$
120 ROOM$(1)="":ROOM$(200)="":ROOM$(2)=R
OOM$
130 ROOM$(1,20)="QQQQQQQQQQQQQQQQQQQQ"
140 FOR I=1 TO 8
150 ROOM$(20*I+1,20*I+1)="Q":ROOM$(20*I+
20,20*I+20)="Q"
160 NEXT I
170 ROOM$(181,200)=ROOM$(1,20)
180 DIM MONS(300),TRES(300)
190 FOR R=1 TO 300
200 M=INT(RND(0)*10)
210 IF C(R)=999 THEN 270
220 IF M<INT(R/100+1)*2 THEN MONS(R)=1:G
OTO 240
230 MONS(R)=0
240 T=INT(RND(0)*10)
250 IF T>5 THEN TRES(R)=INT((RND(0)*(R+5
0))/5)*5:GOTO 270
260 TRES(R)=0
270 NEXT R
280 DIM WEAPON(300)
290 FOR Q=1 TO 300
300 WEAPON(Q)=0
310 NEXT Q
320 FOR T=0 TO 2
330 FOR I=1 TO 25
340 PLACE=INT(RND(0)*100)+(T)*100+1
350 IF C(PLACE)=999 THEN GOTO 340
360 IF WEAPON(PLACE)<>0 THEN 340
```

```
370 WEAPON(PLACE)=I
380 NEXT I
390 NEXT T
400 DATA skeleton,mummy,demon,zombie,ele
mental,vampire,banshee,wraith,dragon,wer
ewolf,cyclops,sandman,harpie
410 DATA serpent,balrog,lich
420 DIM MONSTER$(160),DUM$(10)
430 RESTORE 400
440 FOR MONSTER=1 TO 16
450 READ DUM$
460 CHAR=LEN(DUM$)
470 MONSTER$((MONSTER-1)*10+1)=CHR$(CHAR
)
480 MONSTER$((MONSTER-1)*10+2)=DUM$
490 NEXT MONSTER
500 FOR T=1 TO 100
510 IF MONS(T)=1 THEN MONS(T)=INT(RND(0)
*10)+1
520 NEXT T
530 FOR T=101 TO 300
540 IF MONS(T)=1 THEN MONS(T)=INT(RND(0)
*15)+1
550 NEXT T
560 REM PLACE LICH IN FINAL LEVEL
570 PLACE=INT(RND(0)*100)+201
580 IF C(PLACE)<>999 THEN MONS(PLACE)=16
:GOTO 600
590 GOTO 570
600 X=1:Y=4:POKE 77,0:GOTO 4330
610 DIM C(300)
620 LEV=1:ROOM=12
630 FOR A=1 TO 10:C(A)=999:NEXT A
640 FOR A=90 TO 110:C(A)=999:NEXT A
650 FOR A=190 TO 210:C(A)=999:NEXT A
660 FOR A=290 TO 300:C(A)=999:NEXT A
670 FOR A=11 TO 291 STEP 10:C(A)=999:NEX
T A
```

```
680 FOR A=10 TO 300 STEP 10:C(A)=999:NEX
T A
690 B=INT(RND(0)*30)+40
700 FOR A=1 TO B
710 D=INT(RND(0)*280)+11
720 IF C(D)=999 THEN GOTO 710
730 IF C(D+11)=999 THEN GOTO 710
740 IF C(D-11)=999 OR C(D-9)=999 OR C(D+
9)=999 THEN GOTO 760
750 C(D)=999
760 NEXT A
770 D=0
780 FOR A=10 TO 290
790 IF C(A)=999 THEN GOTO 820
800 D=D+1
810 C(A)=D
820 NEXT A
830 REM SET UP STAIRS,HEALING WELL,FIRE
PIT
840 SONE=INT(RND(0)*90)+10
850 IF C(SONE)=999 THEN 840
860 STWO=INT(RND(0)*90)+110
870 IF C(STWO)=999 THEN 860
880 STHREE=INT(RND(0)*90)+210
890 IF C(STHREE)=999 THEN 880
900 HW=INT(RND(0)*300)+1
910 IF C(HW)=999 THEN GOTO 900
920 FP=INT(RND(0)*200)+100
930 IF C(FP)=999 OR FP=HW THEN GOTO 920
940 DIM TREA$(375),OM$(15),Y(26)
950 J=1:TREA$=""
960 RESTORE 1010
970 FOR N=1 TO 26
980 READ OM$
990 TREA$(J)=OM$:Y(N)=LEN(TREA$):J=1+Y(N
)
1000 NEXT N
1010 DATA DUMMY
```

