COMPUTER SPACEGAMES

For ZX Spectrum, ZX81, BBC, TRS-80, APPLE, VIC & PET



GAMES PROGRAMS FOR YOUR MICRO

USBORNE COMPUTER PROGRAMS

COMPUTER SPACEGAMES

Daniel Isaaman
and Jenny Tyler

Contents

2 About this Book
4 Starship Takeoff
6 Intergalactic Games
8 Evil Alien
10 Beat the Bug Eyes
12 Moonlander
14 Monsters of Galacticon
16 Alien Snipers
18 Asteroid Belt
20 Trip into the Future
22 Death Valley
24 Space Mines
26 Space Rescue
30 Touchdown: TRS-80 version
31 Touchdown: VIC 20 version
32 Touchdown: ZX81 version
33 Touchdown ZX Spectrum version
34 Touchdown: BBC version
35 Touchdown: Apple version
36 Adding to the programs
38 Writing your own programs
40 Summary of BASIC
46 Conversion chart
47 Puzzle Answers

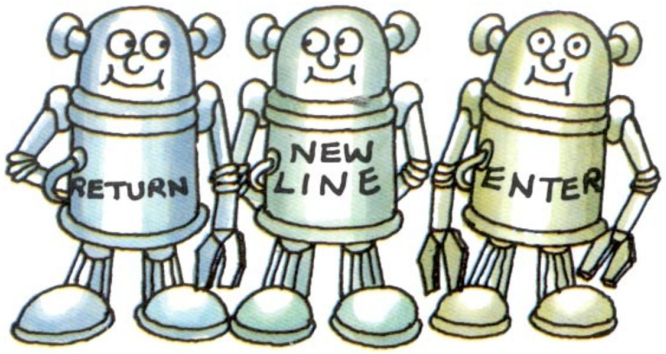
Illustrated by
Martin Newton, Tony Baskeyfield, Graham Round, Jim Bamber, Mark Duffin and John Bolton

Designed by
Graham Round and Roger Priddy

First published in 1982 by Usborne Publishing, 20 Garrick Street, London WC2E 9BJ, England
© 1982 Usborne Publishing Ltd

Beat the Bug Eyes program by Bob Merry
Starship Takeoff program by Richard Nash

About This Book



This book contains simple games programs to play on a microcomputer. They are written for use on ZX81, ZX Spectrum, BBC, VIC 20, TRS-80 and Pet and Apple micros, and many are short enough to fit into the ZX81's 1K of memory.

Most micros use the language BASIC, but they all have their own variations or dialects. In this book, the main listing for each program works on the ZX81 and lines which need changing for the other computers are marked with symbols and printed underneath. The fact that the programs are written for several micros means that they do not make full use of each one's facilities. You could try finding ways of making the programs shorter and neater for your micro.

For each game, there are ideas for changing and adding to the programs and towards the back of the book you will find tips and hints on writing games of your own. Also in the book is a conversion chart to help you adapt programs in magazines and other books for your micro and a summary of the BASIC terms used in this book.

Typing in the programs

Lines which need changing for computers other than ZX81 are marked with these symbols:

- ▲ VIC and Pet
- ★ BBC and Acorn Electron
- ■ TRS-80
- ● Apple
- s ZX Spectrum

Every time you see the symbol for the micro you are using, look below for the corresponding line number with the same symbol and type that in instead.

VIC 20 versions of all except the graphics program should work on Pet computers.

Points to remember

1 Type the lines exactly as they are printed including all punctuation and spaces.

2 Press RETURN, NEWLINE or ENTER key at the end of each program line.

3 Check each line as you go.

4 Make sure you don't miss out a line or confuse one with another. A piece of paper or a ruler is useful to mark your place in the listing.

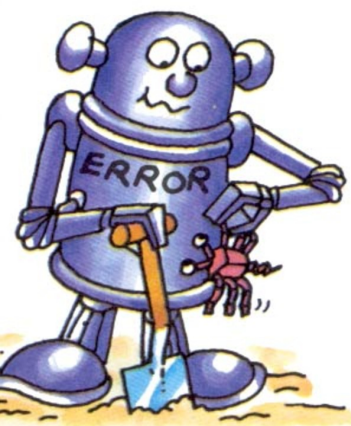
5 Look out for the symbols and make sure you use the correct lines for your computer.

6 If you are using a ZX81 or ZX Spectrum, remember not to type the program instructions letter by letter but to use the special key for each instruction instead.

You may find it easier to get someone to read the program out to you while you type it in. Don't forget to explain that they must read every comma, fullstop, bracket and space and differentiate between letter "O" and zero, FOR and 4, and TO and 2.

Debugging programs

When you have typed in the program, check your manual to find out how to display it on the screen. (Usually you type LIST followed by the line numbers of the section you want to see.)







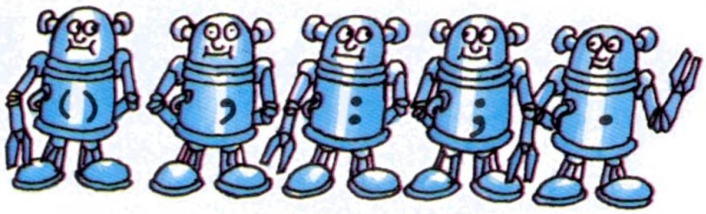
Check you have typed everything correctly. It is easy to make mistakes, so expect to find some. Use your manual to find out how to make changes to the program once it is typed in. If in doubt, you can always type the line again. All the computers will replace an existing line with a new one with the same number.

Here is a checklist of common mistakes to look out for:

1 Line missed out

2 Line wrongly numbered

3 The beginning of one line joined onto the end of another.



4 Brackets, commas, colons, semi-colons, fullstops or spaces missed out, especially in long, complicated lines. Watch for double brackets in particular.

5 Wrong line used for your computer.

6 Letter "O" confused with zero.

7 Wrong numbers used, e.g. extra zeros included.

Playing the games

To start the game you must type RUN. In some games things happen very quickly, so make sure you have read the instructions and know what you are supposed to do.

It is quite likely that the program still has a mistake in it and either won't run at all or the game won't work properly. Sometimes your computer will give you an error code which you can look up in the manual. This may help you find the mistake, though not always. List the program again and check it carefully against the book.

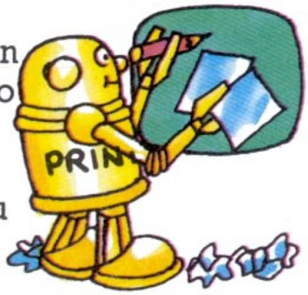
When the game is over, the computer will usually say something like BREAK IN LINE 200. To play again, you have to type RUN.

Experimenting with the games

There are suggestions for changing and adding to the programs throughout the book, but don't be afraid to experiment with changes of your own. You can't damage the computer and you can always change back to the original if the changes don't work.

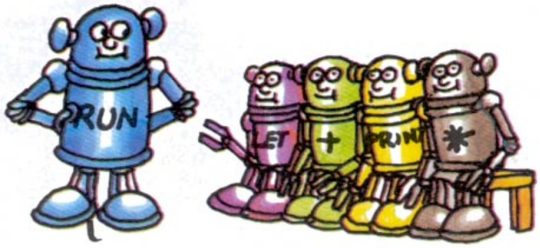
You will probably find you want to adjust the speed of some games,* especially after you have played them a number of times. You will find out which line to change on each program page.

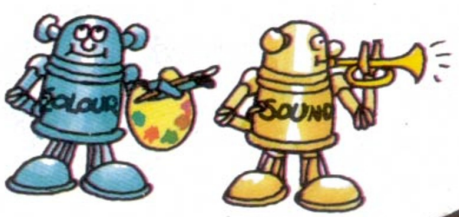
Wherever you see PRINT, you can change the message in quotes that follows it to whatever you like. Also, unless you have ZX81 with only 1K, you can add extra messages.



Type a line number (say 105 if you want to add a message between lines 100 and 110), then type PRINT, then your message inside quotes.

If your computer can make colours and sounds, you could use your manual to find out how they work and try adding them to the games in this book.

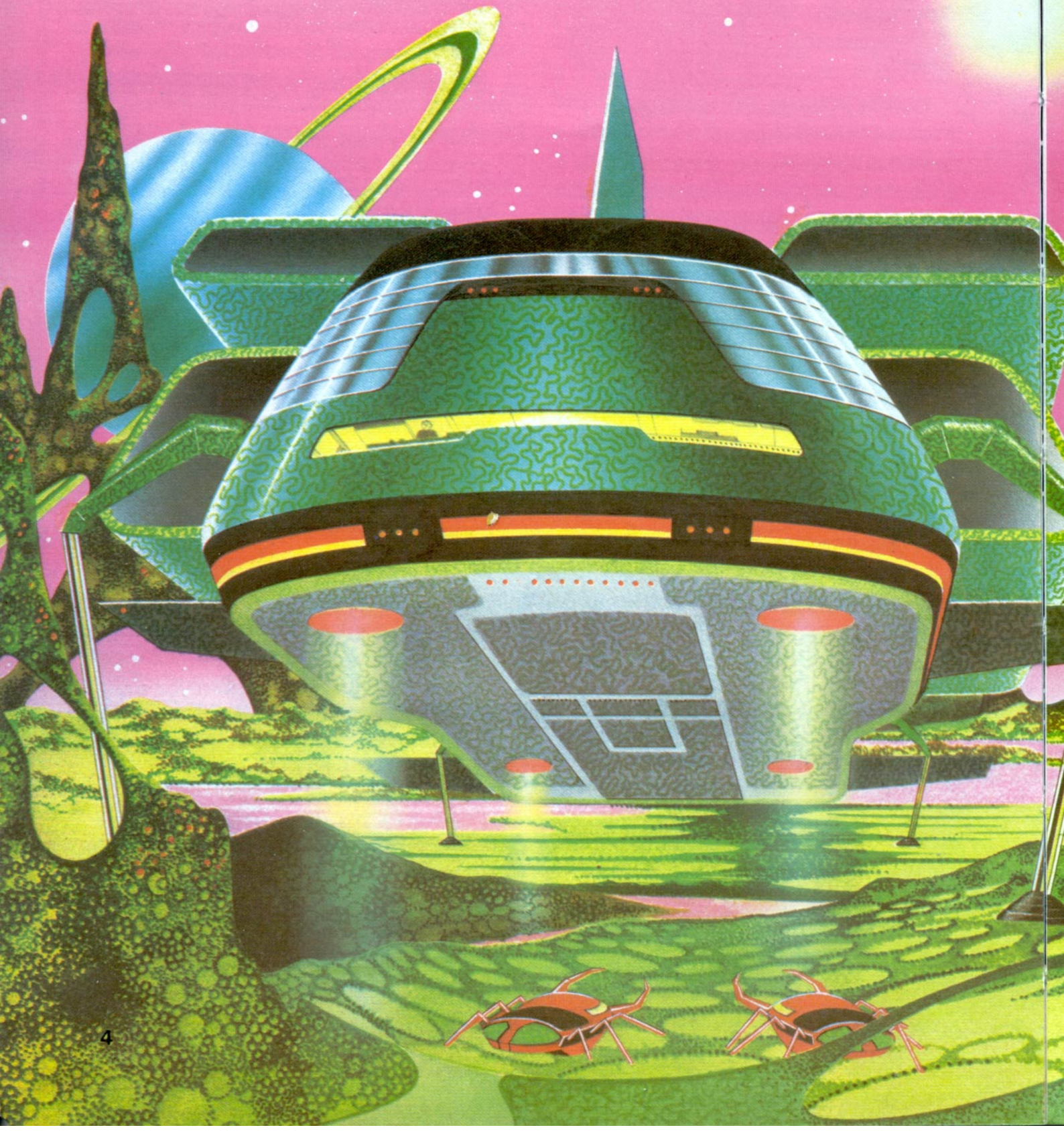




*See page 37 for a special note for BBC and Spectrum users.

Starship Takeoff

You are a starship captain. You have crashed your ship on a strange planet and must take off again quickly in the alien ship you have captured. The ship's computer tells you the gravity on the planet. You must guess the force required for a successful take off. If you guess too low, the ship will not lift off the ground. If you guess too high, the ship's fail-safe mechanism comes into operation to prevent it being burnt up. If you are still on the planet after ten tries, the aliens will capture you.



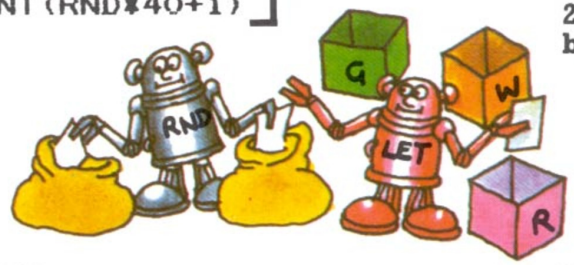
1K

How the program works

```
▲●10 CLS
  20 PRINT "STARSHIP TAKE-OFF"
★■▲●30 LET G=INT(RND*20+1)
★■▲●40 LET W=INT(RND*40+1)
```

Clears the screen.

Computer selects two numbers – one between 1 and 20 to be put in G, the other between 1 and 40 to be put in W.




```
  50 LET R=G*W
```

Multiplies the number in G by number in W. Puts result in R.

```
  60 PRINT "GRAVITY= ";G
```

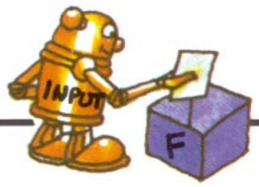
Prints GRAVITY and number in G.

```
  70 PRINT "TYPE IN FORCE"
```

Asks you for a number.

```
  80 FOR C=1 TO 10
```

This begins a loop which tells the computer to repeat the next section 10 times, to give you 10 goes.




```
  90 INPUT F
```

Stores your number in F.

```
 100 IF F>R THEN PRINT "TOO HIGH";
 110 IF F<R THEN PRINT "TOO LOW";
 120 IF F=R THEN GOTO 190
```

Compares number in F with number in R and prints appropriate message or jumps to 190.

```
 130 IF C<>10 THEN PRINT ", TRY AGAIN"
```

Prints if you've had less than 10 goes without a correct answer.

```
 140 NEXT C
```

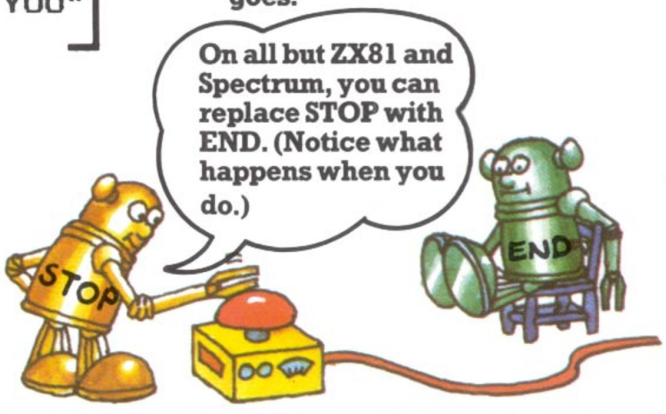
End of loop. Goes back to 80 for another turn.

```
 150 PRINT
 160 PRINT "YOU FAILED -"
 170 PRINT "THE ALIENS GOT YOU"
 180 STOP
 190 PRINT "GOOD TAKE OFF"
```

Prints after 10 unsuccessful goes.

The above listing will work on a ZX81. For other computers, make the changes below.

```
   ●10 HOME
   ▲10 PRINT CHR$(147)
★▲●30 LET G=INT(RND(1)*20)
   ■30 LET G=INT(RND(0)*20)
★▲●40 LET W=INT(RND(1)*40)
   ■40 LET W=INT(RND(0)*40)
```



How to make the game harder

You can change the program to give you less than 10 goes. Do this by altering the last number in line 80 and the number in line 130. (They must be the same.)

