

We're off on the road to

PART 1

Why not write your own adventure game? This new series will show how easy it is.

Adventure!



Writing an adventure is fun. And not only for you. It can give your friends lots of pleasure when they play it later.

And there's another good thing about writing adventures. You don't have to write a completely new program each time you want to create a new game.

Over the next few months we're going to show you how to write a program that will work for lots of different games.

It's called an *Adventure Manager*.



We'll be building it up section by section. In programming terms each of these is called a *routine*.

There will be lines in the program that will make the game jump to these routines when they are needed. These are known as *calls* to the routine.

That's enough of the jargon for now. Let's start building the game!

When you play an adventure your computer needs to understand the words the players type in. Each game will use different words. The program must be able to deal with all of them.



So the first thing we have to do is give our computer a list of words we want it to understand. Type in Lines 4999 to 5500.

This is a simple word list. It includes direction words like

NORTH and SOUTH - all your adventures will need these.

The DATA at line 5500 looks different. It's called a *rogue value* and it marks the end of the list.

The program will be written so that your computer knows it has reached the end of the list when it comes across a star (*). That way you can add more words without having to tell the program each time you do so.



This is an example of an important rule of programming: The more work you do when you start, the less you'll have to do later.

Now type in Lines 999 to 2000. These set up lots of things your program will need later. It's called an *initialisation routine*.

Next you need the line that makes the program jump to the routine - the *call*. That's line 10. Type it in now.

Before you RUN the program you should also type in Line 990. That stops the program when all the calls to the routines have been carried out.

Now you're ready to SAVE and RUN the program. Try it.

Nothing will seem to happen yet. What you have done is give your computer a list of words, each with its own space number.

You'll notice that in the DATA we've only used the first four letters of each word. This is usually

enough for the computer to recognise what the word is. But make sure you don't confuse your machine by using similar words like LIST and LISTEN!

All the words are directions. Look at Line 5010. You can see that EAST and E have both been given the same number - 2. This means the computer will treat each of them as the same word.

So it doesn't matter whether a player types in EAST or E. The computer will look up the figure 2 in its memory and take him in the same direction.



You play an adventure game by typing in instructions or directions. We'll now add some lines that print *What next?* This invites the player to tell the computer what he wants to do.

Type Lines 2009 to 2499. You also need the call to them, so add Line 20.

You'll want the program to show you the word numbers it's found, so add Line 100. *We'll take it out when we add the routines that use descriptions instead of the room numbers.*

SAVE and RUN the program again. It will now sit and wait for you to type something in. Try typing anything.



If you type a word that's not in the data two zeros will be printed. If you type something it understands the word or words will be changed into numbers. Try typing North E. See what happens?

All this may seem a complicated way of recognising a few words. You may think you could write a five line

What happens when you type something in?

When you answer the *What next?* question your reply is stored in Y\$. When you've RUN the program try entering: PRINT Y\$

You'll find your answer stored there. Now get your computer to check if any of the words are in your list.

Line 30 calls Lines 2500 to 2699 and they do this job.

The REMs in some of the lines are there to help you

understand what is going on. But don't worry if you don't quite get the hang of how it works. It will all come clear to you shortly.

REMark

REMs are put in programs to make them more understandable to humans. Your computer ignores them completely!

In this program some of the REMs are at the end of lines. Like this:

```
10 GOSUB 1000:REM SET UP START
```

You don't need to type in :REM or the following words.

Other REMs are alone on a line. For example:

```
999 REM SORT THINGS OUT BEFORE YOU START
```

These lines can be missed out entirely. But it's best not to. They could help you find things later.

program to do the same job.

But our program is very friendly. Any of the following will produce the same result:

```
N
NORTH
GO NORTH
RUN NORTH
PLEASE MAY I GO TO THE NORTH
```

Now you have a program that will look for ANY word you put in the data list. Try adding some of your own.

Just add a line between 5050 and 5500. Such as:

```
5060 DATA 10,HELLO
```

Look what happens when you run the program and type HELLO FROG. You will get:

```
10 0
```

That's because Hello has been given the number 10 and FROG is not in the list. It gets a 0.

Your program can now recognise direction words. But it's not an adventure yet. Your players will need somewhere to go - and exciting things to do.

Next month we'll add the lines that put places in your program. Then you'll be well on the way to seeing your own adventure come to life.

A CHALLENGE

So far your players will only be able to type what they want in capital letters.