```
1020 DATA SPADE,FIRE WHIP,SWORD,SILVER S
WORD,SILVER STAFF,SAINTLY STAFF,TALISMAN
,CROSS,SHIELD,TORCH,INVISIBLE CLOAK
1030 DATA CLUB,HOLY WATER,BOW AND ARROWS
,MAGIC SHIELD,EMPTY BOTTLE,HEALING WATER
,WIZARDS HAND,TELEPORT,FORCESHIELD
1040 DATA PSYCHIC SHIELD,LIGHTNING BOLT,
STONE SPELL,LIMBO SPELL,STONE
1050 DIM ATT$(320),PM$(20),Z(17)
1060 J=1:ATT$=""
1070 RESTORE 1120
1080 FOR N=1 TO 17
1090 READ PM$
1100 ATT$(J)=PM$:Z(N)=LEN(ATT$):J=1+Z(N)
1110 NEXT N
1120 DATA DUMMY
1130 DATA 0103040506071222 24,02071022232
4,06071324,0207102324,06071324,020405061
013222324
1140 DATA 060724,0204050610132223 24,0304
222324,0204060710222324,030414222324,010
5060712132223 24,02030414222324
1150 DATA 02030410222324,03040714222324,
2324
1160 DIM DEF$(342),QM$(18),A(20)
1170 J=1:DEF$=""
1180 RESTORE 1230
1190 FOR N=1 TO 20
1200 READ QM$
1210 DEF$(J)=QM$:A(N)=LEN(DEF$):J=1+A(N)
1220 NEXT N
1230 DATA DUMMY
1240 DATA 09111519,020910111519,07081519
20,0207091011 1519,071319,020708091011131
519
1250 DATA 0207080910111314 19,0708151920,
09111519,020709101113 1519,020910111519
1260 DATA 091113151920,020910111519,0209
```

```
10111519,07091519,192021,1520,21,21
1270 DATA 091113151920,020910111519,0209
10111519,07091519,192021,1520,21,21
1280 REM SET UP PLACE FOR CUSTOM CHARACT
ER SET
1290 CH=(PEEK(106)-8)*256:CHORG=57344
1300 FOR I=0 TO 1024:POKE CH+I,PEEK(CHOR
G+I):NEXT I
1310 LOC=CH/256
1320 GOSUB 3060
1330 RETURN
1340 REM ****MENU SUBROUTINE****
1350 GRAPHICS 0
1360 REM RESTORE NORMAL CHARACTER SET
1370 POKE 756,224
1380 REM INHIBIT CURSOR
1390 POKE 752,1
1400 POSITION 9,0:? "USE JOYSTICK TO SEL
ECT"
1410 POSITION 0,2:? "UP    SELECT WEAPON
"
1420 POSITION 0,4:? "DOWN SELECT DEFENCE
"
1430 POSITION 0,6:? "LEFT      DROP ITEM
"
1440 POSITION 0,8:? "RIGHT       SEE MAP
"
1450 POSITION 6,12:? "PRESS FIRE BUTTON
TO RETURN"
1460 IF STRIG(0)=0 THEN RETURN
1470 S=STICK(0)
1480 IF S=7 THEN S=4
1490 IF S=11 THEN S=3
1500 IF S=13 THEN S=2
1510 IF S=14 THEN S=1
1520 IF S=15 THEN GOTO 1460
1530 ON S GOTO 1600,1860,2140,2470
1540 GOSUB 1560
```

```
1550 GOTO 1460
1560 POSITION 8,14:? " INVALID OPTION. TR
Y AGAIN"
1570 FOR DEL=1 TO 100:NEXT DEL
1580 POSITION 8,14:? "
     "
1590 RETURN
1600 REM SELECT ATTACK WEAPON
1610 POSITION 0,0:? " SELECT ATTACK WEAP
ON"
1620 DEL=0
1630 FOR QQ=1 TO 5
1640 POSITION 0,QQ*2
1650 DUM=HOLD(QQ)
1660 IF DUM<1 THEN DEL=DEL+1:GOTO 1680
1670 ? DUM;".";TREA$(Y(DUM)+1,Y(DUM+1))
1680 NEXT QQ
1690 IF DEL=5 THEN DEL=0:GOTO 1340
1700 POSITION 0,12
1710 ? "INPUT NUMBER OF WEAPON";:INPUT A
TT
1720 FOR QQ=1 TO 5
1730 IF ATT=HOLD(QQ) THEN 1770
1740 NEXT QQ
1750 GOSUB 1560
1760 GOTO 1600
1770 POSITION 0,14:? "SELECTED WEAPON IS
:";TREA$(Y(ATT)+1,Y(ATT+1))
1780 IF ATT>18 AND WHAND<>1 THEN ST=ST-5
1790 IF ST<=0 THEN GOTO 6250
1800 FOR DEL=1 TO 100:NEXT DEL
1810 REM CHECK POTENCY OF ATTACK
1820 GOSUB 3090
1830 REM CLEAR SCREEN BEFORE RETURNING
1840 ? " "
1850 GOTO 1400
1860 REM SELECT DEFENCE WEAPON
1870 POSITION 0,0:? " SELECT DEFENCE WEA
```

```
PON"
1880 DEL=0
1890 FOR QQ=1 TO 5
1900 POSITION 0,QQ*2
1910 DUM=HOLD(QQ)
1920 IF DUM<1 THEN DEL=DEL+1:GOTO 1940
1930 ? DUM;".";TREA$(Y(DUM)+1,Y(DUM+1))
1940 NEXT QQ
1950 IF DEL=5 THEN DEL=0:GOTO 1340
1960 POSITION 0,12
1970 ? "INPUT NUMBER OF DEFENCE";:INPUT
DEF
1980 FOR QQ=1 TO 5
1990 IF DEF=HOLD(QQ) THEN 2030
2000 NEXT QQ
2010 GOSUB 1560
2020 GOTO 1860
2030 POSITION 0,14:? "SELECTED DEFENCE I
S:";TREA$(Y(DEF)+1,Y(DEF+1))
2040 IF ATT>18 AND WHAND<>1 THEN ST=ST-5
2050 IF ST<=0 THEN GOTO 6250
2060 IF DEF=17 THEN WO=WO-20:IF WO<0 THE
N WO=0
2070 IF DEF=17 THEN HOLD(QQ)=0
2080 FOR DEL=1 TO 150:NEXT DEL
2090 REM CHECK POTENCY OF DEFENCE
2100 GOSUB 3160
2110 REM CLEAR SCREEN
2120 ? " "
2130 GOTO 1400
2140 REM DROP WEAPON
2150 IF DROP(ROOM)<1 THEN GOTO 2210
2160 POSITION 0,2:? "  ROOM ALREADY HAS
ONE DROPPED OBJECT!"
2170 FOR DEL=1 TO 150:NEXT DEL
2180 REM CLEAR SCREEN
2190 ? " "
2200 GOTO 1400
```