See page 2 for meaning of ●▲■★



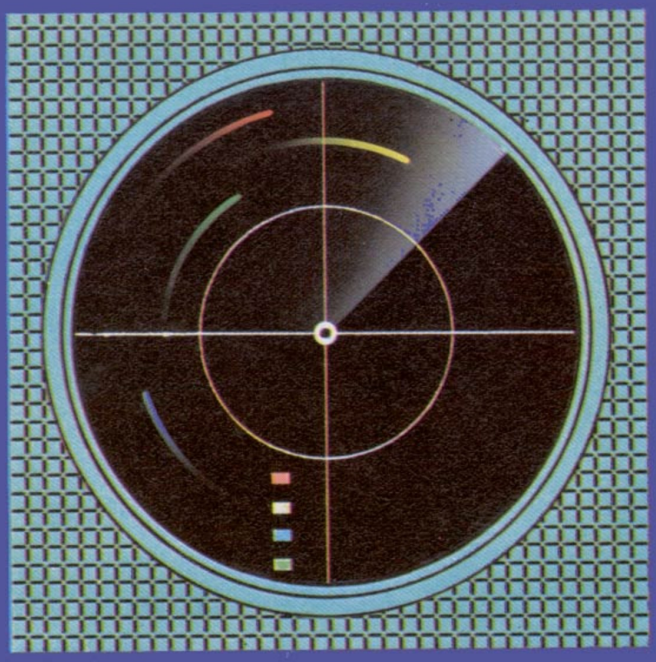
Puzzle corner

You could change the range of possible forces. See if you can work out how.

Intergalactic Games

There is fierce competition among the world's TV companies for exclusive coverage of the First Intergalactic Games. Everything depends on which company wins the race to put a satellite into orbit at the right height.

You are the Engineer in charge of the launch for New Century TV. The crucial decisions about the angle and speed of the launching rocket rests on *your* shoulders. Can you do it?



How the program works

```
    10 PRINT "INTERGALACTIC GAMES"
★■▲●20 LET H=INT(RND*100+1)
    30 PRINT "YOU MUST LAUNCH A SATELLITE"
    40 PRINT "TO A HEIGHT OF ";H
    50 FOR G=1 TO 8
    60 PRINT "ENTER ANGLE (0-90)"
    70 INPUT A
    80 PRINT "ENTER SPEED (0-40000)"
    90 INPUT V
   100 LET A=A-ATN(H/3)*180/3.14159
   110 LET V=V-3000*SQR(H+1/H)
   120 IF ABS(A)<2 AND ABS(V)<100 THEN GOTO 210
   130 IF A<-2 THEN PRINT "TOO SHALLOW"
   140 IF A>2 THEN PRINT "TOO STEEP"
   150 IF V<-100 THEN PRINT "TOO SLOW"
   160 IF V>100 THEN PRINT "TOO FAST"
   170 NEXT G
   180 PRINT "YOU'VE FAILED"
   190 PRINT "YOU'RE FIRED"
   200 STOP
   210 PRINT "YOU'VE DONE IT"
   220 PRINT "NCTV WINS-THANKS TO YOU"
   230 STOP
```

Chooses the height to which you must launch your satellite, puts it in H and prints it.

Beginning of loop to give you 8 goes.

Asks you for an angle and puts it in A.

Asks you for a speed and puts it in V.

Uses H to calculate what the angle should be and subtracts this from your guess to find out how close you were.

Works out what the speed should be and subtracts it from your guess.

Checks if you were close enough to win and if so jumps to 210.

Prints an appropriate comment to help you with your next go.

Goes back for another go.

Prints after 8 unsuccessful goes.

Prints if you win.

The above listing will work on a ZX81. For other computers, make the changes below.

```
   ■20 LET H=INT(RND(0)*100+1)
 ★▲●20 LET H=INT(RND(1)*100+1)
```

Adding to the program

These three extra lines will make the computer give you bonus points depending on how quickly you make a successful launch.

```
222 LET B=INT(1000/G)
225 PRINT "YOU'VE EARNED A"
227 PRINT "BONUS OF ";B;" CREDITS"
```



Puzzle corner

Can you change the program so that, if you win, it automatically goes back for another game, adding the bonus you've already earned to any new bonus? (Hint: you need to change two lines and add one.)

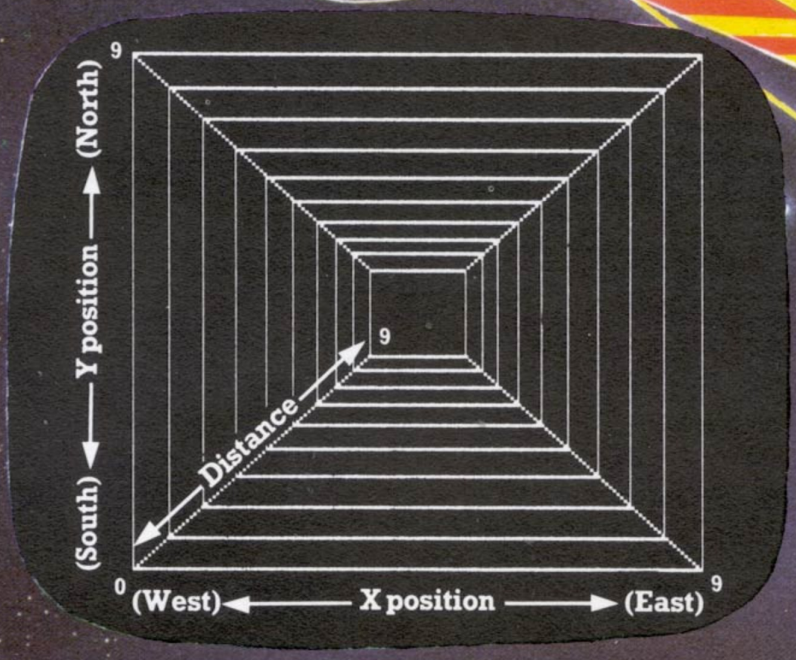
See how long you can play before NCTV fires you.



See page 2 for meaning of ●▲■★

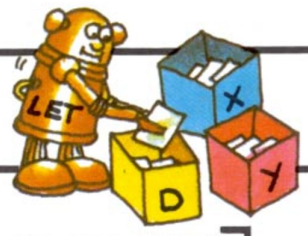
Evil Alien

Somewhere beneath you, in deepest, blackest space, lurks Elron, the Evil Alien. You have managed to deactivate all but his short-range weapons but he can still make his ship invisible. You know he is somewhere within the three-dimensional grid you can see on your ship's screen (see below), but where? You have four space bombs. Each one can be exploded at a position in the grid specified by three numbers between 0 and 9, which your computer will ask you for. Can you blast the Evil Elron out of space before he creeps up and captures you?



How the program works

```
▲●5 CLS
   10 PRINT "EVIL ALIEN"
   20 LET S=10
   30 LET G=4
★■▲●40 LET X=INT(RND*S)
★■▲●50 LET Y=INT(RND*S)
★■▲●60 LET D=INT(RND*S)
   70 FOR I=1 TO G
   80 PRINT "X POSITION (0 TO 9)?"
   85 INPUT X1
   90 PRINT "Y POSITION (0 TO 9)?"
  100 INPUT Y1
  110 PRINT "DISTANCE (0 TO 9)?"
  120 INPUT D1
  130 IF X=X1 AND Y=Y1 AND D=D1 THEN GOTO 300
  140 PRINT "SHOT WAS ";
  150 IF Y1>Y THEN PRINT "NORTH";
  160 IF Y1<Y THEN PRINT "SOUTH";
  170 IF X1>X THEN PRINT "EAST";
  180 IF X1<X THEN PRINT "WEST";
  190 PRINT
  200 IF D1>D THEN PRINT "TOO FAR"
  210 IF D1<D THEN PRINT "NOT FAR ENOUGH"
  220 NEXT I
  230 PRINT "YOUR TIME HAS RUN OUT!!"
  240 STOP
  300 PRINT "*BOOM* YOU GOT HIM!"
  310 STOP
```



Sets the size of the grid.

Sets the number of goes.

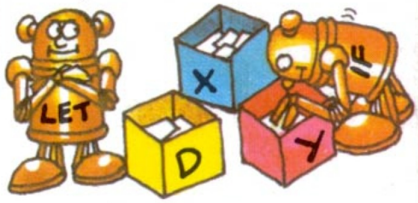
Elron's position is fixed by these 3 lines, which select 3 numbers between 0 and the size of the grid.

Start of a loop which tells the computer to repeat the next 15 lines G times.

This section asks you for your 3 numbers and stores them in X1, Y1 and D1.

Checks if you were right and jumps to 300 if you were.

Your guesses are compared with Elron's position and a report printed.



End of loop. Returns for another go.

Prints if you've used up all your goes.

Prints if you guessed right.

The above listing will work on a ZX81. For other computers, make the changes below.

```
   ●5 HOME
   ▲5 PRINT CHR$(147)
★▲●40 LET X=INT(RND(1)*S)
   ■40 LET X=INT(RND(0)*S)
★▲●50 LET Y=INT(RND(1)*S)
   ■50 LET Y=INT(RND(0)*S)
★▲●60 LET D=INT(RND(1)*S)
   ■60 LET D=INT(RND(0)*S)
```

How to make the game harder

This program has been written so that you can easily change the difficulty by changing the size of the grid. To do this, put a different value for S in line 20.

If you increase the grid size you will need more space bombs to give you a fair chance of blasting Elron. Do this by changing the value of G in line 30.

Puzzle corner

Can you work out how to change the program so that the computer asks you for a difficulty number which it can put into S instead of S being fixed? (Tip: limit the value of S to between 6 and 30 and use INT(S/3) for the value of G in line 30.)



See page 2 for meaning of ●▲■★

Beat the Bug Eyes

You're trapped! Everywhere you turn you catch a glimpse of the steely cold light of a space bug's eyes before it slithers down behind a rock again. Slowly the bugs edge towards you, hemming you in, waiting for a chance to bind you in their sticky web-like extrusions. Luckily you have your proton blaster with you.

The bug eyes pop up in four different places on your screen and these correspond to keys 1 to 4. Press the correct key while the bug's eyes are on the screen and you will blast it. There are 10 bugs in all – the more you blast, the greater your chance of escape.

How the program works

```
10 PRINT "BUG EYES"
20 LET S=0
30 FOR T=1 TO 10
▲●40 CLS
★■▲●50 FOR I=1 TO INT(RND*30+20)
   60 NEXT I
★■▲●70 LET R=INT(RND*4+1)
★■▲●80 GOSUB 210+30*R
   90 PRINT "OO"
★■▲●100 FOR I=1 TO 20
 ●→
 ★▲●110 LET R$=INKEY$
   120 IF R$<>"" THEN GOTO 140
   130 NEXT I
   140 IF VAL("0"+R$)<>R THEN GOTO 210
   150 LET S=S+1
```

Sets the score to zero for start of game.

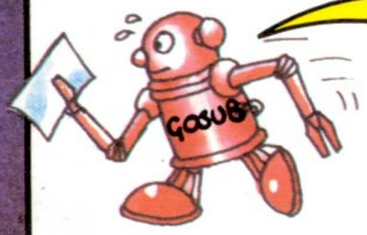
Beginning of loop which gives you 10 turns.

Clears screen.

Delay loop which lasts a varying length of time, depending on the value of RND.

Chooses a number from 1 to 4 and puts it in R.

Jumps to one of four sub-routines depending on the value of R. This gets two numbers which correspond to a position on the screen – "A" spaces across and "D" lines down – and then jumps again to 350 to move the cursor to this position.



GOSUB makes the computer branch out of the main program to a "sub-routine" (see next page). RETURN at the end of the sub-routine sends it back to the main program again.

Prints the "bug eyes" at this position.

Loops round to see if you are pressing a key. If you are, computer jumps to 140 and checks if it is the right one.

Increases score by 1.

```
▲●160 CLS
   170 GOSUB 350
   180 PRINT "*"
★■▲●190 FOR J=1 TO 40
   200 NEXT J
   210 NEXT T
   220 PRINT "YOU BLASTED ";S;"/10 BUGS"
   230 STOP
   240 LET D=5
   250 LET A=1
   260 GOTO 350
   270 LET D=1
   280 LET A=9
   290 GOTO 350
   300 LET D=5
   310 LET A=18
   320 GOTO 350
   330 LET D=10
   340 LET A=7
   350 FOR I=1 TO D
   360 PRINT
   370 NEXT I
   380 PRINT TAB(A);
   390 RETURN
```

Clears screen

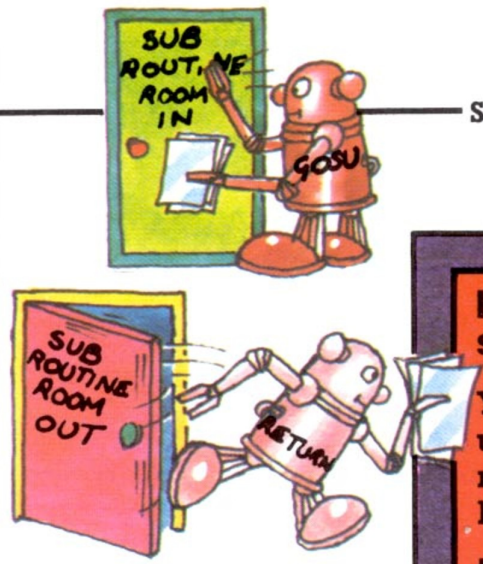
Sends cursor back to the same position and prints a star.

Delay loop to make star stay on screen long enough for you to see it.

Goes back for next turn.

Prints score.

Sub-routines.



The above listing will work on a ZX81. For other computers, make the changes below.

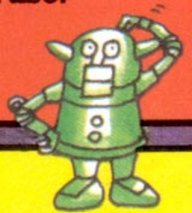
```
   ●40,160 HOME
   ▲40,160 PRINT CHR$(147)
   ■50 FOR I=1 TO INT(RND(0)*300+200)
 ★▲●50 FOR I=1 TO INT(RND(1)*300+200)
   ■70 LET R=INT(RND(0)*4+1)
 ★▲●70 LET R=INT(RND(1)*4+1)
★■▲●80 ON R GOSUB 240,270,300,330
★■▲●100 FOR I=1 TO 150
   ●105 R$=""
   ▲110 GET R$
   ★110 R$=INKEY$(1)
   ●110 IF PEEK(-16384)>127 THEN GET R$
★■▲●190 FOR J=1 TO 300
```

How to change the speed

You can speed the game up by changing the last number in line 100 to a lower one.

How to use more of the screen

This program was written to fit on the smallest screen width (which is the VIC 20). For the other computers, you can increase the values of A in lines 250, 280, 310 and 340. Check your manual to find the maximum width your computer can use.



Puzzle corner

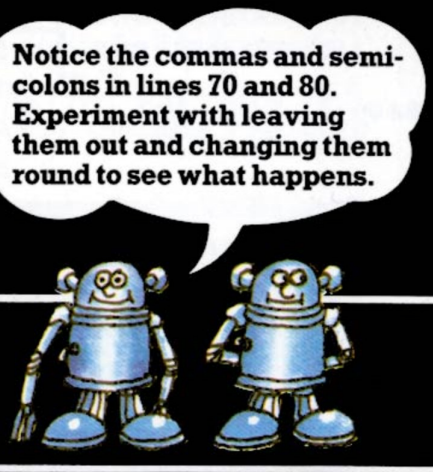
Can you change the program to make the bugs appear in more than four places on the screen? Can you add more bugs too?

See page 2 for meaning of ●▲■★

Moonlander

You are at the controls of a lunar module which is taking a small team of astronauts down to the moon's surface. In order to land safely you must slow down your descent but that takes fuel and you have only a limited supply.

Your computer will tell you your starting height, speed and fuel supply and ask how much fuel you wish to burn. It will then work out your new height and speed. A burn of 5 will keep your speed constant. A higher number will reduce it. Try to have your speed as close to zero as you can when you land. Can *you* land safely on the moon?