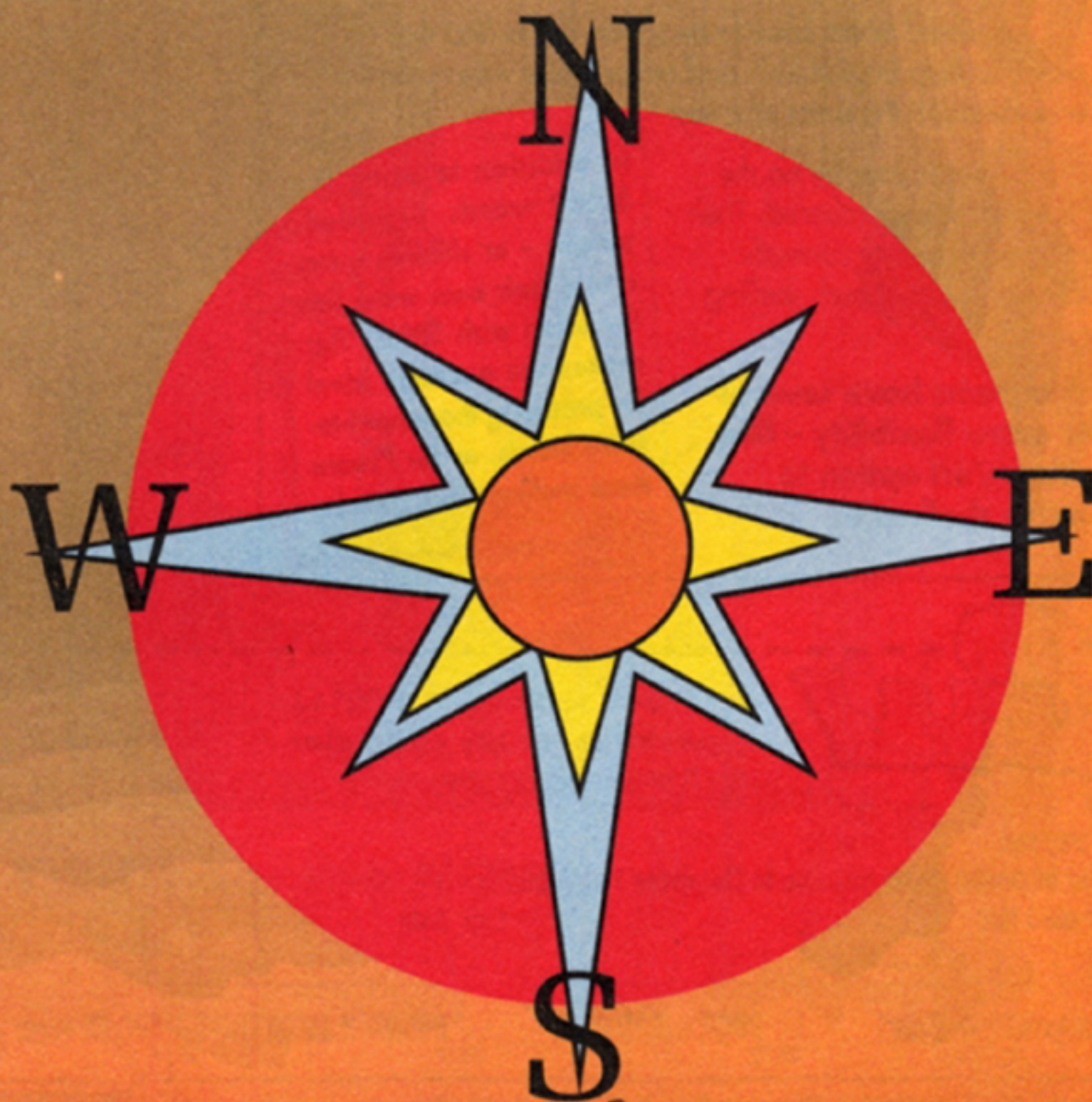
So while you wait for next month's *Let's Compute!* here's a challenge. See if you can make the program work for small letters as well.

IS THIS YOUR COMPUTER?

This program works on a BBC, Archimedes, Electron, CPC, ST(Stos), Amiga and PC(GW-Basic). It will not work on a C64/128 or Spectrum.

```
10 GOSUB 1000:REM SET UP START
20 GOSUB 2010:REM DISPLAY
30 GOSUB 2510:REM GET WORD NUMBERS
100 PRINT"THE WORD NUMBERS WERE ";W(1);
" ";W(2)
990 STOP
998 REM
999 REM SORT THINGS OUT BEFORE YOU STA
RT
1000 LET W=0:REM THIS WILL BE THE NUMB
ER OF WORDS
1005 DIM W(2):REM WORD NUMBERS FOUND G
O HERE
1010 RESTORE 5000
1020 READ X,XS:REM THESE VARIABLES ARE
JUST FOR COUNTING
1030 IF X>0 THEN LET W=W+1:GOTO 1020:R
EM NOT THE END OF THE LIST?
1040 DIM WRDNUM(W):DIM WRDS(W):REM SET
UP ARRAYS TO PUT WORDS IN
1045 RESTORE 5000:REM GO BACK TO THE S
TART OF THE LIST
1050 FOR X=1 TO W:READ WRDNUM(X),WRDS(
X):NEXT X
2000 RETURN:REM-FINISHED SORTING THING
S OUT
2005 REM
2009 REM DISPLAY THINGS
2010 REM BITS GO IN HERE LATER
2200 PRINT "What next ";:INPUT YS:LET
YS=YS+" "
2499 RETURN
2500 REM GET WORD NUMBERS
2510 LET W(1)=0:LET W(2)=0:LET P=0:LET
WHICH=1:REM THE WORD NUMBERS GO IN HERE
, P FOR POINTER, WHICH IS FIRST OR SECON
```

```
D WORD FOUND
2520 LET P=P+1:IF P>LEN(YS) THEN RETUR
N:REM HAVE WE FINISHED?
2530 IF MID$(YS,P,1)=" " THEN GOTO 252
0:REM FIND THE NEXT WORD
2540 LET WS="":REM NO WORD YET
2550 IF MID$(YS,P,1)=" " THEN GOTO 260
0:REM END OF THE WORD?
2560 LET WS=WS+MID$(YS,P,1):IF LEN(WS)
=4 THEN GOTO 2600:REM 4 LETTERS YET?
2570 LET P=P+1:GOTO 2550:REM GET THE N
EXT LETTER
2599 REM NOW SEE IF THE WORD MATCHES
2600 FOR X = 1 TO W
2610 IF WRDS(X)=WS THEN W(WHICH)=WRDNU
M(X):LET WHICH=WHICH+1:REM FOUND MATCH -
SAVE IT
2620 NEXT X
2630 IF WHICH=3 THEN RETURN:REM STOP I
F YOU'VE GOT 2 WORDS
2640 IF MID$(YS,P,1)<>" " THEN LET P=P
+1:GOTO 2640:REM IGNORE THE REST OF THAT
WORD
2650 GOTO 2520:REM GET THE NEXT WORD
2699 RETURN
4900 REM
4999 REM START DATA HERE ----- WORDS
FIRST
5000 DATA 1,"NORT",1,"N"
5010 DATA 2,"EAST",2,"E"
5020 DATA 3,"SOUT",3,"S"
5030 DATA 4,"WEST",4,"W"
5040 DATA 5,"UP",5,"U"
5050 DATA 6,"DOWN",6,"D"
5500 DATA 0,"X"
```



IS THIS YOUR COMPUTER?