```
2210 POSITION 0,0:? " SELECT WEAPON TO D
ROP"
2220 DEL=0
2230 FOR QQ=1 TO 5
2240 POSITION 0,QQ*2
2250 DUM=HOLD(QQ)
2260 IF DUM<1 THEN DEL=DEL+1:GOTO 2280
2270 ? DUM;".";TREA$(Y(DUM)+1,Y(DUM+1))
2280 NEXT QQ
2290 IF DEL=5 THEN DEL=0:GOTO 1340
2300 POSITION 0,12
2310 ? "INPUT NUMBER OF WEAPON";:INPUT D
UM
2320 FOR QQ=1 TO 5
2330 IF DUM=HOLD(QQ) THEN 2370
2340 NEXT QQ
2350 GOSUB 1560
2360 GOTO 2140
2370 POSITION 0,14:? "DROPPED WEAPON IS:
";TREA$(Y(DUM)+1,Y(DUM+1))
2380 DROP(ROOM)=HOLD(QQ)
2390 IF DUM=18 THEN WHAND=0
2400 IF ATT=HOLD(QQ) THEN ATT=0
2410 IF DEF=HOLD(QQ) THEN DEF=0
2420 HOLD(QQ)=0
2430 DX=2:DY=2
2440 FOR DEL=1 TO 150:NEXT DEL
2450 GOSUB 4100:DONE=0
2460 GOTO 1340
2470 REM MAP DRAWER
2480 TRAP 2470
2490 ? " "
2500 POKE 756,LOC
2510 FOR T=0 TO 9
2520 FOR QQ=1 TO 10
2530 POSITION QQ,T
2540 IF C((T*10)+QQ+(LEV-1)*100)=999 THE
N ? "Q":GOTO 2570
```

```
2550 IF WEAPON((T*10)+QQ+(LEV-1)*100)>0.
9 THEN ? "Ƶ":GOTO 2570
2560 ? " "
2570 NEXT QQ
2580 NEXT T
2590 GOSUB 2860
2600 T=INT(ROOM/100)
2610 C=ROOM-T*100
2620 QQ=INT(C/10)
2630 T=((C/10)-QQ)*10
2640 POSITION T,QQ
2650 ? "M"
2660 IF LEV=1 THEN T=SONE
2670 IF LEV=2 THEN T=STWO
2680 IF LEV=3 THEN T=STHREE
2690 C=INT(T/100)
2700 P=T-C*100
2710 QQ=INT(P/10)
2720 C=((P/10)-QQ)*10
2730 POSITION C,QQ
2740 ? "X"
2750 TRAP 40000
2760 POSITION 15,1:? "M=PLAYER"
2770 POSITION 15,3:? "Ƶ=WEAPON"
2780 POSITION 15,5:? "Q=WALL"
2790 POSITION 15,7:? "X=STAIRCASE"
2800 POSITION 6,13:? "PRESS FIRE BUTTON
TO RETURN"
2810 IF STRIG(0)=1 THEN 2810
2820 REM RESTORE CHARCTER SET BEFORE RET
URNING
2830 POKE 756,224
2840 ? " "
2850 GOTO 1400
2860 FOR T=1 TO 100
2870 IF DROP(T*LEV)>0.9 THEN GOTO 2900
2880 NEXT T
2890 RETURN
```

```
2900 P=INT(T/100)
2910 C=T-P*100
2920 QQ=INT(C/10)
2930 P=((C/10)-QQ)*10
2940 POSITION P,QQ
2950 ? "Z"
2960 GOTO 2880
2970 REM MAIN PROGRAM AREA
2980 GRAPHICS 2
2990 REM TURN OFF CURSOR
3000 POKE 752,1
3010 POSITION 0,0:? "STRENGTH=";ST,"HIT%
=";HI
3020 ? "WOUNDS=";WO,,"DEF%=";DE
3030 ? "GOLD=";GO,,"LEVEL=";LEV
3040 RETURN
3050 REM GOSUB TO SET UP INITIAL GRAPHIC
S
3060 GOSUB 3680
3070 RETURN
3080 REM SET UP ATTACK AND DEFENCE
3090 QQ=MONS(ROOM)
3100 IF QQ=0 THEN HI=0:DE=0:RETURN
3110 PM$=ATT$(Z(QQ)+1,Z(QQ+1))
3120 FOR Q=1 TO LEN(PM$) STEP 2
3130 IF ATT=VAL(PM$(Q,Q+1)) THEN HI=1:GO
TO 3160
3140 NEXT Q
3150 HI=0
3160 PM$=DEF$(A(QQ)+1,A(QQ+1))
3170 FOR Q=1 TO LEN(PM$) STEP 2
3180 IF DEF=VAL(PM$(Q,Q+1)) THEN DE=1:GO
TO 3210
3190 NEXT Q
3200 DE=0
3210 DE=DE*(INT(RND(0)*22)+78/LEV)
3220 HI=HI*(INT(RND(0)*22)+78/LEV)
3230 RETURN
```