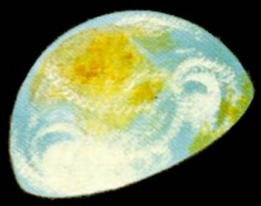



```
▲●10 CLS
  20 PRINT "MOONLANDER"
  30 LET T=0
  40 LET H=500
  50 LET V=50
  60 LET F=120
  70 PRINT "TIME";T,"HEIGHT";H
  80 PRINT "VEL.";V,"FUEL";F

  90 IF F=0 THEN GOTO 140

 100 PRINT "BURN? (0-30)"
 110 INPUT B
 120 IF B<0 THEN LET B=0
 130 IF B>30 THEN LET B=30
 140 IF B>F THEN LET B=F

 150 LET V1=V-B+5
 160 LET F=F-B
 170 IF (V1+V)/2>=H THEN GOTO 220
```



```
 180 LET H=H-(V1+V)/2
 190 LET T=T+1
 200 LET V=V1

 210 GOTO 70

 220 LET V1=V+(5-B)*H/V
 230 IF V1>5 THEN PRINT "YOU CRASHED-ALL DEAD"
 240 IF V1>1 AND V1<=5 THEN PRINT "OK-BUT SOME INJURIES"
 250 IF V1<=1 THEN PRINT "GOOD LANDING."
 260 STOP
```

The above listing will work on a ZX81. For other computers, make the changes below.

```
●10 HOME
▲10 PRINT CHR$(147)
```


See page 2 for meaning of ●▲



How the program works

Sets the starting values for time, height, speed and fuel and prints them.

If you have no fuel left, computer jumps down the program, bypassing the section which asks you for a burn. It then prints a running commentary of your progress as you approach the moon's surface.

Gets a number from you for the amount of fuel you wish to burn and checks it is within the correct limits.

Calculates your new speed, V1.

Calculates your new fuel level.

Checks if the distance travelled in your last go is greater or equal to your height above the moon. If it is, you've landed. Computer then jumps down program to see how good a landing you made.

Calculates your new height.

Increases time by 1.

Puts your new velocity into V so it will print in line 80 for your next go.

Goes back to beginning of loop for next go.

Calculates your speed on touch-down and checks what kind of landing it gives you.

Adding to the program

If you add the following lines, you will see a star printed each go. The distance of the star from the left-hand side of the screen corresponds to your height above the moon.

```
85 FOR I=2 TO H/500*nn
86 PRINT " ";
87 NEXT I
88 PRINT "*"
```

Replace nn with the width of your screen.

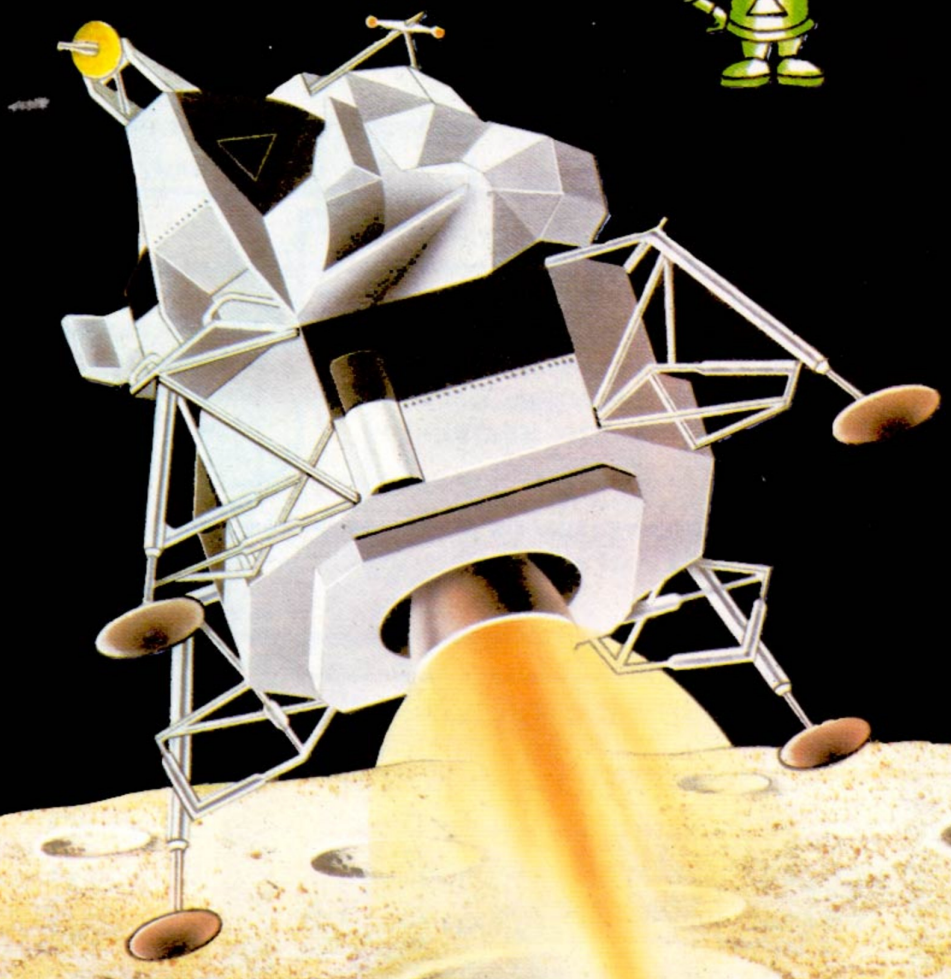
Changes to try

Try changing the values of H, V and F in lines 40 to 60 and see what happens.

Puzzle corner

You could make the game easier by increasing the maximum speed allowed for a safe landing. How would you change the program to do this?





Monsters of Galacticon

Landing on Galacticon was easy – but no-one warned you that some of the nastiest monsters in the known Universe are to be found there.

As each monster threatens, you must choose one of your weapons – a ray gun, a trypton blaster or a sword laser – to use against it. Can you make the right choices? If so, you may live to conquer Galacticon.



```
    10 PRINT "MONSTERS OF GALACTICON"

    20 DIM M$(4)

    30 LET N=4

    40 LET M=5

    50 LET M$(1)="SULFACIDOR"
    60 LET M$(2)="FLAMGONDAR"
    70 LET M$(3)="BALNOLOTIN"
    80 LET M$(4)="GOLANDOR"

    90 FOR I=1 TO N
★■▲●100 LET A=INT(RND*N+1)
★■▲●110 LET B=INT(RND*N+1)
   120 LET T$=M$(A)
   130 LET M$(A)=M$(B)
   140 LET M$(B)=T$
   150 NEXT I

   160 FOR T=1 TO 8
  ▲●170 CLS
★■▲●180 LET R=INT(RND*N+1)
   190 PRINT "MONSTER COMING..."
   200 PRINT "IT'S A ";M$(R)

   210 PRINT "WHICH WEAPON ? (R,S OR T) "
   220 INPUT R$
s★■▲●230 LET W=CODE(R$)-54+R
 ★■▲240 LET W=W-3*(W>3)-3*(W>6)

   250 IF W=2 THEN GOTO 300
```



How the program works

Sets up a storage place (an "array") labelled M$ with 4 compartments in it – M$(1), M$(2), M$(3) and M$(4) – one for each monster name.

Sets the number of monsters to 4.

Sets the number of people in your group to 5.

Puts the 4 monster names into the array.

These lines shuffle the monsters up. Computer loops round N times. Each loop, it selects two numbers between 1 and N and switches the names in the compartments with those numbers. T$ is a temporary string used during the switch round process.

Beginning of loop for 8 turns.

Clears screen

Chooses one of the monsters and prints its name.

The computer uses the values in R and R$ to work out a weapon number, W, which will be 1, 2 or 3.

If W is 2, the computer jumps to 300 to say you've killed the monster.

See page 2 for meaning of ●▲■★s



```
260 IF W=3 THEN GOTO 320
270 PRINT "NO USE. IT'S EATEN ONE"
280 PRINT "OF YOUR GROUP"
290 GOTO 360
300 PRINT "YOU'VE KILLED IT"
310 GOTO 380
320 PRINT "NO EFFECT"
★■▲●330 IF RND>.4 THEN GOTO 380
340 PRINT "YOU ANGERED THE ";M$(R);"AND IT"
350 PRINT "KILLED ONE OF YOUR GROUP"

360 LET M=M-1
370 IF M<1 THEN GOTO 440

★■▲●380 FOR I=1 TO 20
390 NEXT I
400 NEXT T
410 PRINT "YOU HAVE SURVIVED TO"
420 PRINT "CONQUER GALACTICON"
430 STOP
440 PRINT "YOU'RE ALL DEAD"
450 STOP
```

If W is 3, computer jumps to 320 to say "no effect".

Prints if W is *not* 2 or 3. Then jumps to 360 to decrease number of people in group by one.

Leading on from the "no effect" line is a random possibility of the monster being angered and killing someone.

Checks if you still have some people left. If you haven't it jumps to 440 to tell you.

Delay loop to keep messages on screen long enough for you to read them.

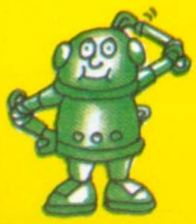
Goes back for next turn.

Prints if you still have people left after all your turns.

The above listing will work on a ZX81. For other computers, make the changes below.

```
   ■100 LET A=INT(RND(0)*N+1)
 ★▲●100 LET A=INT(RND(1)*N+1)
   ■110 LET B=INT(RND(0)*N+1)
 ★▲●110 LET B=INT(RND(1)*N+1)
   ●170 HOME
   ▲170 PRINT CHR$(147)
   ■180 LET R=INT(RND(0)*N+1)
 ★▲●180 LET R=INT(RND(1)*N+1)
★■▲●230 LET W=ASC(R$)-81+R
   $230 LET W=CODE(R$)-81+R
 ★■▲240 LET W=W+3*(W>3)+3*(W>6)
 ★▲●330 IF RND(1)>.4 THEN GOTO 380
   ■330 IF RND(0)>.4 THEN GOTO 380
★■▲●380 FOR I=1 TO 300
```

Puzzle corner



There are at least four ways of making this game harder.
Can you work out what they are?

Alien Snipers



You are the captain of an interstellar cruiser which, through damage to one of its hyperspace motors, has strayed into a forbidden area. Alien sniper ships are attacking you and, to make things worse, are using a jamming device on your radar which makes it give false readings. Luckily your computer knows a code which you can use to work out the correct locations of the enemy ships. But you must be quick – they don't stay in one place for long!

Your computer will print a letter (the false enemy location) and a code number. You must think through the alphabet by that number of letters and type in the letter you get to. E.g. for M 4, you must type Q, and for C 2, you must type E and so on. Typing in a letter automatically triggers off your laser gun, so if your letter is correct you'll score a hit. You can choose the difficulty of each game. This is a number between 1 and 10 and is the maximum number of letters to be added on each time.

You get 10 alien snipers each game. See how many you can hit.

See page 2 for meaning of ●▲■★s

How the program works

1K

```
▲●10 CLS
  20 PRINT "ALIEN SNIPERS"
  30 PRINT
  40 PRINT "DIFFICULTY (1-10)"
  50 INPUT D
  60 IF D<1 OR D>10 THEN GOTO 50
  70 LET S=0
  80 FOR G=1 TO 10
s★■▲●90 LET L$=CHR$(INT(RND*(26-D)+38))
★■▲●100 LET N=INT(RND*D+1)
  110 CLS
  120 PRINT
  130 PRINT L$,N
★■▲●140 FOR I=1 TO 20+D*5
 ★▲●150 LET I$=INKEY$
  160 IF I$<>"" THEN GOTO 190
  170 NEXT I
  180 GOTO 200
★■▲●190 IF I$=CHR$(CODE(L$)+N) THEN LET S=S+1
  200 NEXT G
  210 PRINT "YOU HIT ";S;"/10"
  220 STOP
```

Gets a difficulty number from you, puts it in D, and checks it is within the correct limits.

Sets opening score to zero.

Beginning of loop which gives you 10 goes.

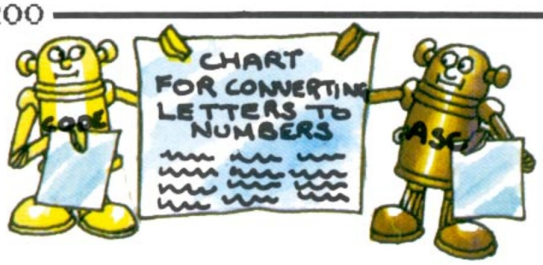
Selects a letter between A and the letter which is your difficulty number before the end of the alphabet.

Selects a number between 1 and D.

Prints the letter and the number.

Checks if you are pressing a key and jumps to 190 if you are.

If you didn't press a key, computer jumps to line 200 which sends it back for another go.



Checks if you the pressed right key and, if so, increases your score by 1.

End of loop. Goes back for another turn.

Prints score after 10 goes.

The above listing will work on a ZX81. For other computers, make the changes below.

```
  ●10,110 HOME
  ▲10,110 PRINT CHR$(147)
  ■90 LET L$=CHR$(INT(RND(0)*(26-D)+65))
★▲●90 LET L$=CHR$(INT(RND(1)*(26-D)+65))
  s90 LET L$=CHR$(INT(RND*(26-D)+65))
  ■100 LET N=INT(RND(0)*D+1)
★▲●100 LET N=INT(RND(1)*D+1)
★■▲140 FOR I=1 TO 200+D*50
  ●140 FOR I=1 TO 100+D*50
  ●145 I$=""
  ★150 LET I$=INKEY$(1)
  ▲150 GET I$
  ●150 IF PEEK(-16384)>127 THEN GET I$
★■▲●190 IF I$=CHR$(ASC(L$)+N) THEN LET S=S+1
```

How to change the speed

If the game is too quick for you, put a higher number into the middle of line 140 (i.e. to replace 20 or 200). You can speed it up with a lower number.



How to make the game harder

You could change the 1 in lines 40 and 60 to, say, 3 to allow difficulties of only 3 or more.

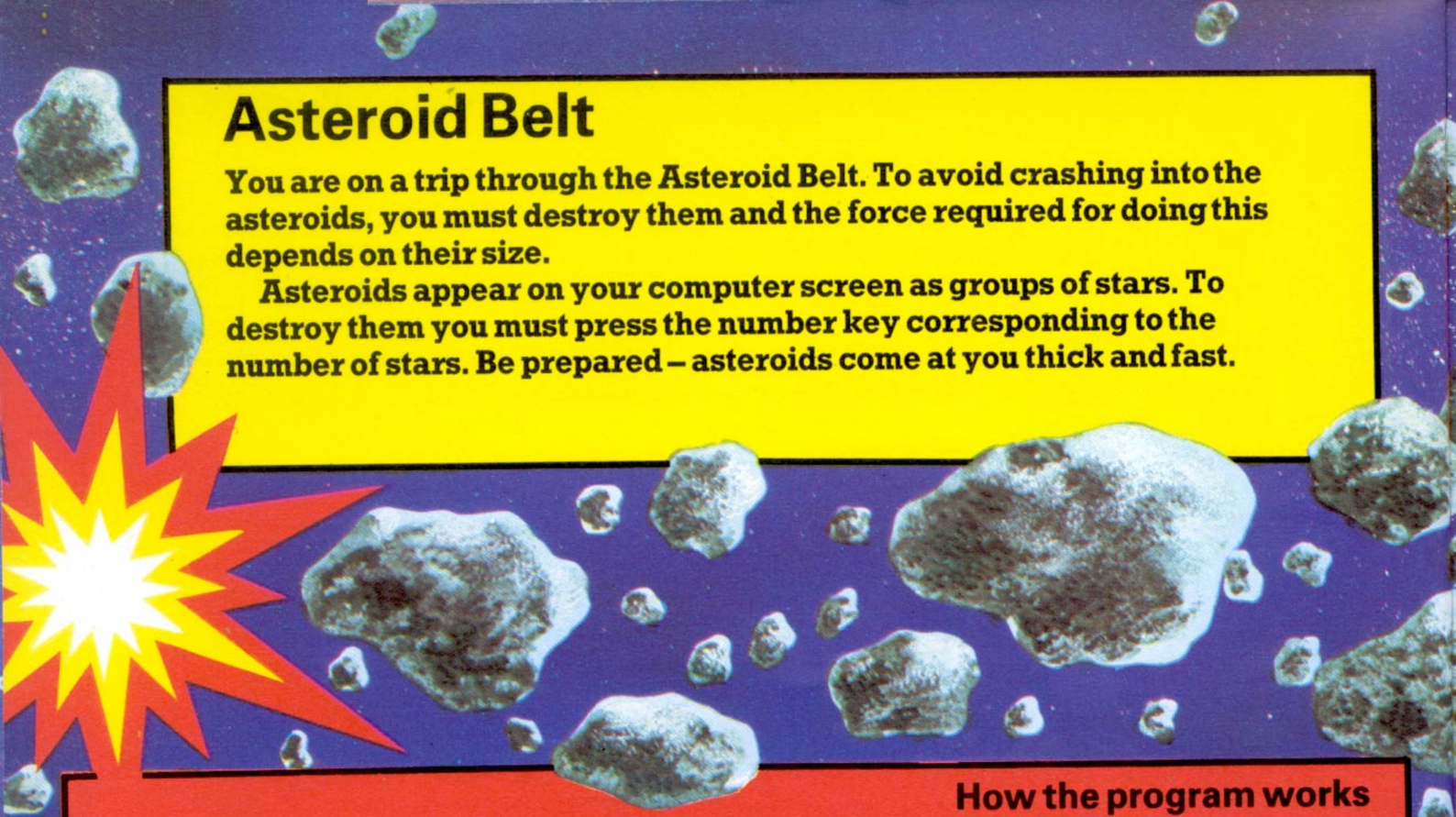
Puzzle corner

Can you adjust the scoring so that it fits the code number i.e. you score 1 point if the code is 1, 2 points if the code is 2 and so on?