This program works on a BBC, Archimedes, Electron, CPC, ST(Stos), Amiga and PC(GW-Basic). It will not work on a C64/128 or Spectrum.

```

35 IF W(1)>=1 AND W(1)<=10 GOSUB 2710
:GOTO20:REM FOR MORE THAN 10 ROOMS CHANG
E THE 10 IN THIS LINE
990 GOTO 20
1060 ROOM=1:Y$="start the adventure":ME
SS1$="":MESS2$="":MESS3$=""
2010 RESTORE 5520:REM GO TO THE START O
F THE ROOM LIST
2020 FOR C=1 TO ROOM:READ DIR$,DESC$,CO
DESS:NEXT C:REM GET INFO FOR THAT ROOM
2025 CLS:PRINT"You are ";DESC$;".":REM
PRINT WHERE YOU ARE
2030 D$="":IF INSTR(DIR$,"N")>0 D$=D$+"
North "
2031 IF INSTR(DIR$,"E")>0 D$=D$+"East "
2032 IF INSTR(DIR$,"S")>0 D$=D$+"South
"
2033 IF INSTR(DIR$,"W")>0 D$=D$+"West "
2034 IF INSTR(DIR$,"U")>0 D$=D$+"Up "
2035 IF INSTR(DIR$,"D")>0 D$=D$+"Down":
REM D$ CONTAINS THE DIRECTION YOU CAN GO
2190 IF D$>"" THEN PRINT:PRINT"You can
go ";D$
2191 PRINT"*****"
*****:REM 39 STARS
2192 PRINT:PRINT "You wanted to ";Y$:PR
INT:REM REMIND THE PLAYER WANT THEY WANT
ED TO DO
2193 PRINT MESS1$
2194 PRINT MESS2$
2195 PRINT MESS3$:PRINT:MESS1$="":MESS2
$="":MESS3$="":REM PRINT ANY MESSAGES AN
D CLEAR THEM
2700 REM TRY TO MOVE TO A NEW ROOM
2710 P=1:REM POINTER IN ROOM CODE
2720 IF VAL(MID$(CODESS$,P,2))=W(1) THEN
ROOM=VAL(MID$(CODESS$,P+2,2)):RETURN
2730 IF VAL(MID$(CODESS$,P,2))=99 THEN M
ESS1$="You can't go that way!":RETURN
2740 P=P+4:GOTO2720
5510 REM ROOMS START HERE
5520 DATA S,in the control room,030399
5530 DATA E,in the weapons room,020399
5540 DATA NSEW,in a corridor,0101020403
05040299
5550 DATA W,in a store room,040399
5560 DATA NS,in a corridor,0103030799
5570 DATA ES,in the recreation room,020
7031099
5580 DATA NSEW,in a corridor,0105020803
11040699
5590 DATA ESW,in the air lock,020903120
40799
5600 DATA W,in the shuttle,040899
5610 DATA NES,in the sleeping quarters,
01060211031399
5620 DATA NSEW,in a corridor,0107021203
14041099
5630 DATA NSW,standing by a crate in th
e hold,01080315041199
5640 DATA NEW,crawling through an air du
ct,01100214042099
5650 DATA NSEW,in a corridor,0111021503
16041399
5660 DATA NW,in the hold,0112041499
5670 DATA N,in the engine room,011499
5680 DATA NEWS,crawling through an air d
uct,011802180319041999
5690 DATA NEWS,crawling through an air d
uct,011702130320041799
5700 DATA NEWS,crawling through an air d
uct,011702200320041799
5710 DATA NEWS,crawling through an air d
uct,011802130319041999
5900 DATA X,X,X

```

*Now your great
adventure really
starts to take
shape as you
design the maze*



Last month we showed you how to start writing your own Adventure Manager.

The program we gave you recognises words the player types in. It lets you put in any words you want your computer to understand.

The first six words we used were direction words. Now that your computer can recognise these it needs places to move to.

First you need to draw a map. Ours is for a spaceship. But you can draw your own for anywhere you want - your house, your school or your town for example.

Try ours first, then have a go with your own.



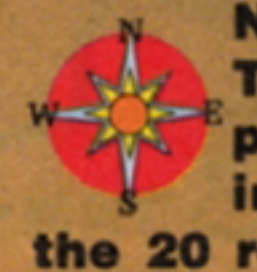
We'll call each place on the map a room. Even if it's really a cabin, a hold or a corridor. That way we

can talk about, for instance, *Room 8*. You can even talk about your garden as being a room on your own map.

Once you've drawn your map you're ready to begin working on your computer.

Start by loading last month's program. You're going to add more lines to it.

Type 100 and press Return. This removes Line 100 - remember it was just there to let you know your program was working.



Now look at the map. This is what your computer needs to know. The instructions for each of

the 20 rooms are in Lines 5510 to 5900. Add these to your program.

The panel on the right explains how these DATA lines are made up. Just follow the instructions there to make your own DATA lines.

Now add Line 1060. This tells the computer which room to

Mapp Ad

PART 2

Always draw your map first. Then work out the codes on paper. That way you will find it much easier to sort out any mistakes.

start in. It also gives it the message that needs to be printed at the start.



Lines 2010 and 2020 find the computer's description of the room from the DATA. Then Lines 2025 to 2195 display what's been read. There's also a bit that can send messages to the player. You will need these later.

Type these lines in and run the program. You'll be in the control room with this message showing: *You can go South*