```
3240 IF WEAPON(ROOM)>0.9 THEN GOSUB 4030
3250 IF DROP(ROOM)>0.9 THEN GOSUB 4100
3260 IF MONS(ROOM)>0.9 THEN GOSUB 4170
3270 GRAPHICS 2:POSITION 0,0:POKE 756,LO
C:? #6;ROOM$
3280 IF C(ROOM-1)<>999 THEN POSITION 0,4
:? #6;""
3290 IF C(ROOM+1)<>999 THEN POSITION 19,
4:? #6;""
3300 IF C(ROOM+10)<>999 THEN POSITION 10
,9:? #6;""
3310 IF C(ROOM-10)<>999 THEN POSITION 10
,0:? #6;""
* 3320 IF TRES(ROOM)<>0.9 THEN POSITION 10,
5:? #6;"<"
* 3330 IF LEV=1 AND SONE=ROOM THEN POSITIO
N 18,1:? #6;"X"
* 3340 IF LEV=2 AND STWO=ROOM THEN POSITIO
N 18,1:? #6;"X"
* 3350 IF LEV=3 AND STHREE=ROOM THEN POSIT
ION 18,1:? #6;"X"
3360 IF DONE=0 THEN GOSUB 3420
3370 IF MONS(ROOM)>0.9 THEN POSITION MX,
MY:? #6;"!&":POSITION MX,MY+1:? #6;"#$"
3380 IF WEAPON(ROOM)>0.9 THEN POSITION W
X,WY:? #6;"("
3390 IF DROP(ROOM)>0.9 THEN POSITION DX,
DY:? #6;")"
3400 GOSUB 3010
* 3410 RETURN
3420 IF MONS(ROOM)<0.9 THEN GOTO 3470
3430 MX=8:MY=3
3440 POSITION MX,MY:? #6;"!&":POSITION M
X,MY+1:? #6;"#$"
3450 LI=1:REM SET FLAG THAT MONSTER IS A
LIVE
3460 MST=INT(RND(0)*MONS(ROOM))+(20+MONS
(ROOM))*LEV:REM ALGORITHM FOR MONSTER ST
```

```
RENGTH
3470 IF WEAPON(ROOM)<0.9 THEN GOTO 3520
3480 WX=INT(RND(0)*17)+2:WY=INT(RND(0)*6
)+2
3490 LOCATE WX,WY,QQ
3500 IF QQ=0 THEN POSITION WX,WY:? #6;"(
":GOTO 3520
3510 GOTO 3480
3520 IF DROP(ROOM)<0.9 THEN GOTO 3570
3530 DX=INT(RND(0)*17)+2:DY=INT(RND(0)*6
)+2
3540 LOCATE DX,DY,QQ
3550 IF QQ=0 THEN POSITION DX,DY:? #6;")
":GOTO 3570
3560 GOTO 3530
3570 IF HW<>ROOM THEN GOTO 3620
3580 HX=INT(RND(0)*17)+2:HY=INT(RND(0)*6
)+2
3590 LOCATE HX,HY,QQ
3600 IF QQ=0 THEN POSITION HX,HY:? #6;"'
":GOTO 3620
3610 GOTO 3580
3620 IF FP<>ROOM THEN GOTO 3670
3630 FX=INT(RND(0)*17)+2:FY=INT(RND(0)*6
)+2
3640 LOCATE FX,FY,QQ
3650 IF QQ=0 THEN POSITION FX,FY:? #6;"@
":GOTO 3670
3660 GOTO 3630
3670 DONE=1:RETURN
3680 REM SET UP CHARACTER SET
3690 DIM C$(13)
3700 C$="MXQ?!&#$'@()<"
3710 RESTORE 3810
3720 FOR T=1 TO LEN(C$)
3730 CHPOS=CH+(ASC(C$(T))-32)*8
3740 FOR I=0 TO 7
3750 READ QQ
```

```
3760 POKE CHPOS+I,QQ
3770 NEXT I
3780 NEXT T
3790 REM DATA FOR CHARACTERS THAT DO NOT
 CHANGE
3800 REM PLAYER
3810 DATA 56,56,16,254,16,40,68,130
3820 REM STAIRCASE
3830 DATA 0,3,3,15,15,63,63,255
3840 REM WALL
3850 DATA 219,0,102,0,219,0,102,0
3860 REM TREASURE ON MAP
3870 DATA 247,129,149,137,149,161,193,24
7
3880 REM FOUR CHARACTERS TO BE USED FOR
MONSTER ARE FILLED WITH ZEROS
3890 DATA 0,0,0,0,0,0,0,0
3900 DATA 0,0,0,0,0,0,0,0
3910 DATA 0,0,0,0,0,0,0,0
3920 DATA 0,0,0,0,0,0,0,0
3930 REM HEALING WELL
3940 DATA 0,28,62,126,124,56,24,0
3950 REM FIRE PIT
3960 DATA 217,202,2,176,53,128,109,108
3970 REM DUMMY CHARACTERS FOR TREASURE A
ND DROPPED TREASURE
3980 DATA 0,0,0,0,0,0,0,0
3990 DATA 0,0,0,0,0,0,0,0
4000 REM TREASURE CHEST
4010 DATA 0,24,60,60,126,126,126,126
4020 RETURN
4030 RESTORE 5350+WEAPON(ROOM)*10
4040 CHPOS=CH+(ASC("(")-32)*8
4050 FOR I=0 TO 7
4060 READ QQ
4070 POKE CHPOS+I,QQ
4080 NEXT I
4090 RETURN
```