Asteroid Belt

You are on a trip through the Asteroid Belt. To avoid crashing into the asteroids, you must destroy them and the force required for doing this depends on their size.

Asteroids appear on your computer screen as groups of stars. To destroy them you must press the number key corresponding to the number of stars. Be prepared – asteroids come at you thick and fast.



How the program works

```
10 PRINT "ASTEROID BELT"
20 LET S=0
30 FOR G=1 TO 10
40 CLS
50 LET A=INT(RND*18+1)
60 LET D=INT(RND*12+1)
70 LET N=INT(RND*9+1)
80 FOR I=1 TO D
90 PRINT
100 NEXT I
110 FOR I=1 TO N
120 IF I<>1 AND I<>4 AND I<>7 THEN GOTO 150
130 PRINT
140 PRINT TAB(A);
150 PRINT "*";
160 NEXT I
170 PRINT
180 FOR I=1 TO 10
190 LET Q=VAL("0"+INKEY$)
200 IF Q<>0 THEN GOTO 240
210 NEXT I
220 PRINT "CRASHED INTO ASTEROID"
230 GOTO 290
240 IF Q<>N THEN GOTO 270
```

Sets the opening score to zero.

Starts a loop which gives 10 goes.

Chooses a number for the position of the asteroid across the screen. Puts this in A.

Chooses a number (1 to 12) for the position of the asteroid down the screen and puts it in D.

Chooses a number (1 to 9) for the number of stars in the asteroid.

Moves the cursor D lines down the screen.

Loops round N times printing a star each time in the appropriate position.

Loops round to see if you are pressing a key and jumps to 240 if you are.

Prints if you run out of time.

Checks if your number is the same as N and jumps to 270 if it isn't.



```
250 PRINT "YOU DESTROYED IT"
260 LET S=S+1
270 IF Q<N THEN PRINT "NOT STRONG ENOUGH"
280 IF Q>N THEN PRINT "TOO STRONG"
★■▲●290 FOR I=1 TO 50
300 NEXT I
310 NEXT G
320 PRINT "YOU HIT ";S;" OUT OF 10"
330 STOP
```

Prints if you pressed the right number.

Increases your score by one.

Compares your number with N and prints an appropriate message.

Delay loop to keep messages on screen long enough for you to read them.

Goes back for another go.

Prints your score after 10 goes.

The above listing will work on a ZX81. For other computers, make the changes below.

```
●40 HOME
▲40 PRINT CHR$(147)
■50 LET A=INT(RND(0)*18+1)
★▲●50 LET A=INT(RND(1)*18+1)
■60 LET D=INT(RND(0)*12+1)
★▲●60 LET D=INT(RND(1)*12+1)
■70 LET N=INT(RND(0)*9+1)
★▲●70 LET N=INT(RND(1)*9+1)
●175 Q=0
★180
■▲●180 FOR I=1 TO 100
▲190 GET Q
●190 IF PEEK(-16384)>127 THEN GET Q
★190 Q=INKEY(100)-48
★210
★■290 FOR I=1 TO 500
▲●290 FOR I=1 TO 250
```

Changing the speed of the game

Line 180 (190 for the BBC) controls how much time you have to press a key. Change the last number in 180, or the number in brackets in the BBC line 190, to a lower number to speed up the game.

Puzzle corner

Can you adapt the scoring system so that, for each asteroid, you get the same number of points as there are stars in it?



See page 2 for meaning of ●▲■★

Trip into the Future



Imagine you are in a spaceship travelling nearly as fast as light. Strangely time is passing more slowly inside your spaceship than outside. So, having set off on a long, fast space trip, you can return to Earth further in the future than the clocks inside your ship indicate.

In this game, your computer tells you how many years must elapse on Earth before you return. You then decide the length of your trip (in light years) and the speed of your ship (as a fraction of the speed of light) in order to achieve this. Take care not to travel too far too slowly or you will die of old age on the way.

```
▲●10 CLS
  20 PRINT "TRIP INTO THE FUTURE"
★■▲●30 LET T=INT(RND*100+25)
  40 PRINT "YOU WISH TO RETURN ";T
  50 PRINT "YEARS IN THE FUTURE."
  60 PRINT
  70 PRINT "SPEED OF SHIP (0-1)"
  80 INPUT V
  90 IF V>=1 OR V<=0 THEN GOTO 70
 100 PRINT "DISTANCE OF TRIP"
 110 INPUT D
 120 LET T1=D/V
```

1K

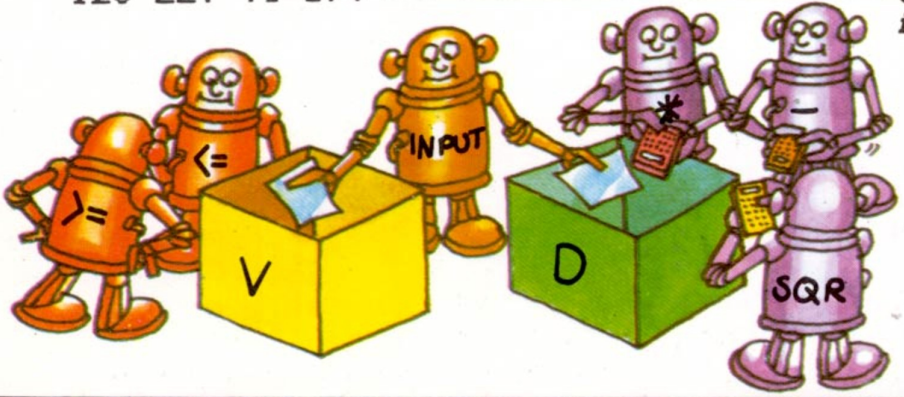
How the program works

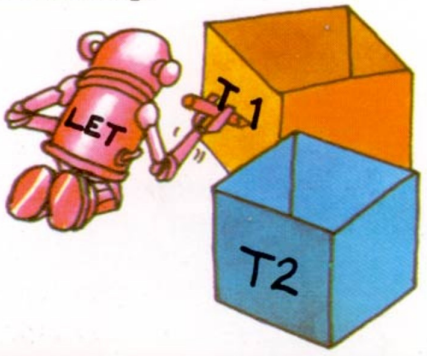
Chooses a whole number between 25 and 124 for the years which must elapse before your return, and prints it.

Gets a speed from you and checks it is within the correct limits.

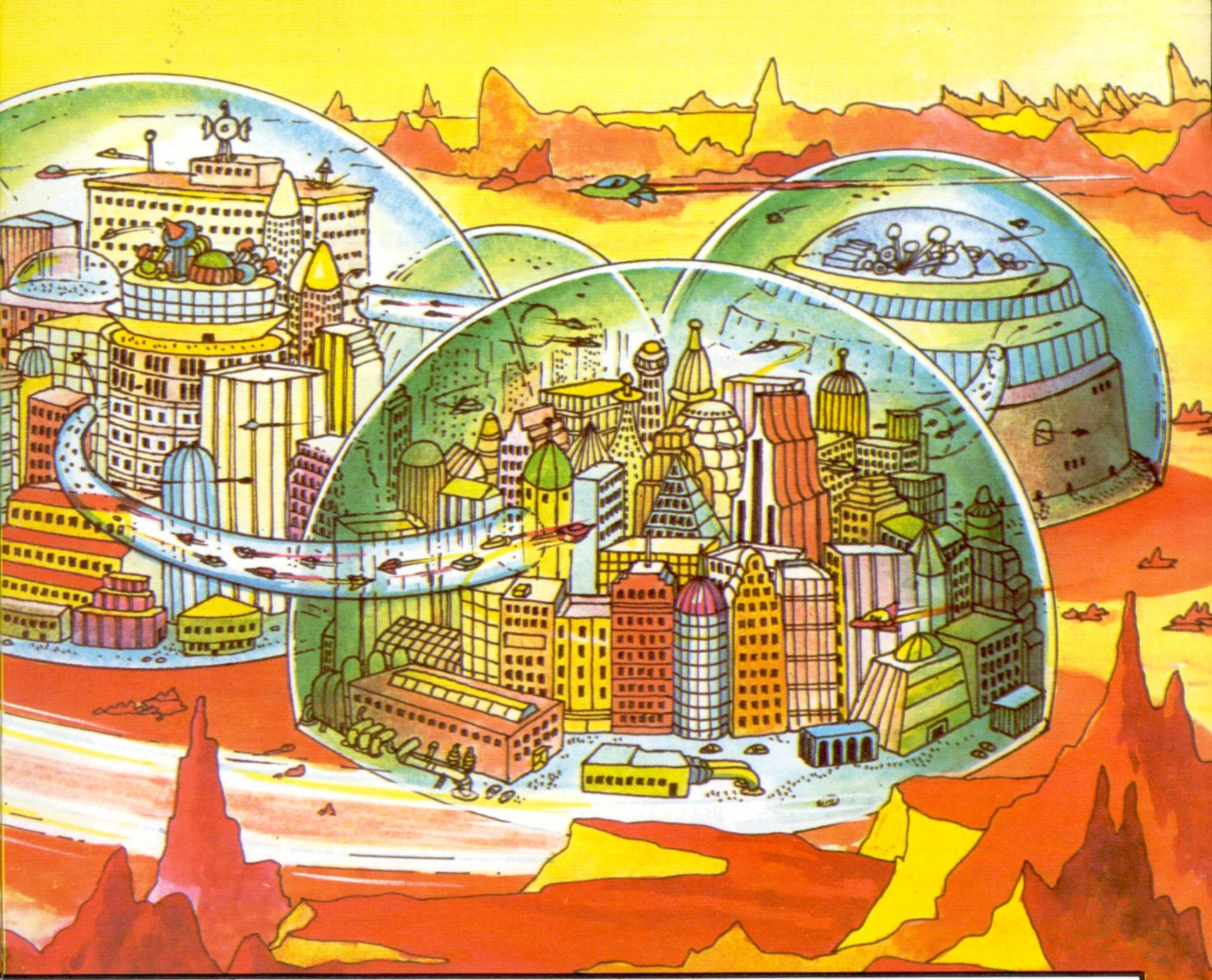
Gets a distance from you.

Calculates the time taken *inside* the ship.





See page 2 for meaning of ●▲■★



```
130 LET T2=T1/SQR(1-V*V)
```
Calculates the time taken *outside* the ship (i.e. on Earth).

```
140 PRINT "YOU TOOK ";T1;"YEARS"
150 PRINT "AND ARRIVED ";T2;"YEARS"
160 PRINT "IN THE FUTURE."
```
Prints these times.

```
170 IF T1>50 THEN GOTO 210
```
Checks if you took longer than your lifetime (50 years). Jumps to line 210 if you did.

```
180 IF ABS(T-T2)<=5 THEN PRINT "YOU ARRIVED ON TIME"
190 IF ABS(T-T2)>5 THEN PRINT "NOT EVEN CLOSE"
200 STOP
210 PRINT "YOU DIED ON THE WAY"
220 STOP
```
Checks if you were within 5 years and prints a message.

The above listing will work on a ZX81. For other computers, make the changes below.

```
●10 HOME
▲10 PRINT CHR$(147)
■30 LET T=INT(RND(0)*100+25)
★▲●30 LET T=INT(RND(1)*100+25)
```

Puzzle corner

Can you work out how to change the program to do the following things?

1) Give a wider range of years which must elapse before you return to Earth.
2) Increase the accuracy required from within 5 years to within 2 years.
3) Shorten or lengthen your lifetime.

Death Valley

There is only one way to escape the forces of the evil Dissectitrons. You will have to steel every nerve and fly your single-seater Speed Dart along the jagged, bottomless ravine known as Death Valley.

Your computer will ask you for the width of the valley. Try 15* first and then work your way down – 8 is quite difficult. Steer your Speed Dart by pressing Q to go left and P to go right, and see if you can make it safely through Death Valley.

***If you are using a VIC 20, then use widths of 6 to 10.**

How the program works

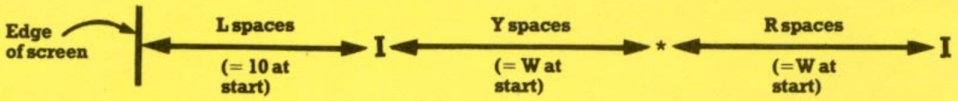
```
10 PRINT "DEATH VALLEY"
20 LET S=0
30 LET M=200
40 PRINT "WIDTH?"
50 INPUT W
60 LET W=INT(W/2)
▲70 LET L=10
80 LET Y=W
90 LET R=W
```

Sets the number of goes used to zero for start of game.

M is the maximum number of goes allowed.

Gets a width number from you, divides it by 2 and uses INT to remove any halves.

L, Y and R are the distances between walls and Speed Dart.

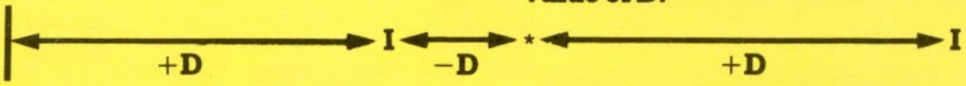



```
★■▲●100 LET D=INT(RND*3-1)
110 IF L+D<0 OR L+D>20 THEN GOTO 100
120 LET L=L+D
130 LET Y=Y-D
140 LET R=R+D
```

Selects −1, 0 or +1 and puts it in D.

Checks L+D isn't so big or small that columns disappear off sides of screen.

Changes spaces between columns according to the value of D.




```
s★■▲●145 SCROLL
150 LET N=L
160 GOSUB 1000
170 PRINT "I";
180 LET N=Y
190 GOSUB 1000
200 PRINT "*";
210 LET N=R
220 GOSUB 1000
230 PRINT "I"
●→
★▲●240 LET I$=INKEY$
250 IF I$<>"Q" THEN GOTO 280
260 LET Y=Y-1
270 LET R=R+1
280 IF I$<>"P" THEN GOTO 310
```

Moves cursor L spaces across the screen and prints I. (The semi-colon then stops the cursor going down to the next line.)

Moves cursor a further Y spaces and prints.*

Moves cursor a further R spaces and prints another I. (No semi-colon this time, so the cursor then goes down to the next line.)

Checks to see if you are pressing a key.

If the key you are pressing is *not* Q, computer jumps to 280.

If key *is* Q then Y is decreased by one and R increased by one so star moves left.

Checks if key being pressed is P.

```
290 LET Y=Y+1
300 LET R=R-1
```

If so, star is moved right by increasing Y by one and decreasing R by one.

```
310 IF Y<1 OR R<1 THEN GOTO 370
```

Checks if you have crashed into a wall. Jumps to 370 to tell you if you have.

```
320 LET S=S+1
```

Increases number of goes used by one.

```
330 IF S<M THEN GOTO 100
```

Goes back for another turn if you have had less than M goes.

```
340 PRINT "WELL DONE-YOU MADE IT"
350 PRINT "THROUGH DEATH VALLEY"
```

Prints if you have had M goes and no crash.

```
360 STOP
370 PRINT "YOU CRASHED INTO THE WALL"
380 PRINT "AND DISINTEGRATED"
390 STOP
```

```
1000 IF N=0 THEN RETURN
1010 FOR I=1 TO N
1020 PRINT " ";
1030 NEXT I
1040 RETURN
```

Sub-routine for moving the cursor to the appropriate places for printing I and *.