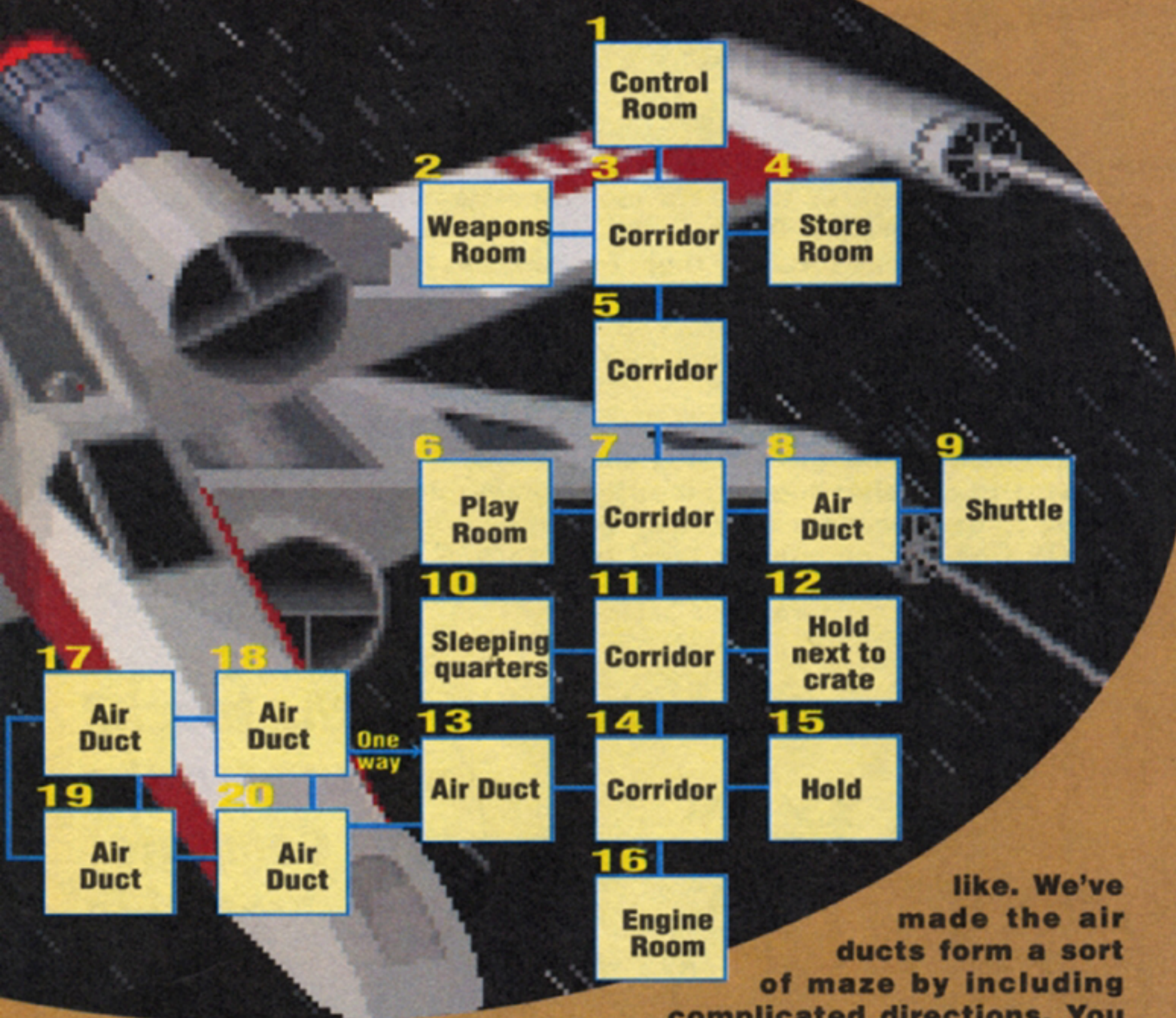


Now type in Lines 2700 to 2740. They check the first word typed in by the player to find which new room to move to.

If your computer doesn't know the word you won't be able to go that way.

Now we must make the computer use this routine. Line 35 checks to see if the first word found is a direction word. If it is

ing the road to venture!



it goes to the movement sub-routine.

Finally, replace the old line 990 with:

```
990 GOTO 20
```

This makes the whole program go in a complete loop so you can keep playing.

Now **SAVE** your complete program and try **RUN**ning it.



You will be able to move around the spaceship in the map. We haven't used any **UP** or **DOWN** to keep things simple. But they are in the program ready for you to use if you write your own version.

The connections between rooms can be as difficult as you

like. We've made the air ducts form a sort of maze by including complicated directions. You can make a few rooms seem like a lot by doing this sort of trick.

You can have one way paths like the one we've put between rooms 18 and 13.

You can have magic tunnels that join one end of your map to the other. You could make our spaceship into a circular space station by adding connections between rooms 1 and 16.

Now try using your own set of rooms. Start with something simple like your house or your school.

Then move on to making up your own adventure landscape.

Next month we'll look at how you can add things like space suits to your adventure.

What's in the DATA lines?

Each room needs three bits of data. As an example, look at Line 5590. This describes room number 8 – the Air duct. The program knows it is room 8 because it is the eighth room DATA line.

The first bit of data, ESW, shows the directions you can move in. The second bit, in the air lock, is the description of the room. Your computer will add *You are* to each description.

The next group of numbers tell the program which words send you where. Word 02 will move us to room 09, 03 to room 12 and 04 to room 07.

Each number must be two figures. So, for example, the number 2 must be written as 02 in the DATA.

The 99 at the end tells the program that's the end of the list.

TRY THIS!

You could make a maze seem to go on for ever by connecting a room to itself. Try changing the codes for room 19 to:

```
01190219031904051906191999
```

Also alter the directions to **NSEWUD** and see what happens when you enter that room.

ANSWER TO LAST MONTH'S CHALLENGE

There are many ways to make your computer understand capital or lower case letters. Here's a way that will work on all computers:

```
2201 LET YYS="":FOR ZZ=1 TO LEN(YS)
2202 LET LLS=MIDS(YS,ZZ,1):LET
LL=ASC(LLS)
2203 IF LL>91 THEN LET LLS=CHR$(LL-32)
2204 LET YYS=YYS+LLS
2205 NEXT ZZ
2206 LET YS=YYS
```

ESC**EDUCATIONAL SOFTWARE CLUB**

PHONE 0702 600557 • FAX 0702 613747

32A SOUTHCHURCH ROAD • SOUTHBOROUGH • ESSEX SS1 2ND

We Have Over 70 Top
Class Educational
Programs For The
Amiga, IBM/PC &
Atari ST.

All Ages Covered
From 3 To Adult.

Contact Us Now For
Your **Free Catalogue.**



Designasaurus: Print, Create &
Survive like the dinosaurs in their
own ecosystem.
Ages 6+ Amiga or IBM/PC £29.95.

SPECIAL OFFERS

EARLY LEARNING MATHS: A
really fun collection of 12 maths
programs to quiz your kids. Ideal
for 4 to 9 year olds.
Atari ST/Amiga £14.95

LIZZY'S SPELLICOPTER: Fly
Lizzy's Helicopter around and
spell the words presented. Lots
of speech and graphics make
this the best Amiga Spelling
Program. Ages from 4 to 9.
Amiga Only £14.95

FUNTIME 1: A collection of 4
simple programs. Excellent
graphic presentation for the
Under 6's.
Amiga/Atari ST £3.00

FUNTIME 2: Another set of 4
simple yet brilliant programs.
Excellent graphic presentation.
Great for the Under 6's.
Amiga/Atari ST £3.00

BOX of 10 5¼"
Branded Floppy Disks
Only £4.50 Per Box

THIS IS NOT A MISPRINT

**ONLY BY PURCHASING OVER 3 MILLION DISKS
A YEAR CAN WE OFFER THE FOLLOWING
SENSATIONAL PRICES...**

Every diskette supplied by
Dial a Disc is certified and
tested 100% error free.