```
4100 RESTORE 5350+DROP(ROOM)*10
4110 CHPOS=CH+(ASC(")")-32)*8
4120 FOR I=0 TO 7
4130 READ QQ
4140 POKE CHPOS+I,QQ
4150 NEXT I
4160 RETURN
4170 RESTORE 5600+MONS(ROOM)*40-30
4180 ? " ":POSITION 5,0:? MONSTER$(MONS(
ROOM)*10-8,MONS(ROOM)*10-9+ASC(MONSTER$(
MONS(ROOM)*10-9)))
4190 CHPOS=CH+(ASC("!")-32)*8
4200 GOSUB 4280
4210 CHPOS=CH+(ASC("&")-32)*8
4220 GOSUB 4280
4230 CHPOS=CH+(ASC("#")-32)*8
4240 GOSUB 4280
4250 CHPOS=CH+(ASC("$")-32)*8
4260 GOSUB 4280
4270 RETURN
4280 FOR I=0 TO 7
4290 READ QQ
4300 POKE CHPOS+I,QQ
4310 NEXT I
4320 RETURN
4330 REM LET'S MOVE THE PLAYER+MONSTER &
 HAVE A FIGHT!
4340 GOSUB 3090
4350 GOSUB 3240:POKE 756,LOC:POSITION X,
Y:? #6;"m"
4360 IF STRIG(0)=0 THEN GOSUB 1340:GOSUB
 3270
4370 S=STICK(0):DIRX=0:DIRY=0
4380 IF S=15 AND LI=1 THEN GOSUB 4520
4390 IF S=15 THEN GOTO 4360
4400 IF S=7 THEN LOCATE X+1,Y,Z:DIRX=1:D
IRY=0
4410 IF S=11 THEN LOCATE X-1,Y,Z:DIRX=-1
```

```
:DIRY=0
4420 IF S=13 THEN LOCATE X,Y+1,Z:DIRX=0:
DIRY=1
4430 IF S=14 THEN LOCATE X,Y-1,Z:DIRX=0:
DIRY=-1
4440 IF Z<>0 THEN GOSUB 4950:DIRX=0:DIRY
=0
4450 IF X+DIRX=0 OR X+DIRX=19 OR Y+DIRY=
0 OR Y+DIRY=9 THEN GOTO 5290
4460 SOUND 0,120,8,10:SOUND 0,0,0,0
4470 POSITION X,Y:? #6;""
4480 X=X+DIRX:Y=Y+DIRY
4490 POSITION X,Y:? #6;"m"
4500 IF LI=1 THEN GOSUB 4520
4510 GOTO 4360
4520 IF RND(0)<0.5 THEN RETURN
4530 IF MX<X THEN GOSUB 4570:RETURN
4540 IF MX>X THEN GOSUB 4640:RETURN
4550 IF MY<Y THEN GOSUB 4710:RETURN
4560 IF MY>Y THEN GOSUB 4770:RETURN
4570 LOCATE MX+2,MY,Z:IF Z<>0 THEN POP :
GOTO 4830
4580 LOCATE MX+2,MY+1,Z
4590 IF Z<>0 THEN POP :GOTO 4830
4600 POSITION MX,MY:? #6;"":POSITION MX,
MY+1:? #6;""
4610 MX=MX+1
4620 POSITION MX,MY:? #6;"!&":POSITION M
X,MY+1:? #6;"#$"
4630 RETURN
4640 LOCATE MX-1,MY,Z:IF Z<>0 THEN POP :
GOTO 4830
4650 LOCATE MX-1,MY+1,Z
4660 IF Z<>0 THEN POP :GOTO 4830
4670 POSITION MX,MY:? #6;"":POSITION MX,
MY+1:? #6;""
4680 MX=MX-1
4690 POSITION MX,MY:? #6;"!&":POSITION M
```

```
X,MY+1:? #6;"#$"
4700 RETURN
4710 LOCATE MX,MY+2,Z:IF Z<>0 THEN POP :
GOTO 4830
4720 LOCATE MX+1,MY+2,Z:IF Z<>0 THEN POP
 :GOTO 4830
4730 POSITION MX,MY:? #6;"":POSITION MX,
MY+1:? #6;""
4740 MY=MY+1
4750 POSITION MX,MY:? #6;"%&":POSITION M
X,MY+1:? #6;"#$"
4760 RETURN
4770 LOCATE MX,MY-1,Z:IF Z<>0 THEN POP :
GOTO 4830
4780 LOCATE MX+1,MY-1,Z:IF Z<>0 THEN POP
 :GOTO 4830
4790 POSITION MX,MY:? #6;"":POSITION MX,
MY+1:? #6;""
4800 MY=MY-1
4810 POSITION MX,MY:? #6;"%&":POSITION M
X,MY+1:? #6;"#$"
4820 RETURN
4830 IF Z<>ASC("m") THEN SOUND 0,90,8,10
:FOR T=1 TO 5:NEXT T:SOUND 0,0,0,0:GOTO
4360
4840 REM COMBAT ROUTINE
4850 STRIKE=INT(RND(0)*100)+1
4860 IF STRIKE>DE THEN SOUND 0,160,6,15:
SOUND 0,0,0,0:WO=INT(WO+(STRIKE-DE)/4)
4870 GOSUB 3010
4880 IF WO>=100 THEN GOTO 6250
4890 STRIKE=INT(RND(0)*100)+1
4900 IF STRIKE<HI THEN SOUND 0,120,6,15:
SOUND 0,0,0,0:MST=INT(MST-(HI-STRIKE)/2)
4910 IF MST<=0 AND MONS(ROOM)=16 THEN EY
E=1
4920 IF MST<=0 THEN SOUND 0,14,10,15:SOU
ND 0,0,0,0:MONS(ROOM)=0:LI=0:POSITION MX
```