The above listing will work on a ZX81. For other computers, make the changes below.

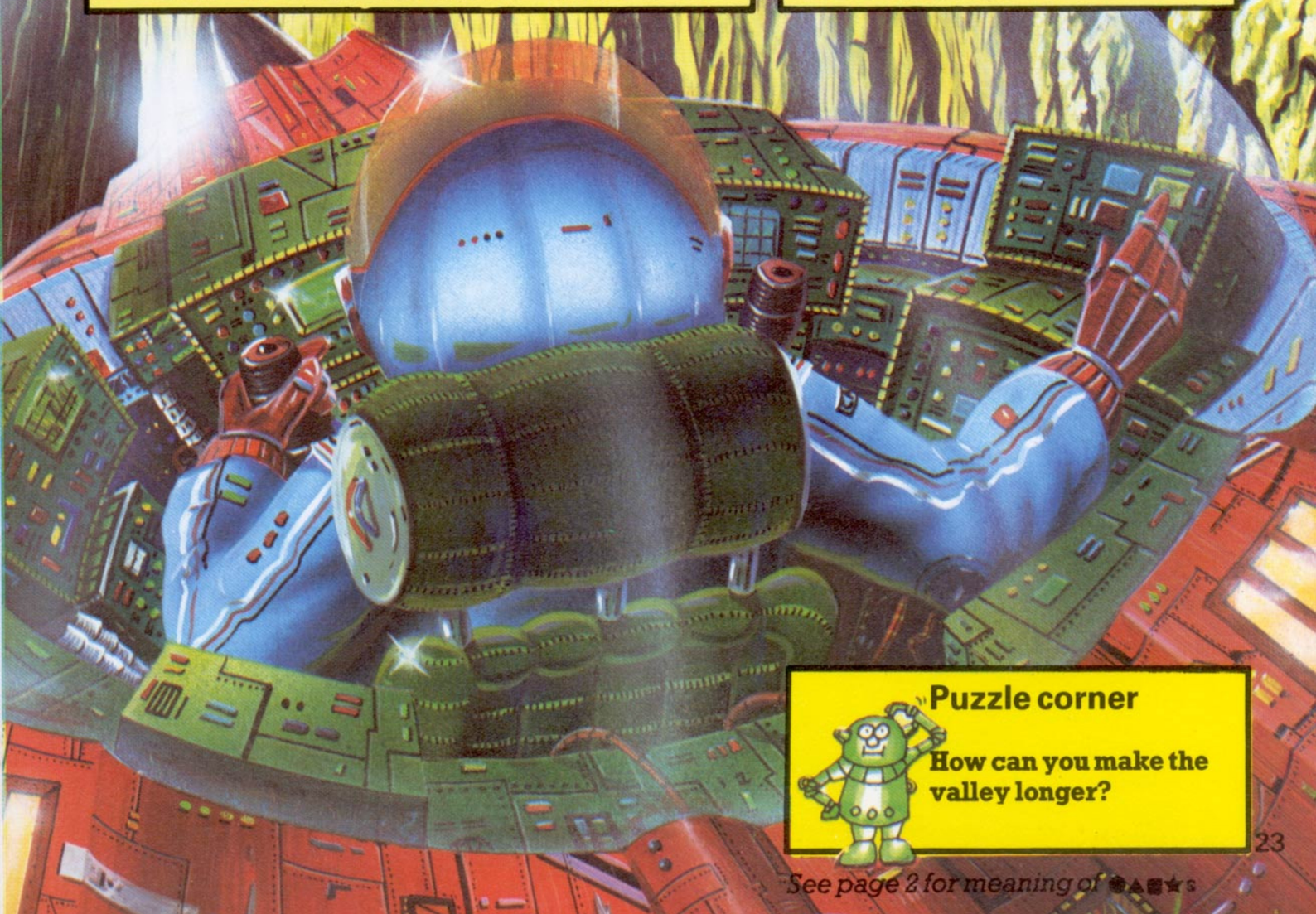
```
▲70 LET L=4
■100 LET D=INT(RND(0)*3-1)
★▲●100 LET D=INT(RND(1)*3-1)
s★■▲●145
●235 I$=""
▲240 GET I$
★240 LET I$=INKEY$(1)
●240 IF PEEK(-16384)>127 THEN GET I$
```

Slowing down the game

If this game is too fast for you, you can add a delay loop at lines 141 and 142:

```
141 FOR J=1 TO 100
142 NEXT J
```

Change the number in line 141 to adjust the speed – the lower the number the faster the game.



Puzzle corner

How can you make the valley longer?

See page 2 for meaning of ●▲■★s

Space Mines

You are the newly elected leader of a mining colony on the Planet Astron. All decisions concerning the sale of ore to Intergalactic Traders, food purchase and sale and purchase of mines are made by you. Can you keep the people satisfied and survive your 10 years in office or will life in the colony end in disaster under your rule?

How the program works

```
★■▲●10 LET L=INT(RND*3+5)
       20 LET P=INT(RND*60+40)
★■▲●   30 LET M=INT(RND*50+10)*P
       40 LET FP=INT(RND*40+80)
       50 LET CE=INT(RND*40+80)
```

These lines decide the number of mines (L), number of people (P), amount of money (M), price of food (FP) and amount of ore produced per mine (CE) for the start of the game.

```
       60 LET C=0
```

Sets the amount of ore in storage to zero for start.

```
       70 LET S=1
```

Sets the satisfaction factor to 1.

```
       80 LET Y=1
```

Sets the year number to 1.

```
★■▲●90 LET LP=INT(RND*2000+2000)
```

Selects buying/selling price for mines.

```
★■▲●100 LET CP=INT(RND*12+7)
```

Selects selling price for ore.

```
  ▲●110 CLS
       120 PRINT "YEAR";Y
       130 PRINT
       140 PRINT "THERE ARE ";P;" PEOPLE IN THE COLONY."
       150 PRINT "YOU HAVE ";L;" MINES,AND $"M
       160 PRINT "SATISFACTION FACTOR ";S
       170 PRINT
       180 PRINT "YOUR MINES PRODUCED ";CE;" TONS EACH"
       190 LET C=C+CE*L
       200 PRINT "ORE IN STORE=";C;" TONS"
       210 PRINT "SELLING"
       220 PRINT "ORE SELLING PRICE=";CP
       230 PRINT "MINE SELLING PRICE=";LP;"/MINE"
```

Prints current state of affairs in the colony.

```
       240 PRINT "HOW MUCH ORE TO SELL?"
       250 INPUT CS
       260 IF CS<0 OR CS>C THEN GOTO 240
```

Asks how much ore you want to sell, puts it CS and checks you've got that much in store.

```
       270 LET C=C-CS
```

Takes amount sold away from amount in store.

```
       280 LET M=M+CS*CP
```

Works out how much ore sold is worth and adds this to your money supply.

```
       290 PRINT "HOW MANY MINES TO SELL?"
       300 INPUT LS
       310 IF LS<0 OR LS>L THEN GOTO 290
       320 LET L=L-LS
       330 LET M=M+LS*LP
```

Does the same process for selling mines.

```
       340 PRINT
       350 PRINT "YOU HAVE $";M
       360 PRINT
```

Prints your new money supply.

```
       370 PRINT "BUYING"
       380 PRINT "HOW MUCH TO SPEND ON FOOD ? (APPR. $100 EA.) "
       390 INPUT FB
```

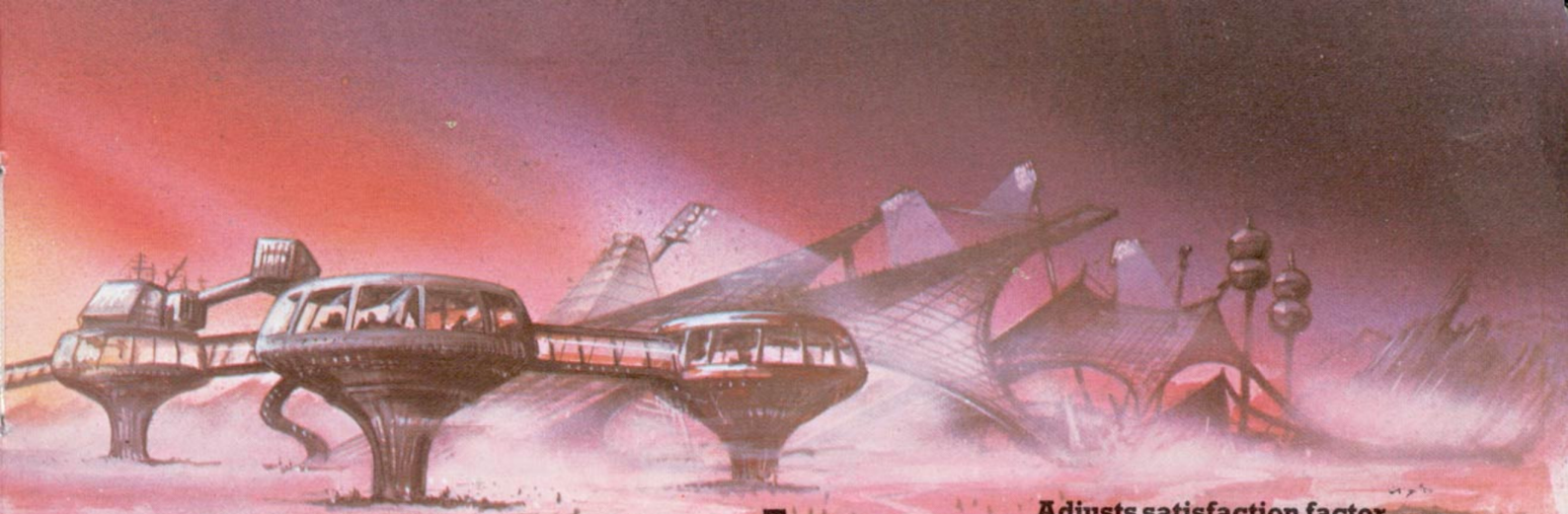
Asks how much you want to spend on food and puts this in FB.

```
       400 IF FB<0 OR FB>M THEN GOTO 380
```

Checks you have enough money to pay.

```
       410 LET M=M-FB
```

Adjusts your money supply.



```
420 IF FB/P>120 THEN LET S=S+.1
430 IF FB/P<80 THEN LET S=S-.2
```

Adjusts satisfaction factor according to how much you spent on food.

```
440 PRINT "HOW MANY MORE MINES TO BUY?"
450 INPUT LB
460 IF LB<0 OR LB*LP>M THEN GOTO 440
```

Asks how many mines you want to buy and checks you can afford them.

```
470 LET L=L+LB
```

Increases number of mines if necessary.

```
480 LET M=M-LB*LP
```

Adjusts your money again.

```
490 IF S<.6 THEN GOTO 660
```

Checks on value of satisfaction factor. If this is very low, computer jumps to 660 to end the game.

```
★■▲●500 IF S>1.1 THEN LET CE=CE+INT(RND*20+1)
```

If S is high then amount produced per mine is increased.

```
★■▲●510 IF S<.9 THEN LET CE=CE-INT(RND*20+1)
```

If S low, amount produced is decreased.

```
520 IF P/L<10 THEN GOTO 680
```

If there are less than 10 people per mine, game is over.

```
★■▲●530 IF S>1.1 THEN LET P=P+INT(RND*10+1)
```

More people arrive if S is high.

```
★■▲●540 IF S<.9 THEN LET P=P-INT(RND*10+1)
```

People leave if S is low.

```
550 IF P<30 THEN GOTO 700
```

If there are less than 30 people, game is over.

```
★■▲●560 IF RND>.01 THEN GOTO 590
570 PRINT "RADIOACTIVE LEAK....MANY DIE"
580 LET P=INT(P/2)
```

Introduces small chance of half the people being killed.

```
590 IF CE<150 THEN GOTO 620
600 PRINT "MARKET GLUT - PRICE DROPS"
610 LET CE=INT(CE/2)
```

If the amount produced per mine is very high, ore price is halved.

```
620 LET Y=Y+1
```

Year number increased by 1 and if it is less than 11, computer goes back to 90 for another go.

```
630 IF Y<11 THEN GOTO 90
640 PRINT "YOU SURVIVED YOUR TERM OF OFFICE"
650 STOP
660 PRINT "THE PEOPLE REVOLTED"
670 STOP
680 PRINT "YOU'VE OVERWORKED EVERYONE"
690 STOP
700 PRINT "NOT ENOUGH PEOPLE LEFT"
710 STOP
```

Prints if the computer reaches this stage in the program on your 10th go.

The above listing will work on a ZX81. For other computers, make the changes below.

```
●110 HOME
▲110 PRINT CHR$(147)
★▲●10,20,30,40,50,90,100,500,510,530,540,560
   change RND to RND(1)
■10,20,30,40,50,90,100,500,510,530,540,560
   change RND to RND(0)
```

See page 2 for meaning of ●▲■★



Puzzle corner

Can you make the computer ask if you would like another game and add the money you have ended up with to the new money supply for the next game?

Space Rescue

You must make an urgent trip across the spiral arm of the Galaxy to a developing planet which is in need of medical supplies. The trip involves such huge distances that for most of it you will be in a deep sleep, but before this you must program the ship for the journey. The computer will ask how much energy you want to allocate to the engines, life support system and shields and then put you to sleep.

When you wake up, it will give you a report on what happened during the trip and, if all went well, you will be orbiting the planet. You must now allocate your remaining energy to the landing boosters and shields in order to make a good landing on the planet.

If you accomplish the mission safely, you stand a good chance of being promoted to Space Admiral. Good luck!

```
▲●10 CLS
  20 PRINT "SPACE RESCUE"
  30 PRINT
  40 PRINT "DO YOU WANT INSTRUCTIONS? "
  50 INPUT I$
★■▲●60 IF I$(1)="Y" THEN GOSUB 1000
★■▲●70 LET D=INT(RND*800+101)
★■▲●80 LET E=INT(RND*400+401)
  90 LET T=INT(D/SQR(E/5)+.5)
 100 PRINT "THE PLANET IS ";D;" UNITS AWAY"
 110 PRINT "YOU HAVE ";E;" UNITS OF ENERGY"
 120 PRINT "AND A TIME LIMIT OF ";T;" DAYS"
 130 PRINT
 140 PRINT "ENERGY DISTRIBUTION:"
 150 PRINT "TO ENGINES?"
 160 INPUT P
 170 PRINT "TO LIFE SUPPORT?"
 180 INPUT L
 190 PRINT "TO SHIELDS?"
 200 INPUT S
 210 IF P+L+S>E THEN GOTO 140
 220 LET X=E-P-L-S
 230 LET V=INT(SQR(P))
 240 LET T1=INT(D/V)
▲●250 CLS
 260 PRINT "YOUR VELOCITY IS ";V
 270 PRINT "YOU HAVE AN ETA OF ";T1;" DAYS"
 280 PRINT
★■▲●290 FOR I=1 TO INT(RND*5+6)
★■▲●300 IF RND>.5 THEN GOTO 430
★■▲●310 GOTO 320+INT(RND*4)*30
 320 PRINT "ASTEROID STORM - SHIELDS DAMAGED"
```

```
★■▲●330 LET S=S-20-INT(RND*40+1)
   340 GOTO 430
   350 PRINT "COMPUTER BREAKDOWN - DELAY IN REPAIRING"
★■▲●360 LET D=D+INT(RND*20+1)
   370 GOTO 430
   380 PRINT "ENGINE TROUBLE - MUST SLOW DOWN"
   390 LET V=V-.5
   400 GOTO 430
   410 PRINT "X-RAY DAMAGE - LIFE SUPPORT DAMAGED"
   420 LET L=L-20-INT(RND*40+1)
★■▲●430 FOR J=1 TO 50
   440 NEXT J
   450 NEXT I
   460 LET T1=INT(D/V)
  ▲●470 CLS
   480 PRINT "ARRIVED IN ";T1;" DAYS"
   490 IF S<0 THEN PRINT "SHIELDS DESTROYED"
   495 IF S<0 THEN PRINT "YOU WERE BLOWN UP"
   500 IF L<=0 THEN PRINT "LIFE SUPPORT INACTIVE"
   505 IF L<=0 THEN PRINT "YOU'RE DEAD"
   510 IF V<=0 THEN PRINT "ENGINES ARE NON-FUNCTIONAL"
   520 IF T1>T THEN PRINT "YOU TOOK TOO LONG ABOUT IT"
   530 IF S<0 OR L<=0 OR V<=0 OR T1>T THEN STOP
★■▲●540 LET G=INT(RND*10+5)
   550 LET G$="HIGH"
   560 IF G<12 THEN LET G$="MEDIUM"
   570 IF G<8 THEN LET G$="LOW"
★■▲●580 LET A=INT(RND*10+5)
   590 LET A$="HIGH"
   600 IF A<12 THEN LET A$="MEDIUM"
   610 IF A<8 THEN LET A$="LOW"
   620 PRINT
   630 PRINT "YOU ARE NOW ORBITING THE PLANET"
   640 PRINT "SURPLUS ENERGY=";X
   650 PRINT "GRAVITY IS ";G$
   660 PRINT "ATMOSPHERE IS ";A$
   670 PRINT
   680 PRINT "HOW MUCH ENERGY TO BOOSTERS?"
   690 INPUT B
   700 PRINT "HOW MUCH ENERGY TO HEAT SHIELDS?"
   710 INPUT S
   720 IF B+S>X THEN GOTO 680
  ▲●730 CLS
   740 IF B>=G*10 THEN GOTO 770
```