200 DSDD 3.5"
ONLY £60.00

3.5" DSDD 135TPI

50	100	150	200	400	500
£22	£33	£48	£60	£115	£135

3.5" HD
ONLY
60p each

5.25" DSDD
ONLY
23p each

5.25" DSHD
ONLY
40p each

DISK STORAGE BOXES

3.5" 40 capacity.....	£2.95
3.5" 80 capacity.....	£3.45
3.5" 100 capacity.....	£3.95
only if bought with disks	

OUR PROMISE IS SIMPLE

100% SATISFACTION OR MONEY BACK

DIAL A DISC

203 Southborough Lane, Bromley, Kent, BR2 8AR.
081-467 0131

ALL prices include VAT & delivery. All offers subject to availability. E/OE.

The object

Adv



Up to now we've created an adventure land-
scape that can be moved around at will.
Now let's move on to the next important part
of your adventure manager - placing the
objects.

Most adventures use objects. Remember the ones in
The Golden Crown in February's issue of *Let's Compute*?

There you had objects like a large bucket and the
cloak. You could pick them up and use them as part of
the game.

Now we are going to put objects into OUR adventure.
You can see them in the DATA in Lines 6000 to 6300. Add
these to your program.

Note that the last DATA line contains a *rogue value*. It
works like the one in the word list. This was explained in
the March issue.

Remember, using this system means you can easily
add things to your list. You don't have to tell the program
how many things there are in it.



Look at the DATA lines. You can see that
each object has a number in front of it. This
is the number of the room the object will
start in.

Some of the objects have zeros in front of
them. This just means they will not appear in
any room at the start. You can think of room zero as a
sort of store room for your objects when you're not using
them.

Notice that objects are not always things you can pick
up and use. They can do other things. For instance, the
red light in the air lock will eventually be changed to the
green light.

The objects and room numbers are in DATA lines. Your
computer now needs to READ them.

This is done in the same way as we did with the
words. Type in lines 1080 to 1140.



These are now part of the setting up rou-
tine. Every time you run the program the
objects will be put in the correct rooms.

The array **OBJECT\$** contains the descrip-
tions of the objects. **OBJROOM** holds the
room numbers. So **OBJECT\$(1)** contains laser rifle and
OBJROOM(1) contains 2, the room the laser starts in.

Like the room numbers, each object is numbered by its
place in the list. For example, the laser is number 1, the
space suit number 4 and the fixed cable number 14.

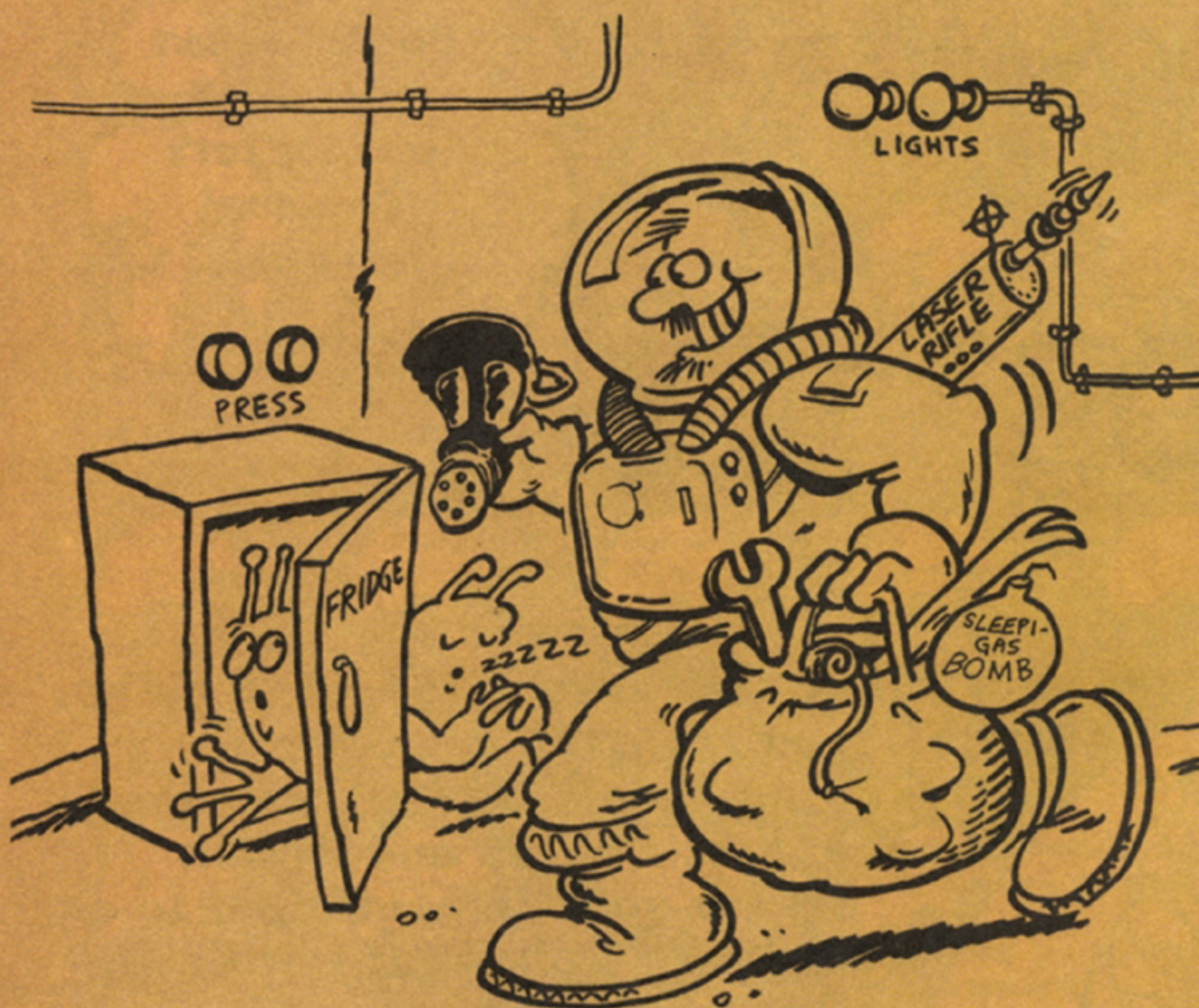
Next we want to display the objects as part of the

ect of an

Add a new dimension to
your adventure by placing
things to pick up

PART 3

enture!