```
,MY:? #6;""
4930 IF LI=0 THEN POSITION MX,MY+1:? #6;
"":MK=MK+1
4940 GOTO 4360
X 4950 REM COLLISIONS
4960 IF Z>32 AND Z<39 THEN POP :GOTO 483
0:REM  CLEAR GOSUB STACK THEN GOTO COLLI
SION WITH MONSTER
X 4970 REM HEALING WELL(DRIES UP AFTER USE
)
4980 IF Z=ASC("'") THEN WO=0:HW=0:FOR I=
110 TO 90 STEP -1:SOUND 0,I,10,10:NEXT I
:SOUND 0,0,0,0
X 4990 REM ERASE WELL
5000 IF Z=ASC("'") THEN POSITION HX,HY:?
 #6;"":RETURN
5010 IF Z=ASC("Q") THEN SOUND 0,120,8,15
:FOR DEL=1 TO 5:NEXT DEL:SOUND 0,0,0,0:R
EM COLLISION WITH WALL
5020 IF Z=ASC("Q") THEN ST=ST-1:IF ST<=0
 THEN GOTO 6250
5030 IF Z=ASC("Q") THEN GOSUB 3010:RETUR
N
5040 IF Z=ASC("@") AND EYE=0 THEN GOTO 6
250
5050 IF Z=ASC("@") AND EYE=1 THEN GOTO 6
380:REM COMPLETED MISSION
5060 IF Z=ASC("X") AND LEV<>3 THEN LEV=L
EV+1:ROOM=ROOM+100:GOSUB 3010:DONE=0:GOS
UB 3240:RETURN
5070 IF Z=ASC("X") AND LEV=3 THEN LEV=LE
V-1:ROOM=ROOM-100:GOSUB 3010:DONE=0:GOSU
B 3240:RETURN
5080 IF Z=ASC("<") THEN GO=GO+TRES(ROOM)
:POSITION 10,4:? #6;"":TRES(ROOM)=0:GOSU
B 3010:RETURN
5090 IF Z=ASC("(") THEN GOSUB 5110:RETUR
N
```

```
5100 IF Z=ASC(")") THEN GOSUB 5200:RETUR
N
5110 FOR QQ=1 TO 5
5120 IF HOLD(QQ)<1 THEN GOTO 5150
5130 NEXT QQ
5140 RETURN
5150 HOLD(QQ)=WEAPON(ROOM)
5160 IF WEAPON(ROOM)=18 THEN WHAND=1
5170 WEAPON(ROOM)=0
5180 POSITION WX,WY:? #6;""
5190 RETURN
5200 FOR QQ=1 TO 5
5210 IF HOLD(QQ)<1 THEN GOTO 5240
5220 NEXT QQ
5230 RETURN
5240 HOLD(QQ)=DROP(ROOM)
5250 IF DROP(ROOM)=18 THEN WHAND=1
5260 DROP(ROOM)=0
5270 POSITION DX,DY:? #6;""
5280 RETURN
5290 REM ENTERING NEW ROOM
5300 IF X+DIRX=0 THEN ROOM=ROOM-1:X=18
5310 IF X+DIRX=19 THEN ROOM=ROOM+1:X=1
5320 IF Y+DIRY=9 THEN ROOM=ROOM+10:Y=1
5330 IF Y+DIRY=0 THEN ROOM=ROOM-10:Y=7
5340 DONE=0:LI=0
5350 GOTO 4330
5360 DATA 5,2,5,72,112,112,120,0
5370 DATA 0,7,139,139,139,147,227,0
5380 DATA 9,6,6,9,16,32,64,128
5390 DATA 9,6,6,9,16,32,64,128
5400 DATA 14,10,2,4,8,16,32,64
5410 DATA 14,10,2,4,8,16,32,64
5420 DATA 24,102,165,153,153,165,102,24
5430 DATA 60,231,129,231,36,36,36,60
5440 DATA 24,126,255,255,255,255,126,24
5450 DATA 55,111,222,254,248,136,144,224
5460 DATA 24,36,36,66,66,66,102,60
```