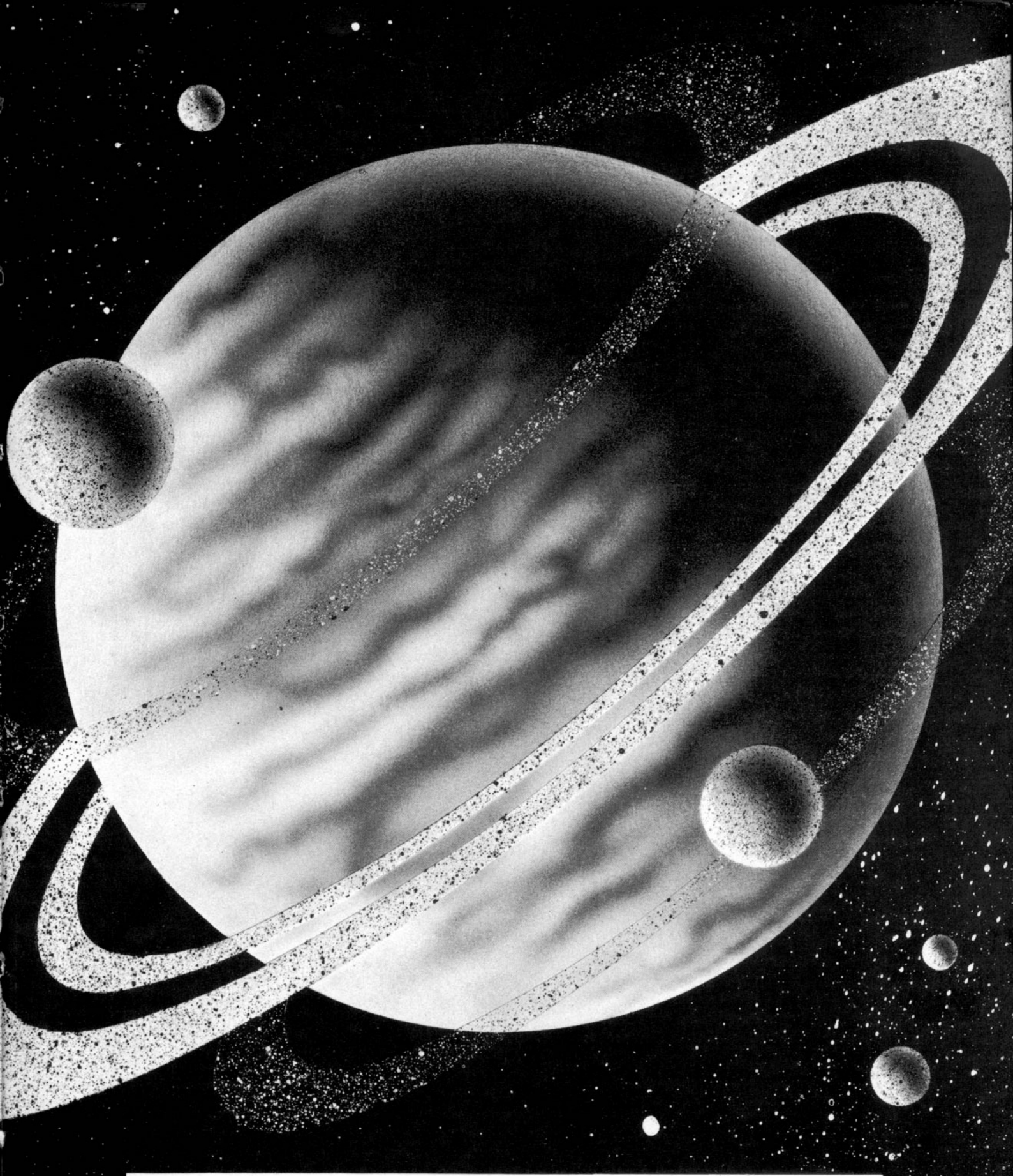
Program continued on next page.

Space Rescue continued

```
  750 PRINT "YOU MADE A NEW CRATER"
  760 GOTO 840
  770 IF S>=A*10 THEN GOTO 800
  780 PRINT "YOU MADE A WONDERFUL SHOOTING STAR"
  790 GOTO 840
  800 PRINT "YOU LANDED SUCCESSFULLY - WELL DONE"
  810 IF X-S-B>25 THEN GOTO 840
  820 PRINT "PITY YOU DON'T HAVE ENOUGH"
  830 PRINT "ENERGY TO OPEN THE DOOR"
  840 STOP
 1000 PRINT
 1010 PRINT "YOU ARE ABOUT TO EMBARK ON A"
 1020 PRINT "MISSION TO A DISTANT PLANET"
 1030 PRINT "IN URGENT NEED OF MEDICAL"
 1040 PRINT "SUPPLIES. YOU MUST FIRST READY"
 1050 PRINT "YOUR SHIP FOR THE TRIP BY"
 1060 PRINT "ALLOCATING SOME OF THE SHIP'S"
 1070 PRINT "ENERGY TO THE ENGINES,SHIELDS"
 1080 PRINT "AND LIFE-SUPPORT. YOU ARE"
 1090 PRINT "THEN PUT TO SLEEP FOR THE MAIN"
 1100 PRINT "PART OF THE TRIP, AFTER WHICH"
 1110 PRINT "YOU WILL GET A REPORT OF THE"
 1120 PRINT "EVENTS ON THE WAY. YOU MUST"
 1130 PRINT "THEN LAND ON THE PLANET....."
 1140 PRINT "PRESS ANY KEY"
★■▲●1150 IF INKEY$="" THEN GOTO 1150
 ▲●1160 CLS
 1170 RETURN
```

The above listing will work on a ZX81. For other computers, make the changes below.

```
  ■all RND to RND(0)
 ★▲●all RND to RND(1)
  ●10,250,470,730,1160 HOME
  ▲10,250,470,730,1160 PRINT CHR$(147)
★■▲●60 IF LEFT$(I$,1)="Y" THEN GOSUB 1000
★■▲●310 ON INT(RND*4+1) GOTO 320,350,380,410
 ■▲●430 FOR J=1 TO 500
  ★430 FOR J=1 TO 1000
  ★1150 I=GET
  ●1150 GET I$
 ■▲1150 GET I$ : IF I$="" THEN GOTO 1150
```



Adding to the game

This game is really made up of two parts. In the first part you set off on your space journey with the aim of orbiting the planet and in the second you attempt to land on the planet. You could perhaps try adding a third part in which you make the treacherous crossing from the landing site to the Intergalactic Red Cross H.Q.

Touchdown

This game is different from the others in this book because it uses graphics. As the computers vary so much in the way their graphics work, there is a separate program for each one. Read the instructions on this page for how to play the game and then look through the pages that follow for the version for your computer.



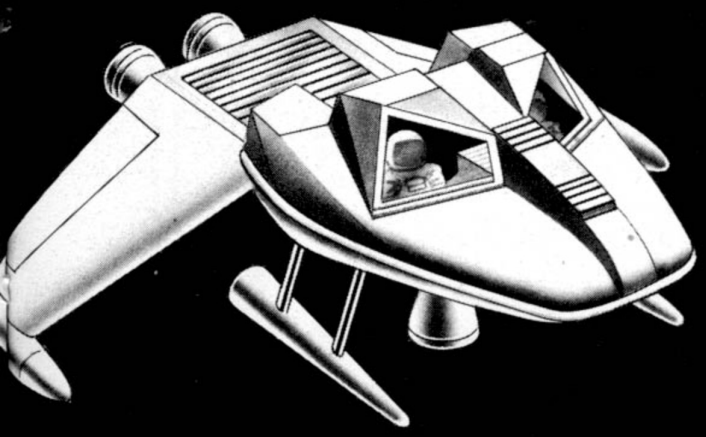
How to play Touchdown

Ace space pilot, Captain Flash, is sitting next to you as you take the final part of your Advanced Spacecraft Handling Test (Part III). Your lightweight, two-man landing craft is rapidly approaching the Moon's surface. Your velocity must be almost zero as you touch down. Deftly you control the thrust, pressing A to increase it and D to decrease it*, watching your progress on the screen all the time. If you use too much thrust you will begin to go back up again. Too little and you will make a new crater on the Moon. Can you impress Captain Flash with your skill?

***For VIC, use the cursor down key to increase thrust and the cursor right key to decrease it.**

Touchdown: TRS-80 version

```
20 CLS
30 CLEAR 200
31 B$=STRING$(25,131)
33 M1$=CHR$(194)+STRING$(2,176)
34 M2$=" "+STRING$(4,191)
35 M3$=CHR$(131)+CHR$(135)+STRING$
   (2,131)+CHR$(139)+CHR$(131)
40 GOSUB 250
50 GOSUB 300
60 C=1:GOSUB 390
70 A=1:B=F:GOSUB 460
80 A=2:B=ABS(V):GOSUB 460
90 A=3:B=H:GOSUB 460
100 A=4:B=T:GOSUB 460
110 GOSUB 530
120 V1=V-T/20+G : F=F-T/10
130 H1=H-(V+V1)/10
140 C=0:GOSUB 390
150 IF H1<0 THEN 200
160 H=H1:V=V1
170 IF H<=100 THEN 60
180 GOSUB 590
190 GOTO 220
200 H=0:C=1:GOSUB 390
210 GOSUB 660
220 END
250 H=100:F=100:T=0
260 V=INT(RND(0)*10+6)
270 G=INT(RND(0)*40+41)/100
280 RETURN
300 FOR X=80 TO 127
320 SET (X,47-INT(RND(0)*5))
330 NEXT
340 PRINT "GRAVITY=";G
350 PRINT @192,"FUEL:"
355 PRINT @384,"VEL:"
360 PRINT @576,"HEIGHT:"
365 PRINT @768,"THRUST:"
370 RETURN
390 Y=818-64*INT(H/8)
400 PRINT @Y,; : IF C=1 THEN PRINT
    M1$; ELSE PRINT CHR$(196);
410 PRINT @Y+64,; : IF C=1 THEN
    PRINT M2$; ELSE PRINT CHR$
    (198);
420 PRINT @Y+128,; : IF C=1 THEN
    PRINT M3$; ELSE PRINT CHR$
    (198)
440 RETURN
460 Y=(A*3+1)*64
470 PRINT @Y,CHR$(217);
480 PRINT @Y,LEFT$(B$,B/4);
510 RETURN
530 I$=INKEY$
540 IF I$="A" THEN T=T+4 : IF
    T>100 THEN T=100
550 IF I$="D" THEN T=T-4 : IF
    T<0 THEN T=0
560 IF T>F THEN T=F
570 RETURN
590 CLS
600 FOR I=1 TO 20
610 PRINT @INT(RND(0)*1024),"*"
620 NEXT
630 PRINT @470,"LOST IN SPACE!!"
640 RETURN
650 CLS
660 PRINT "LANDED AT VEL ";
    INT((V+V1)*5)/10
670 IF (V+V1)<8 THEN PRINT "SAFELY"
    ELSE PRINT "ALL DEAD"
680 RETURN
```



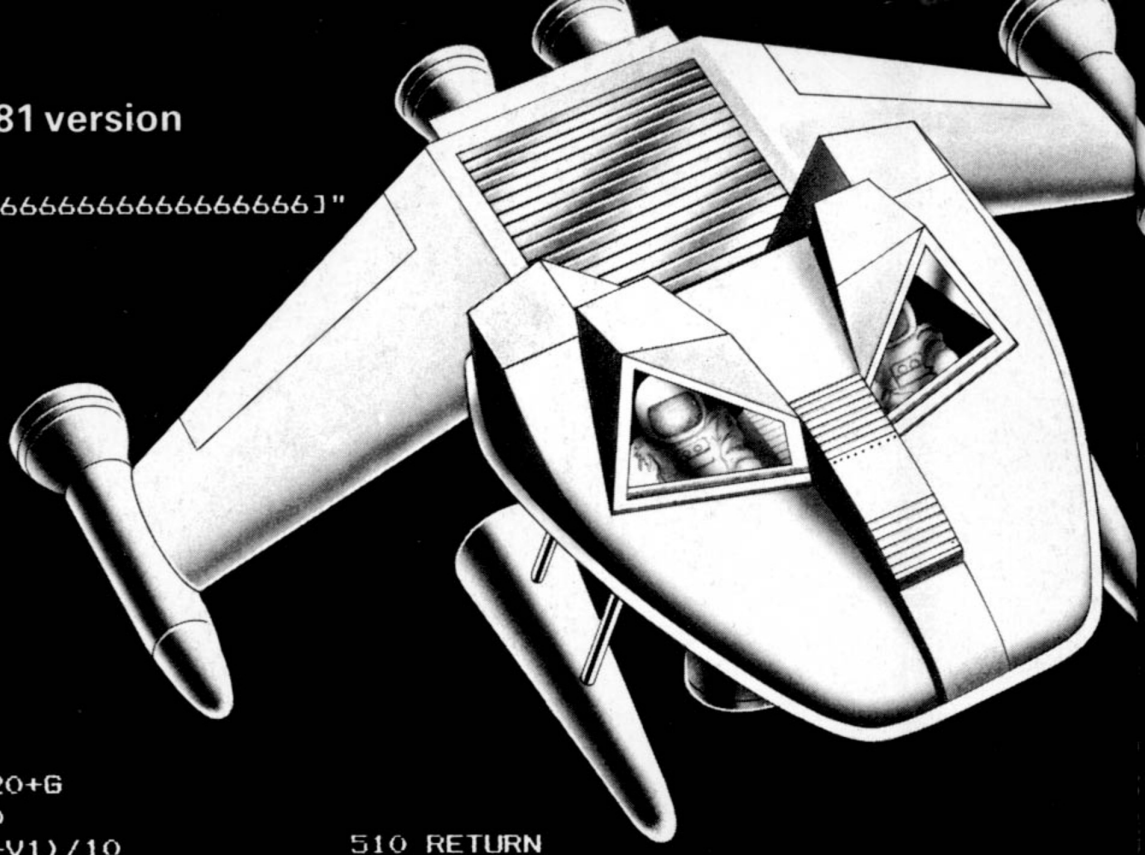
Touchdown: VIC 20 version

```
20 PRINT CHR$(147)CHR$(5);
25 POKE 36879,8
30 DEF FNR(X)=INT(RND(1)*X+1)
40 GOSUB 250
50 GOSUB 300
60 C=1:GOSUB 390
70 A=1:B=F:GOSUB 460
80 A=2:B=ABS(V):GOSUB 460
90 A=3:B=H:GOSUB 460
100 A=4:B=T:GOSUB 460
110 GOSUB 530
120 V1=V-T/20+G : F=F-T/10
130 H1=H-(V+V1)/10
140 C=0:GOSUB 390
150 IF H1<0 THEN 200
160 H=H1:V=V1
170 IF H<=100 THEN 60
180 GOSUB 590
190 GOTO 220
200 H=0:C=1:GOSUB 390
210 GOSUB 660
220 END
250 H=100:F=100:T=0
260 V=5+FNR(10)
270 G=(FNR(40)+40)/100
280 RETURN
300 FOR X=8178 TO 8185
320 POKE X,98+2*FNR(3)
330 NEXT
340 PRINT "GRAVITY=";G
350 PRINT "QQFUEL:"
355 PRINT "QQQVEL:"
360 PRINT "QQQHEIGHT:"
365 PRINT "QQQTHRUST:"
370 RETURN
390 Y=8137-22*INT(H/5)
400 IF C=0 THEN 425
405 POKE Y,108 : POKE Y+1,123
410 POKE Y+22,160 : POKE Y+23,160
415 POKE Y+44,75 : POKE Y+45,74
420 GOTO 440
425 FOR Z=0 TO 44 STEP 22
430 POKE Y+Z,32 : POKE Y+Z+1,32
435 NEXT
440 RETURN
460 FOR X=0 TO 9
470 Y=A*88+X+7724
480 IF X<B/10 THEN POKE Y,102 :
    GOTO 500
485 IF X<B/10+.5 THEN POKE Y,92 :
    GOTO 500
490 POKE Y,32
500 NEXT
510 RETURN
530 GET I$
540 IF I$="Q" THEN T=T+4 : IF T>100
    THEN T=100
550 IF I$="]" THEN T=T-4 : IF T<0
    THEN T=0
560 IF T>F THEN T=F
570 RETURN
590 PRINT CHR$(147)
600 FOR I=1 TO 20
610 POKE 7679+FNR(506),42
620 NEXT
630 PRINT "LOST IN SPACE!!"
640 RETURN
650 PRINT CHR$(147)"LANDED"
660 PRINT "AT VEL ";INT((V+V1)*5)/10
670 IF (V+V1)<8 THEN PRINT "SAFELY"
    :RETURN
680 PRINT "ALL DEAD":RETURN
```