IS YOUR COMPUTER HERE?

This program works on a BBC, Archimedes, Electron, CPC, Atari ST(Stos), Amiga and PC(GW-Basic). It will not work on a C64/128 or Spectrum.

```

35 IF W(1)>=1 and W(1)<=10 GO SUB 271
0:FOR MORE THAN 10 MOVEMENT WORDS (NOT ROOMS AS STATED IN APRIL) CHANGE THE 10 IN THIS LINE.
1070 REM NOW SORT OUT THE OBJECTS - FIRST COUNT THEM
1080 RESTORE 6000
1085 0=0:REM THIS WILL BE THE NUMBER OF OBJECTS
1090 READX,X$:REM THESE VARIABLES ARE JUST FOR COUNTING
1100 IF X>-1 0=0+1:GOTO 1090:REM NOT THE END OF THE LIST?
1110 DIM OBJROOM(0):DIM OBJECTS(0)
1120 REM NOW PUT THE OBJECTS INTO AN ARRAY
1130 RESTORE 6000:REM BACK TO THE START OF THE OBJECT LIST
1140 FOR X=1 TO 0:READ OBJROOM(X),OBJECTS(X):NEXT
2040 REM NOW TELL US WHAT YOU CAN SEE
2050 PRINT:PRINT "You can see :";
2060 FOR X=1 TO 0
2070 IF OBJROOM(X)=ROOM PRINTTAB(14);OBJECTS(X)
2080 NEXT X:PRINT
5990 REM OBJECTS START HERE
6000 DATA 2,lazer rifle
6010 DATA 4,sleepi-gas bomb
6020 DATA 1,red button
6030 DATA 1,green button
6040 DATA 0,space suit
6050 DATA 19,spanner
6060 DATA 18,crow-bar
6070 DATA 8,red light
6080 DATA 0,green light
6090 DATA 13,hungry alien
6100 DATA 0,sleeping alien
6110 DATA 12,gas mask
6120 DATA 16, loose cable
6130 DATA 0,fixed cable
6300 DATA -1,X
    
```

room description. Lines 2050 to 2080 do this. So type them in.

The lines search the **OBJROOM** array. As they do they check if any of the room numbers match the one you are in. If they do the description of the object is printed underneath the room description.

Now try **RUN**ning the program. You'll see our objects in your adventure. Why not put in some of your own?

You could put teachers into classrooms, tools into sheds or dragons into caves. Just use your imagination and scatter the map with all sorts of things!

● **At the moment the objects are just things in the rooms. Next month we'll find out how to carry them around. Then they'll really become part of the adventure!**

A CHALLENGE

So far your program displays the objects in a room. This is fine as long as there are some objects there!

If there aren't it prints:

You can see :

Can you alter the display routine yourself? It shouldn't print anything at all if there are no objects in the room. Or perhaps you could make it print:

You can see : nothing interesting

So far we've invented an adventure full of things waiting to be picked up. But what use are they if you can't take and carry them?

The first thing we need to do is get the computer to understand the names of the objects. You do this by adding them to the word list.

Add Lines 5060 to 5490 to your program. Words 11 and 12 give some words which mean TAKE and DROP. You could think of some more to add yourself. But there's a small snag!

You can see that some objects are easy to deal with. The laser rifle becomes word 50 - LASE and RIFE. Remember we are only using the first four letters of each word.

But the gas bomb and gas mask are a problem. If the player types TAKE GAS MASK or TAKE GAS BOMB the word numbers would be the same. The program would not know which object to deal with.

One way round this that we'll use for now is to leave out the word GAS. Now TAKE MASK or TAKE BOMB give different word numbers.

This shows that you have to think very carefully before you start. When you design your adventures look at just what players will try to type.

There will be the same problems with the buttons and lights. But as we can't carry these they will never be in the same room at once so we don't have to worry so much. We'll look at a better way of getting round the problem in a future issue of *Let's Compute!*

Now we need to give the program instructions on how to deal with the word combinations that might be typed. We call these conditions and actions.

If a condition is true then the program will carry out the listed actions. Type in Lines 6399 to 7000. This is the DATA.

Now type in Lines 1150 to 1220. These read the actions and conditions into their own arrays.

They come in pairs. Look at Line 6400. If words 12 and 50 are typed in the program will do action A, try to take object 01 - the laser rifle.

Like the room connection codes we must make sure that all our numbers have two digits. Remember, that's so we can scan through the list easily.

At the moment you can take one of three actions - A, B or C. As you can see in Lines 6400 to 6490, each is followed by a number.

The actions that are there so far are:

- Ann - Try to take object nn
- Bnn - Try to drop object nn
- C - List all the objects carried.

Each list of conditions and actions ends with a #.

Look through the table of conditions and actions and work out which words do what. You can see that the player will not be able to take the alien, the cable, the buttons or the lights.

Now we have to check to see if any of the words match the combinations in the condition table. Type in Lines 2999 to 3100.

They step through the condition table to see if the first word matches. If it doesn't the program goes to the next set of conditions. If the first word matches, it checks the second.

Look at Line 6460. There you'll find is the

Carry things Adventure

condition/action code for INVENTORY. That's the word which means *What am I carrying?* This is word 99. But we don't need another word so we follow it with 00. The program takes this to mean any word. You will find the inventory command very useful later on.

In future issues of *Let's Compute!* we'll be checking other things. For instance we'll look at what room we're in and whether we have a certain object.

At the moment we are only interested in matching the words. Now the program has both words matched it can go to the actions routine.

Type in Lines 3499 to 3590. These carry out the actions. They look at the first letter and use that at Line 3520 to GOSUB to that letter's routine.

So letter A causes a GOSUB to 3610, B to 3630 and C to 3650. You can see that it's very easy to add more letters with different actions to this system.

If you try to take an object the program checks two things. It makes sure it is there and that you aren't already carrying it. Then it lets you take it.

DROPPing first checks to see that you have the object. Room -1 is used for the inventory so carried objects have OBJROOM set to -1.