```
5470 DATA 3,7,14,60,120,248,240,240
5480 DATA 0,24,24,24,60,126,126,60
5490 DATA 124,78,39,19,73,37,19,8
5500 DATA 60,126,66,90,90,66,126,60
5510 DATA 0,24,24,24,36,66,66,60
5520 DATA 0,24,24,24,60,126,126,60
5530 DATA 0,14,240,255,240,14,16,8
5540 DATA 255,129,129,231,36,36,36,60
5550 DATA 0,231,132,132,231,129,129,135
5560 DATA 0,247,148,148,247,129,129,135
5570 DATA 0,143,137,137,142,137,137,239
5580 DATA 0,231,132,132,231,33,33,231
5590 DATA 0,135,132,132,135,129,129,247
5600 DATA 62,126,96,124,62,6,126,124
5610 DATA 0,0,1,1,1,0,3,4
5620 DATA 0,0,192,192,192,128,224,144
5630 DATA 8,19,1,1,1,1,1,3
5640 DATA 136,228,64,64,64,64,64,96
5650 DATA 0,0,1,3,3,3,1,3
5660 DATA 0,0,128,192,192,192,128,192
5670 DATA 5,9,17,1,2,4,4,12
5680 DATA 160,144,136,128,64,32,32,48
5690 DATA 2,3,66,99,115,115,121,127
5700 DATA 64,194,70,199,206,207,158,254
5710 DATA 5,9,17,1,2,4,4,12
5720 DATA 160,144,136,128,64,32,32,48
5730 DATA 0,0,1,3,3,3,1,3
5740 DATA 0,0,128,192,192,192,128,192
5750 DATA 5,9,17,1,2,4,4,12
5760 DATA 160,144,136,128,64,32,32,48
5770 DATA 45,163,151,79,93,63,62,60
5780 DATA 16,160,236,233,177,250,124,60
5790 DATA 158,191,112,50,100,120,96,224
5800 DATA 116,249,237,206,12,38,30,7
5810 DATA 0,0,1,3,3,3,1,3
5820 DATA 0,0,128,192,192,192,128,192
5830 DATA 5,9,17,1,2,4,4,12
5840 DATA 160,144,136,128,64,32,32,48
```

```
 5850 DATA 1,3,7,13,7,3,1,3
 5860 DATA 128,192,224,176,224,192,128,19
 2
×5870 DATA 5,9,17,1,2,4,4,12
×5880 DATA 160,144,136,128,64,32,32,48
 5890 DATA 0,1,3,199,237,111,110,60
 5900 DATA 0,128,192,227,179,246,118,60
 5910 DATA 15,31,63,124,124,112,96,64
 5920 DATA 240,224,128,128,0,0,0,0
 5930 DATA 0,0,1,3,142,207,225,241
 5940 DATA 0,0,128,192,193,193,195,199
 5950 DATA 249,253,255,253,249,241,194,15
 6
 5960 DATA 207,223,255,223,207,135,3,1
 5970 DATA 1,3,15,1,7,9,17,17
 5980 DATA 0,128,128,128,224,144,136,136
 5990 DATA 17,2,4,8,8,8,24,0
 6000 DATA 136,64,32,16,16,16,24,0
 6010 DATA 0,0,1,3,2,3,1,3
 6020 DATA 0,0,128,192,64,192,128,192
×6030 DATA 5,9,17,1,2,4,4,12
×6040 DATA 160,144,136,128,64,32,32,48
 6050 DATA 0,3,7,13,63,126,204,207
 6060 DATA 0,192,224,176,252,126,51,243
 6070 DATA 15,12,24,48,48,48,112,0
 6080 DATA 240,48,24,12,12,12,14,0
 6090 DATA 0,0,65,99,115,115,121,127
 6100 DATA 0,2,134,199,206,207,158,254
 6110 DATA 123,117,109,125,122,116,100,76
 6120 DATA 222,174,182,190,94,38,34,48
 6130 DATA 0,1,1,1,3,7,14,31
 6140 DATA 0,128,224,240,128,0,0,248
 6150 DATA 31,48,63,63,96,127,127,0
 6160 DATA 248,0,252,252,0,254,254,0
 6170 DATA 2,2,2,2,3,3,1,3
 6180 DATA 64,64,64,64,192,192,128,192
×6190 DATA 5,9,17,1,2,4,4,12
×6200 DATA 160,144,136,128,64,32,32,48
```