Q is cursor down key

] is cursor right key

Touchdown: ZX81 version



```
20 CLS
30 LET B$="[6666666666666666666]"
40 GOSUB 250
50 GOSUB 300
60 LET C=1
65 GOSUB 390
70 LET A=1
75 LET B=F
77 GOSUB 460
80 LET A=2
85 LET B=ABS(V)
87 GOSUB 460
90 LET A=3
95 LET B=H
97 GOSUB 460
100 LET A=4
105 LET B=T
107 GOSUB 460
110 GOSUB 530
120 LET V1=V-T/20+G
125 LET F=F-T/10
130 LET H1=H-(V+V1)/10
140 LET C=0
145 GOSUB 390
150 IF H1<0 THEN GOTO 200
160 LET H=H1
165 LET V=V1
170 IF H<=100 THEN GOTO 60
180 GOSUB 590
190 GOTO 220
200 LET H=0
205 LET C=1
207 GOSUB 390
210 GOSUB 660
220 STOP
250 LET H=100
252 LET F=100
255 LET T=0
260 LET V=5+INT(RND*10+1)
270 LET G=INT(RND*40+40)/100
280 RETURN
300 PRINT AT 21,20;"[SDADSADAS]"
340 PRINT AT 0,0;"GRAVITY=";G
350 PRINT AT 3,0;"FUEL:"
355 PRINT AT 7,0;"VEL:"
360 PRINT AT 11,0;"HEIGHT:"
365 PRINT AT 15,0;"THRUST:"
370 RETURN
390 LET Y=(100-H)/6+1
400 IF C=0 THEN GOTO 425
405 PRINT AT Y,24;"[34]"
410 PRINT AT Y+1,23;"[Q  W]"
415 PRINT AT Y+2,23;"[TGGY]"
420 GOTO 440
425 PRINT AT Y,24;"  "
430 PRINT AT Y+1,23;"    "
435 PRINT AT Y+2,23;"    "
440 RETURN
460 LET Y=A*4
470 PRINT AT Y,0;"                    "
480 PRINT AT Y,0;B$( TO B/5)
510 RETURN
530 LET I$=INKEY$
540 IF I$="A" THEN LET T=T+4
545 IF T>100 THEN LET T=100
550 IF I$="D" THEN LET T=T-4
555 IF T<0 THEN LET T=0
560 IF T>F THEN LET T=F
570 RETURN
590 CLS
600 FOR I=1 TO 20
610 PRINT AT INT(RND*22),
    INT(RND*32);"*"
620 NEXT I
630 PRINT AT 10,10;"LOST IN SPACE"
640 RETURN
650 CLS
660 PRINT AT 20,0;"LANDED AT VEL "
    ;INT((V+V1)*5)/10
670 IF (V+V1)<8 THEN PRINT "SAFELY"
675 IF (V+V1)>=8 THEN PRINT
    "ALL DEAD"
680 RETURN
```

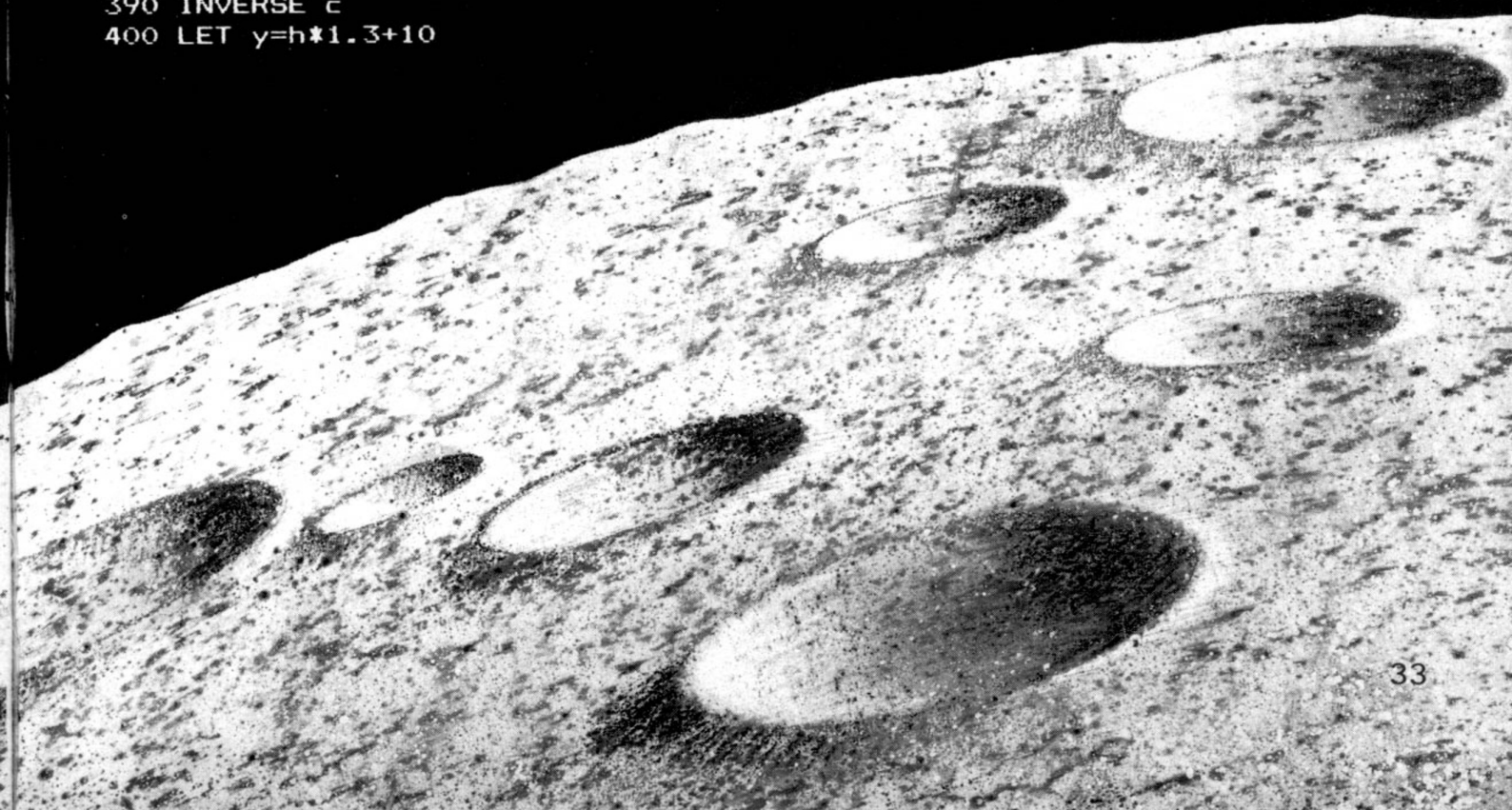
(20 SPACES)

Special note for ZX81 users

Graphic characters are shown in the listing using the following convention. A '[' character means press the GRAPHICS key, and then type the following letters/numbers SHIFTED until ']' is reached, when you type the GRAPHICS key again, and go back to normal.

Touchdown: ZX Spectrum version

```
20 CLS
30 DEF FNr(x)=INT(RND*x+1)
40 GOSUB 250
50 GOSUB 300
60 LET c=0: GOSUB 390
70 LET a=1: LET b=f: LET c=2*(f<25)
75 GOSUB 460
80 LET a=2: LET b=ABS v:
   LET c=4*(v<0)
85 GOSUB 460
90 LET a=3: LET b=h: LET c=2*(h<25)
95 GOSUB 460
100 LET a=4: LET b=t: LET c=0
105 GOSUB 460
110 GOSUB 530
120 LET v1=v-t/20+g: LET f=f-t/10
130 LET h1=h-(v+v1)/10
140 LET c=1: GOSUB 390
150 IF h1<0 THEN GOTO 200
160 LET h=h1: LET v=v1
170 IF h<=100 THEN GOTO 60
180 GOSUB 590
190 GOTO 220
200 LET h=0: LET c=0: GOSUB 390
210 GOSUB 650
220 STOP
250 LET h=100: LET f=100: LET t=0
260 LET v=5+FNr(10)
270 LET G=(FNr(40)+40)/100
280 RETURN
300 PLOT 180,8
310 FOR x=1 TO 15
320 DRAW 5,FNr(3)-2
330 NEXT x
340 PRINT "Gravity=";g
350 PRINT '''"Fuel:"''''"Vel:"
360 PRINT ''''"Height:"''''"Thrust:"
370 RETURN
390 INVERSE c
400 LET y=h*1.3+10
410 PLOT 200,y: DRAW 34,0
420 DRAW -4,20: DRAW -13,10
430 DRAW -13,-10: DRAW -4,-20
440 RETURN
460 LET y=172-a*32
470 INK c
480 PLOT 0,y
490 DRAW b,0
500 DRAW INVERSE 1,100-b,0
510 RETURN
530 LET i$=INKEY$
540 IF i$="a" THEN LET t=t+4 :
    IF t>100 THEN LET t=100
550 IF i$="d" THEN LET t=t-4 :
    IF t<0 THEN LET t=0
560 IF t>f THEN LET t=f
570 RETURN
590 CLS
600 FOR i=1 TO 20
610 PRINT AT FNr(21),FNr(31);"*"
620 NEXT i
630 PRINT "Lost in space!!!"
640 RETURN
650 PRINT AT 0,0;"Landed at ";
    INT((v+v1)*5)/10'
660 IF (v+v1)<8 THEN GOTO 680
670 PRINT "All dead": RETURN
680 PRINT "Safely": RETURN
```



Touchdown: BBC version

```
20 MODE 5
30 *FX 12,1
40 PROCSETVAR
50 PROCDISPLAY
60 PROCMODULE(H,3)
70 PROCBAR(1,F,3+2*(F<25))
80 PROCBAR(2,ABSV,3+(V<0))
90 PROCBAR(3,H,3+2*(H<25))
100 PROCBAR(4,T,3)
110 PROCTHRUST
120 V1=V-T/20+G : F=F-T/10
130 H1=H-(V+V1)/10
140 PROCMODULE(H,0)
150 IF H1<0 THEN 200
160 H=H1:V=V1
170 IF H<=100 THEN 60
180 PROCLOST
190 GOTO 220
200 PROCMODULE(0,2)
210 PROCLANDED
220 *FX 12
230 END
240 DEF PROCSETVAR
250 H=100:F=100:T=0
260 V=5+RND(10)
270 G=(RND(40)+40)/100
280 ENDPROC
290 DEF PROCDISPLAY
300 MOVE 800,30
310 FOR X=800 TO 1280 STEP 16
320 DRAW X,10+RND(40)
330 NEXT
340 PRINT "GRAVITY=";G
350 PRINT ''''"FUEL:"''''''"VEL:"
360 PRINT '''''"HEIGHT:"''''''"THRUST:"''
370 ENDPROC
380 DEF PROCMODULE(H,C)
390 GCOL 0,C
400 Y=H*8.5+150
410 MOVE 1040,Y : PLOT 1,-40,-40
420 PLOT 1,-8,-60 : PLOT 1,96,0
430 PLOT 1,-8,60 : PLOT 1,-40,40
440 ENDPROC
450 DEF PROCBAR(N,V,C)
460 Y=1000-192*N
470 GCOL 0,C
480 MOVE 0,Y : MOVE 0,Y-16
490 PLOT 85,V*4,Y : PLOT 85,V*4,Y-16
500 PLOT 87,400,Y : PLOT 87,400,Y-16
510 ENDPROC
520 DEF PROCTHRUST
530 *FX 15,1
540 IF INKEY(-194) THEN T=T+4 :
    IF T>100 THEN T=100
550 IF INKEY(-179) THEN T=T-4 :
    IF T<0 THEN T=0
560 IF T>F THEN T=F
570 ENDPROC
580 DEF PROCLOST
590 CLS
600 FOR I=1 TO 20
610 VDU 31,RND(19),RND(31),42
620 NEXT
630 PRINT TAB(4,16)"LOST IN SPACE!!"
640 ENDPROC
650 DEF PROCLANDED
660 VDU 28,0,31,11,0,12
670 PRINT '"LANDED"
680 PRINT '"AT VEL ";INT((V+V1)*5)/10
690 IF (V+V1)<8 THEN PRINT '"SAFELY"
    ELSE PRINT '"ALL DEAD"
700 ENDPROC
```



Touchdown: Apple version

```
15 HOME
20 HGR
30 DEF FNR(X)=INT(RND(1)*X+1)
40 GOSUB 250
50 GOSUB 300
60 C=3:GOSUB 390
70 A=1:B=F:GOSUB 460
80 A=2:B=ABS(V):GOSUB 460
90 A=3:B=H:GOSUB 460
100 A=4:B=T:GOSUB 460
110 GOSUB 530
120 V1=V-T/20+G : F=F-T/10
130 H1=H-(V+V1)/10
140 C=0:GOSUB 390
150 IF H1<0 THEN 200
160 H=H1:V=V1
170 IF H<=100 THEN 60
180 GOSUB 590
190 GOTO 220
200 H=0:C=3:GOSUB 390
210 GOSUB 660
220 END
250 H=100:F=100:T=0
260 V=5+FNR(10)
270 G=(FNR(40)+40)/100
280 RETURN
300 HCOLOR=3
305 HPLOT 0,155
310 FOR X=0 TO 279 STEP 5
320 HPLOT TO X,159-FNR(10)
330 NEXT
335 FOR I=1 TO 30 :HPLOT FNR(279),
    FNR(150)
337 NEXT
340 VTAB 21 : PRINT TAB(34);"G=";G
350 VTAB 21 : PRINT "FUEL:" :
    PRINT "VEL:"
360 PRINT "HEIGHT:" : PRINT "THRUST:";
370 RETURN
390 HCOLOR=C
400 Y=(100-H)*1.3
410 HPLOT 140,Y TO 120,Y+10
420 HPLOT TO 120,Y+20 :
    HPLOT TO 160,Y+20
430 HPLOT TO 160,Y+10 : HPLOT TO 140,Y
435 HPLOT 155,Y+20 TO 160,Y+25
437 HPLOT 125,Y+20 TO 120,Y+25
440 RETURN
460 VTAB (20+A) : HTAB 8
470 INVERSE
480 PRINT SPC(B/4);
490 NORMAL
500 PRINT SPC(26-B/4);
510 RETURN
530 I$="" : IF PEEK(-16384)>127
    THEN GET I$
540 IF I$="A" THEN T=T+4 :
    IF T>100 THEN T=100
550 IF I$="D" THEN T=T-4 :
    IF T<0 THEN T=0
560 IF T>F THEN T=F
570 RETURN
590 HOME : VTAB 23
600 PRINT "LOST IN SPACE!!"
640 RETURN
660 HOME : VTAB 22
670 PRINT "LANDED AT VEL ";
    INT((V+V1)*5)/10
680 IF (V+V1)<8 THEN 700
690 PRINT "ALL DEAD" : RETURN
700 PRINT "SAFELY" : RETURN
```

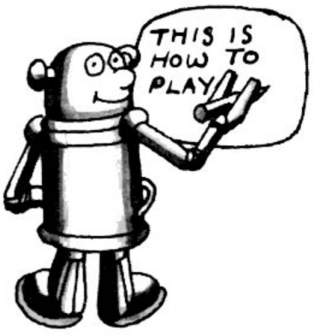


Adding to the programs

Here are some ideas for additions you can make to the programs in this book or to your own programs. In most cases you won't be able to add these to a ZX81 with only 1K as the games themselves fill almost all its memory space, but you should find there is plenty of room on the other computers.