The program keeps track of where it is in the action with the variable P. At the moment this is not important because there is only one action in each set of codes.

Soon we will have several actions. The taking and dropping routines make sure they move this pointer along past the object numbers ready for the next action. Action C doesn't move the pointer because it has no numbers after it.

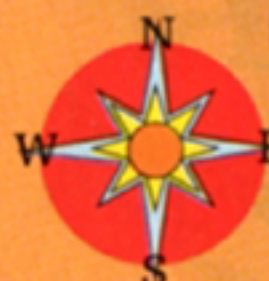
You will see the variable DIS in these routines. This tells the display routine how much to show. For example, the inventory list can be quite long so DIS is set to 2.

This means the list will be displayed but not the room description. Type in Line 1065 to initialise DIS and Lines 2015 and 2196 to put it into the display routine.

Finally, add Line 40. This puts the conditions/actions into the main program. Now you can move around the spaceship taking and dropping the objects we have listed in the condition/action table.

Alter your own adventure so you, too, can take and pick up objects. Your adventure is now really taking shape.

Next time we'll add more complicated conditions and some more actions. Then we will be able to push buttons and fix the cable.



s round your PART 4 nture!

*Here's how to pick
things up as you move
round the maze*

Answer to last month's challenge

Did you stop your computer printing *You can see* when the room is empty? There are lots of ways it can be done.

Here are two. Use one of the sets of line changes below to cure the problem.

The first method stops your computer printing anything at all if nothing is in the room. The second still prints *You can see* but follows it with the words *nothing useful*.

Both ways use what programmers call a flag. We've called ours ZZ. That way it's not likely to have the same name as anything else in the program.

It's set to zero first then changed to one if an object is found in the room. Then the program can print what's needed depending on whether ZZ is zero or 1.

Method 1: Doesn't print anything

```
2050 LET ZZ=0
2070 IF ZZ=1 AND OBJROOM(X)=ROOM THEN P
RINTTAB(14);OBJECT$(X)
2075 IF ZZ=0 AND OBJROOM(X)=ROOM THEN P
RINT "You can see: ";OBJECT$(X):LET ZZ=1
```

Method 2: Prints "You can see : nothing useful"

```
2045 LET ZZ=0
2070 IF OBJROOM(X)=ROOM THEN PRINTTAB(1
4);OBJECT$(X):LET ZZ=1
2080 NEXT X:IF ZZ=0 THEN PRINT "nothin
g useful"
2085 PRINT
```

IS YOUR COMPUTER HERE?

This program works on a BBC, Archimedes, Electron, CPC, Atari ST(Stos), Amiga and PC(GW-Basic). It will not work on a C64/128 or Spectrum.

```
40 GOSUB3000
1065 DIS=1:REM SET INFORMATION DISPLAY
1150 A=0:REM THIS WILL BE THE NUMBER OF
ACTIONS
1160 RESTORE6400
1170 READXS,XS:REM THESE VARIABLES ARE
JUST FOR COUNTING
1180 IF XS<>"X" A=A+1:GOTO 1170:REM NOT
THE END OF THE LIST?
1190 REM NOW PUT THE CONDITIONS AND ACT
IONS INTO ARRAYS
1200 DIM CONS(A):DIM ACTS(A)
1210 RESTORE 6400:REM BACK TO THE START
OF THE LIST
1220 FOR X=1 TO A:READ CONS(X),ACTS(X):
NEXT
2015 ON DIS GOTO 2020,2191
2196 DIS=1:REM RESET DISPLAY TO NORMAL
2999 REM CONDITIONS/ACTIONS
3000 APOS=0:REM WHERE WE ARE IN THE CO
NDITION/ACTION TABLE
3015 WRDFLAG=0:REMTTHIS WILL TELL US T
HAT WORDS MATCHED
3020 APOS=APOS+1:IF APOS=A+1 RETURN:REM
HAVE WE BEEN THROUGH THE LIST?
3030 IF VAL (LEFT$(CONS(APOS),2))<>W(1
) GOTO 3020:REM DOES THE FIRST WORD MATC
H
3040 IF VAL (MID$(CONS(APOS),3,2))<>W(
2) AND VAL (MID$(CONS(APOS),3,2))>0 GOTO
3020:REM DOES THE SECOND WORD MATCH OR
IS THE CONDITION 0
```

```
3050 WRDFLAG=1
3060 GOSUB 3500:REM FOUND A MATCH, GO A
ND DO THE ACTIONS. THIS WILL BE MORE COM
PLICATED LATER.
3100 RETURN
3499 REM DO ACTIONS
3500 P=0:REM POINTER IN ACTION CODE
3510 P=P+1: DO=ASC(MID$(ACTS(APOS),P,P)
)
3515 IF DO=ASC("#") RETURN:REM FINISHED
ALL THE ACTIONS?
3520 ON DO-64 GOSUB 3610,3630,3650
3590 GOTO 3510
3609 REM CARRY AN OBJECT
3610 OB=VAL(MID$(ACTS(APOS),P+1,2)):RE
M GET THE OBJECT NUMBER
3613 IF OBJROOM(OB)=-1 MESS1$="You've
already got it.":GOTO3625:REM ARE WE ALR
EADY CARRYING IT
3615 IF OBJROOM(OB)<>ROOM MESS1$="It i
sn't here.":GOTO3625:REM NOT IN THIS ROO
M
3620 OBJROOM(OB)=-1:MESS1$="You take i
t.":REM MOVE THE OBJECT TO THE INVENTORY
3625 P=P+2:DIS=1:RETURN:REM MOVE THE PO
INTER TO THE NEXT ACTION
3629 REM DROP AN OBJECT
3630 OB=VAL(MID$(ACTS(APOS),P+1,2)):RE
M GET THE OBJECT NUMBER
3635 IF OBJROOM(OB)>-1 MESS1$="You hav
en't got it.":GOTO3645:REM ARE WE CARRIY
NG IT
3640 OBJROOM(OB)=ROOM:MESS1$="You drop
it.":P=P+2:RETURN:REM MOVE THE OBJECT T
O THE CURRENT ROOM
3645 P=P+2:DIS=1:RETURN:REM MOVE THE PO
INTER TO THE NEXT ACTION
3649 REM INVENTORY
```

```
3650 TEMP=0:REM CHECK TO SEE IF NOTHING
IS CARRIED
3653 CLS:PRINT"You are carrying.":FORX=
1TO0:IF OBJROOM(X)=-1 PRINTOBJECT$(X):TE
MP=1
3655 NEXT X
3657 IF TEMP=0 PRINT"Nothing useful."
3660 DIS=2:RETURN
5060 DATA12,TAKE,12,GET,12,BRIN
5070 DATA11,DROP,11,LEAV,11,DUMP
5200 DATA50,LAZE,50,RIFL
5210 DATA51,SLEE,51,BOMB
5220 DATA52,RED
5230 DATA53,GREE
5240 DATA54,SPAC,54,SUIT
5250 DATA55,SPAN
5260 DATA56,CROW,56,BAR
5270 DATA57,ALIE
5280 DATA58,MASK
5290 DATA59,CABL
5310 DATA61,LIGH
5320 DATA62,BUTT
5490 DATA99,INVE
6399 REM CONDITIONS AND ACTIONS IN HERE
6400 DATA1250#,A01#
6410 DATA1251#,A02#
6420 DATA1254#,A05#
6430 DATA1255#,A06#
6440 DATA1256#,A07#
6450 DATA1258#,A12#
6460 DATA9900#,C#
6465 DATA1150#,B01#
6470 DATA1151#,B02#
6475 DATA1154#,B05#
6480 DATA1155#,B06#
6485 DATA1156#,B07#
6490 DATA1158#,B12#
7000 DATA,X
```