```
X 6210 DATA 0,0,1,3,3,3,1,3
X 6220 DATA 0,0,128,192,192,192,128,192
X 6230 DATA 5,9,17,1,2,4,4,12
X 6240 DATA 160,144,136,128,64,32,32,48
  6250 FOR QQ=15 TO 1 STEP 0.2
  6260 SETCOLOR 0,3,15:SETCOLOR 0,0,0
  6270 SOUND 0,120,8,QQ
X 6280 NEXT QQ
  6290 SOUND 0,0,0,0
  6300 GRAPHICS 2
  6310 POSITION 0,1:? #6;"YOU APPEAR TO BE
  "
  6320 POSITION 0,2:? #6;"DEAD!"
  6330 POSITION 0,3:? #6;"monsters killed:
  ";MK
  6340 POSITION 0,4:? #6;"gold collected:"
  ;GO
  6350 POSITION 0,8:? #6;"ANOTHER GO?":INP
  UT DUM$
  6360 IF DUM$(1,1)="Y" THEN RUN
  6370 GOTO 6350
  6380 FOR QQ=15 TO 1 STEP 0.2
  6390 SETCOLOR 0,0,15:SETCOLOR 0,0,0
  6400 SOUND 0,14,10,QQ
X 6410 NEXT QQ
  6420 SOUND 0,0,0,0
  6430 GRAPHICS 2
  6440 POSITION 0,0:? "THIS ADVENTURE IS "
  6450 POSITION 0,1:? "OVER."
  6460 POSITION 0,2:? "CONGRATULATIONS"
  6470 END
```

# Index

Other titles from Sunshine

## SPECTRUM BOOKS

**Spectrum Adventures**
A guide to playing and writing adventures
Tony Bridge & Roy Carnell — **£5.95**
ISBN 0 946408 07 6

**ZX Spectrum Astronomy**
Maurice Gavin — **£6.95**
ISBN 0 946408 24 6

**Spectrum Machine Code Applications**
David Laine — **£6.95**
ISBN 0 946408 17 3

**The Working Spectrum**
David Lawrence — **£5.95**
ISBN 0 946408 00 9

**Master your ZX Microdrive**
Andrew Pennell — **£6.95**
ISBN 0 946408 19 X

## COMMODORE 64 BOOKS

**Graphic Art for the Commodore 64**
Boris Allan — **£5.95**
ISBN 0 946408 15 7

**Commodore 64 Adventures**
Mike Grace — **£5.95**
ISBN 0 946408 11 4

**Business Applications for the Commodore 64**
James Hall — **£5.95**
ISBN 0 946408 12 2

**Mathematics on the Commodore 64**
Czes Kosniowski — **£5.95**
ISBN 0 946408 14 9

**Advanced Programming Techniques on the Commodore 64**
David Lawrence — **£5.95**
ISBN 0 946408 23 8

**The Working Commodore 64**
David Lawrence **£5.95**
ISBN 0 946408 02 5

**Commodore 64 Machine Code Master**
David Lawrence & Mark England **£6.95**
ISBN 0 946408 05 X

## ELECTRON BOOKS

**Programming for Education on the Electron Computer**
John Scriven & Patrick Hall **£5.95**
ISBN 0 946408 21 1

## BBC COMPUTER BOOKS

**Functional Forth for the BBC computer**
Boris Allan **£5.95**
ISBN 0 946408 04 1

**Graphic Art for the BBC computer**
Boris Allan **£5.95**
ISBN 0 946408 08 4

**DIY Robotics and Sensors for the BBC computer**
John Billingsley **£6.95**
ISBN 0 946408 13 0

**Programming for Education on the BBC computer**
John Scriven & Patrick Hall **£5.95**
ISBN 0 946408 10 6

**Making Music on the BBC Computer**
Ian Waugh **£5.95**
ISBN 0 946408 26 2

## DRAGON BOOKS

**Advanced Sound & Graphics for the Dragon**
Keith & Steven Brain **£5.95**
ISBN 0 946408 06 8

**Dragon 32 Games Master**
Keith & Steven Brain **£5.95**
ISBN 0 946408 03 3

**The Working Dragon**
David Lawrence **£5.95**
ISBN 0 946408 01 7

**The Dragon Trainer**
A handbook for beginners
Brian Lloyd **£5.95**
ISBN 0 946408 09 2

## ATARI BOOKS

**Writing Strategy Games on your Atari Computer**
John White **£5.95**
ISBN 0 946408 22 X

Sunshine also publishes

**POPULAR COMPUTING WEEKLY**

The first weekly magazine for home computer users. Each copy contains Top 10 charts of the best-selling software and books and up-to-the-minute details of the latest games. Other features in the magazine include regular hardware and software reviews, programming hints, computer swap, adventure corner and pages of listings for the Spectrum, Dragon, BBC, VIC 20 and 64, ZX 81 and other popular micros. Only 35p a week, a year's subscription costs £19.95 (£9.98 for six months) in the UK and £37.40 (£18.70 for six months) overseas.

**DRAGON USER**

The monthly magazine for all users of Dragon microcomputers. Each issue contains reviews of software and peripherals, programming advice for beginners and advanced users, program listings, a technical advisory service and all the latest news related to the Dragon. A year's subscription (12 issues) costs £10.00 in the UK and £16.00 overseas.

**MICRO ADVENTURER**

The monthly magazine for everyone interested in Adventure games, war gaming and simulation/role playing games. Includes reviews of all the latest software, lists of all the software available and programming advice. A year's subscription (12 issues) costs £10 in the UK and £16 overseas.

**COMMODORE HORIZONS**

The monthly magazine for all users of Commodore computers. Each issue contains reviews of software and peripherals, programming advice for beginners and advanced users, program listings, a technical advisory service and all the latest news. A year's subscription costs £10 in the UK and £16 overseas.

For further information contact:
Sunshine
12–13 Little Newport Street
London WC2R 3LD
01-437 4343

**Adventuring! You and your trusty Atari alone in a dark cave, or deep in the dungeons of a terrifying castle. Alone, that is, except for . . . what's that? The drip of water on stone, or the approaching footsteps of a minotaur, jaws slavering for your blood?**

**Atari Adventures is in two parts. The first part takes a look at the beginnings of Adventure — from the original, text only version, in which the player has to solve many puzzles on the way to finding the treasure, to the pure arcade form, in which bloodthirsty, fearsome monsters have to be fought and vanquished.**

**The second part presents a part-graphic, part-text adventure The Eye of the Star Warrior. Each line is fully discussed and many of the routines can be used in your own programs.**

Tony Bridge writes a regular column in Popular Computing Weekly, and in Britain's only adventure magazine, Micro Adventurer. In his other life, he sits in a dark underground cavern, making records with many of today's major recording artists.


ISBN 0 946408 18 1

**£5.95 net**