Add action to your **PART 5** Adventure!

The excitement really starts as you USE the objects scattered round your maze

If you've been typing in the *Let's Compute!* adventure over the past few months you can already take and drop some of the objects. But the real thrill comes when you do something with them.

This means you need to add more actions and conditions to make the game more interesting.

Type in Lines 3060 to 3110. These let us have more complicated conditions than before.

You will see that the operation is similar to the way we tackled the actions. Each condition will have a letter and some numbers after it.



Again all the numbers must have two digits. And, just like the actions, it is really easy to add new conditions.

The **CFLAG** variable checks to see if the conditions are true. We only want to carry out an action if they are ALL true, so each will set **CFLAG** to one if it is false.

If the condition loop finds it has reached the # at Line 3075 all the conditions must be true. The program then goes on to perform the actions.

At the moment we'll just have one condition:

Condition Ann: Check if object nn is here or being carried.

Type in the new Line 3520 to add this. We're also going to add a new action:

Action Dnnmm: Move object nn to room mm.

Enter Lines 3669 to 3680 to put this in your program.

You will see that Line 3680 adds four to the action pointer - **P**. This is because there are two numbers instead of one.

This action does not affect the display so **DIS** is not set.

You want your player to be able to fix the loose cable and press the buttons in the control room. So you need to add Lines 5080 and 5085. These put the words you need into the program.

They include several ways of saying **FIX**: So **MEND** and **REPAIR** will also do the job. **PUSH** and **PRESS** also mean the same as each other.

You should always think of the different words players might try to use. Don't let them get bored hunting for the one and only right one!

Now we add the conditions and actions. Look at Line 6495. This first checks for words 20 and 59 - **FIX CABLE**.

If they're found, it uses condition A to check that the player is in the room with the spanner and the loose cable - Objects 06 and 13.

If that is true it then goes to action D. First it sets the loose cable to Room 0 - our store room. Then it sets the fixed cable - Object 14 - to Room 16.

So typing **FIX CABLE** will replace the loose cable with the fixed one. But your player must be in a room with the loose cable and the spanner - Room 16 - as the cable cannot be moved.



Type in Lines 6495 to 6505. The first of these sorts out what the program has to do when the player asks for the cable to be fixed.

The other two deal with pressing the buttons in the control room. Can you work out what effect they have? If you can't, just run the game, press the green button and go to the air lock.

If you use the conditions and actions you already have and add a word or two you should be able to put the alien to sleep. Have a go at that.

● Don't worry if you have problems: We'll reveal all next month!

IS YOUR COMPUTER HERE?

This program works on a BBC, Electron, CPC, Amiga and PC (GW-Basic)

It also works on an Atari ST using Stos. But if you're using Stos replace **CONS** with **CNS** throughout your program.

This adventure will not work on a C64/128 or Spectrum.

```
3060 P=4:REM POINTER IN CONDITION CODE
SET PAST WORD-NUMBERS
3065 CFLAG=0:REM CONDITION FLAG STAYS ZERO IF CONDITIONS ARE TRUE
```

```
3070 P=P+1: DO=ASC(MID$(CONS(APOS),P,P))
)
3075 IF DO=ASC("#") THEN GOSUB 3500:RETURN:REM IF ALL CONDITIONS TRUE, DO ACTION, GO BACK TO MAIN PROGRAM
3080 ON DO-64 GOSUB 3100
3085 IF CFLAG=1 THEN GOTO 3020:REM IF CONDITION NOT TRUE: GO TO NEXT SET OF CONDITIONS
3090 GOTO 3070:REM CONDITION TRUE, GOTO NEXT CONDITION
3099 REM CHECK IF OBJECT IS HERE OR CARRIED
3100 OB=VAL(MID$(CONS(APOS),P+1,2)):REM GET THE OBJECT NUMBER
3105 IF OBJROOM(OB)<>ROOM AND OBJROOM(OB)>-1 CFLAG=1:REM IF NOT HERE OR CARRIED
```

```
SET FLAG
3110 P=P+2:RETURN:REM MOVE POINTER TO NEXT CONDITION
3520 ON DO-64 GOSUB 3610,3630,3650,3670
3669 REM MOVE OBJECTnn TO ROOM nn
3670 OB=VAL(MID$(ACT$(APOS),P+1,2)):REM GET THE OBJECT NUMBER
3675 RO=VAL(MID$(ACT$(APOS),P+3,2)):REM GET THE ROOM NUMBER
3680 OBJROOM(OB)=RO:P=P+4:RETURN:REM MOVE THE OBJECT, RESET POINTER TO NEXT ACTION
5080 DATA20, FIX, 20, MEND, 20, REPA
5085 DATA21, PUSH, 21, PRES
6495 DATA2059A06A13#, D1300D1416#
6500 DATA2152A03#, D0900D0808#
6505 DATA2153A04#, D0800D0908#
```