Remember you will either have to restrict your additions to the spare line numbers in a program or renumber the program. If you decide to renumber, take care you change all the GOTO and GOSUB lines too.

Getting the computer to tell you how to play



You can add a section to any program to make the computer print instructions telling you what to do. The easiest way to do this is to add some lines, such as those below, at the beginning of the program and then put a sub-routine at the end.

```
    10 PRINT "TITLE OF GAME"
    11 PRINT "DO YOU WANT TO"
    12 PRINT "KNOW HOW TO PLAY?"
    15 INPUT I$
 S ZX17 IF I$(1)="Y" THEN GOSUB 1000
★■▲●17 IF LEFT$(I$,1)="Y" THEN GOSUB 1000
```

main program goes here

```
1000 PRINT "WHAT YOU HAVE TO"
1010 PRINT "DO IS........"
1999 RETURN
```

You can add as many print statements as you like for the instructions, just remember to put a number and the word PRINT at the beginning of each one. Restrict the length of the part inside the quotation marks to the number of characters your computer can print on one line. Don't forget to put a RETURN line at the end or the program won't work.

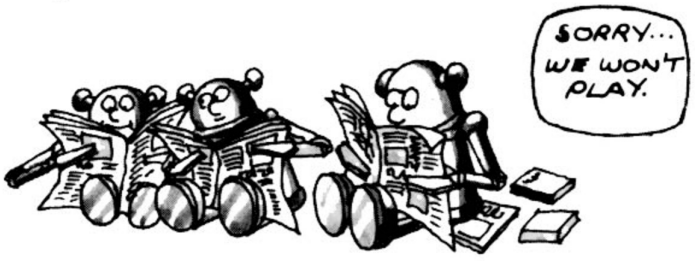
Making the computer stop and wait for you



If your instructions are very long, you may want to insert this sub-routine which stops the program running at a particular point until you press a key. This way you can stop the instructions scrolling off the top of the screen before you have read them. Put a GOSUB line at the place you want the program to stop and then put this sub-routine at the end.

```
    1000 PRINT "PRESS A KEY TO CONTINUE ";
■S ZX1010 IF INKEY$="" THEN GOTO 1010
 ★1010 I$=GET$
 ●1010 GET I$
 ▲1010 GET I$ : IF I$="" THEN GOTO 1010
    1020 PRINT
    1030 RETURN
```

Making the computer "talk" to you



You can make the computer ask you questions and react to your answers. For instance, here is an addition which will make the computer refuse to play with you unless your name begins with J.

```
    1 PRINT "WHAT IS YOUR NAME?"
    2 INPUT I$
    3 IF I$(1)<>"J" THEN GOTO 1000
    3 IF LEFT$(I$,1)<>"J" THEN GOTO 1000
    4 PRINT "OK-YOU CAN PLAY."
    5 PRINT "ARE YOU READY?"
    6 INPUT J$
S ZX7 IF J$(1)<>"Y" THEN GOTO 5
★■▲●7 IF LEFT$(J$,1)<>"Y" THEN GOTO 5
```

main program here

```
1000 PRINT "SORRY THIS GAME IS"
1010 PRINT "ONLY FOR PEOPLE"
1020 PRINT "WHOSE NAMES BEGIN"
1030 PRINT "WITH J"
```

Here is another one where the computer dares you to be brave enough to play.

```
    10 PRINT "VERY SCAREY GAME"
    12 PRINT "ARE YOU BRAVE ENOUGH"
    14 PRINT "TO TACKLE THE GREEN"
    15 PRINT "HAIRY MONSTER?"
    16 INPUT I$
 S ZX17 IF I$(1)="Y" THEN GOTO 20
★■▲●17 IF LEFT$(I$,1)="Y" THEN GOTO 20
    18 PRINT "COWARD"
    19 STOP
```

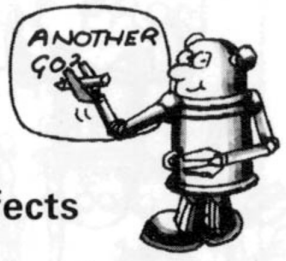
You could combine this with the instruction sub-routine by taking lines 11 to 17 from the instructions section on this page and putting them at lines 20 to 26 of this program. You can then start the main program at line 30 and add the instruction sub-routine at the end.

Would you like another go?

Instead of typing RUN each time you play a game, you can make the computer ask you if you'd like another go. Put these lines at the end of the program, just before the last STOP statement.

```
    1000 PRINT "DO YOU WANT ANOTHER GO?"
    1010 INPUT I$
 S ZX1020 IF I$(1)="Y" THEN RUN
★■▲●1020 IF LEFT$(I$,1)="Y" THEN RUN
    1030 PRINT "OK THEN - BYE"
    1040 STOP
```

Change line numbers according to your program.



Adding sound effects

The BBC, VIC 20, ZX Spectrum and some Apples are able to produce sounds and you can add lines to your programs to make them do so at appropriate places. You could add an explosion for instance, or a little tune which plays if you win. All the computers need different instructions to make sounds though, so you will have to look at your manual. In some cases you can add a single line to your program at the place you want the sound. In others, you need several lines and it is best to put these in as a sub-routine.

As an example, here is the sound of a shot for the BBC. You can experiment with where to put it in the program, but you must give it a line number to make it work:

```
SOUND 0,-15,5,10
```

At the back of the VIC manual you will find some useful sub-routines for sounds such as "laser beam", "explosion" and "red alert". Put a GOSUB line where you want the sound to appear, number the sub-routine and add a RETURN at the end of it.



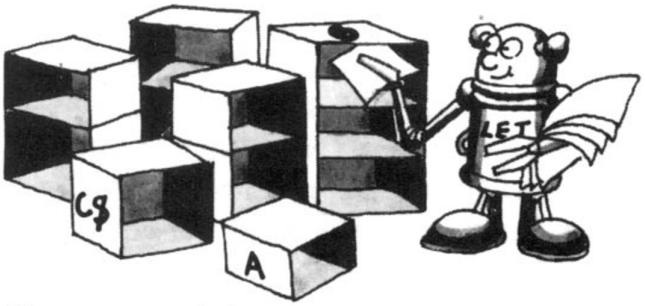
Special note for BBC and Spectrum users

If you have a BBC or a ZX Spectrum you may find that some of the games in this book run too fast for you. You will find a box next to these games containing instructions for changing the speed. Remember, to slow the game up you always need to use a higher number. Later models of the BBC may run up to twice as fast as the earlier models, and this could make the games appear impossible on the first run. Be prepared to make big changes to the speed number to correct this.

Writing your own programs

As you work through the games in the book, you will probably find yourself making more and more changes to them and eventually wanting to write new games of your own. On these two pages you will find some hints on how to set about doing this.

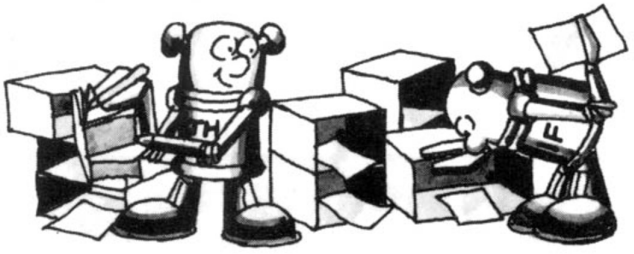
Before you start, it is a good idea to stop and think about what your computer can and cannot do.



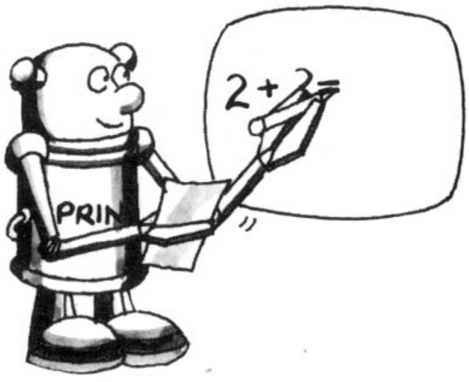
*It can store information



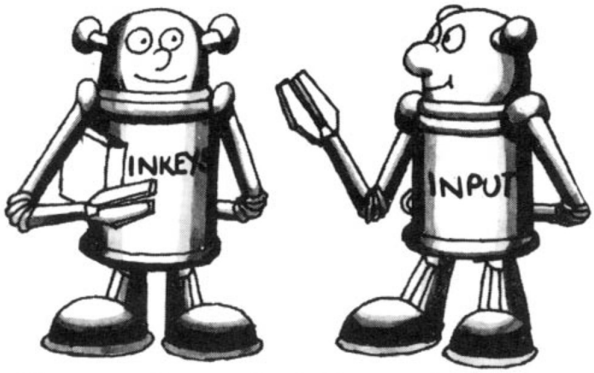
*It can do calculations.



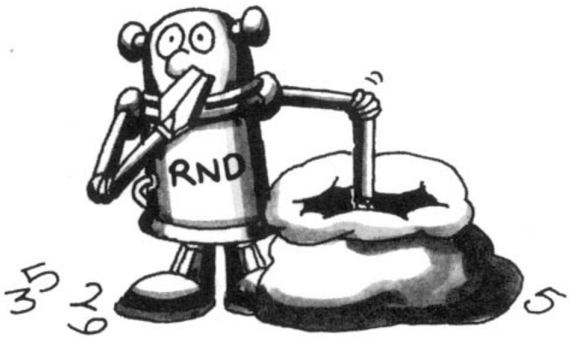
*It can make decisions by comparing items of information in various ways.



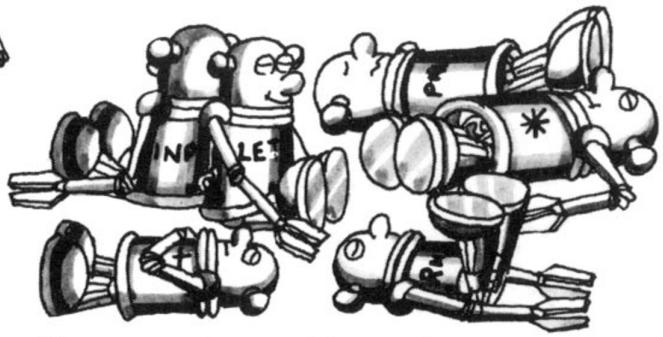
*It can tell you the results of its calculations and decisions and also what is stored in its memory.



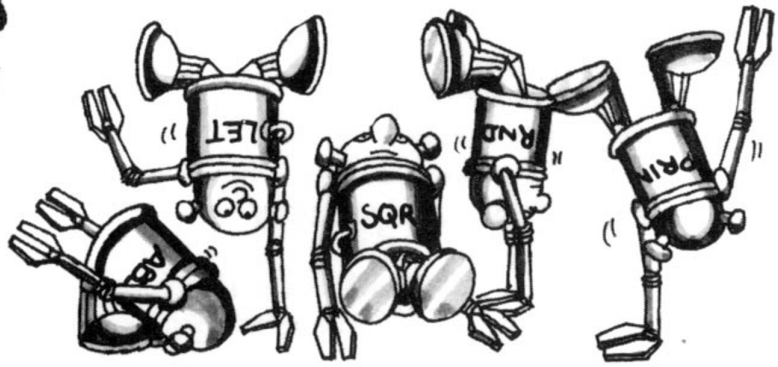
*It can ask you for information.



*It can select numbers at random by using RND.



*It cannot do anything unless you tell it to.



*Provided you use its language correctly, it can do only *exactly* what you tell it, even if it is silly.

Remember, when you are trying to work out a game, not to include anything which your computer won't be able to do.

Planning a game

Before you can tell the computer how to play your game, you must know exactly how to play it and what the rules are yourself. The computer will need a series of simple logical instructions, so work out your game in your head or on paper first and then break it down into simple steps.

Next write a plan (in English – don't try to use BASIC yet) of all the stages of the game in order.

Here is a plan for a simple shooting game, such as firing cannon balls at a pirate ship or shooting laser beams at an alien invader, to give you an idea.

PLAN

1) PRINT TITLE AND INSTRUCTIONS

2) CHOOSE A TARGET FOR THIS GAME

3) BEGIN A LOOP TO GIVE THE PLAYER N GOES

4) GET A SHOT FROM THE PLAYER

5) CHECK IF SHOT WAS ON TARGET

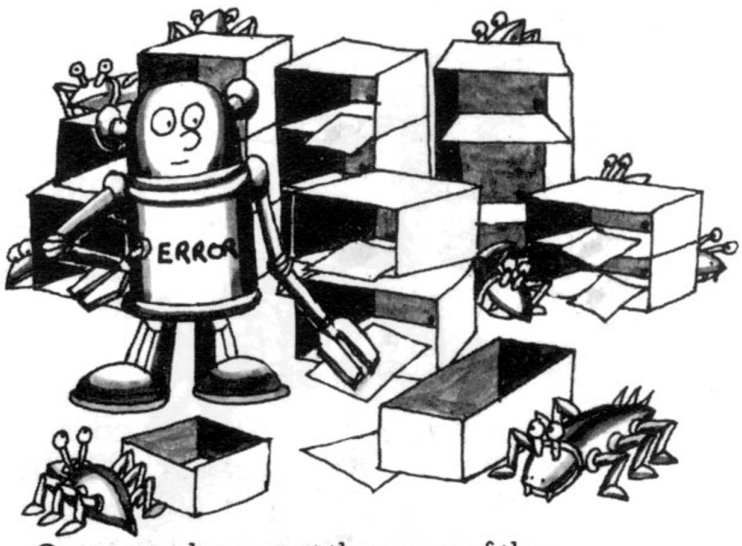
6) PRINT MESSAGE DEPENDING ON ACCURACY OF SHOT

7) GO BACK FOR ANOTHER GO IF SHOT WAS UNSUCCESSFUL

Writing the program

The next stage is to convert your plan into BASIC. Each step in your plan may need several lines in BASIC. Don't forget to leave gaps when numbering your program lines so you can go back and add extra ones if you need to.

Do a first draft of the program on paper first and then start testing on the computer. Your computer will spot errors much more quickly than you will see them yourself and may give you a clue as to what is wrong. Remember that debugging programs is a long, tedious process even for expert programmers, so don't expect to get yours right first time.

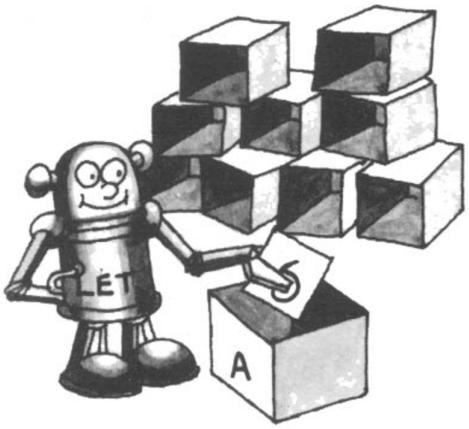


Once you have got the core of the program working, you can add to it. Scoring, extra comments, more targets etc. can all be incorporated later. You could add sections from the programs in this book to your games.

Don't expect to be able to write exciting and original games straight away. Keep your ideas very simple and be prepared to adapt them as you go along. You may find you have included something in your game which is easy for humans to do but very difficult for a computer. As you get more experienced you will begin to know instinctively what your computer can do and find it easier to write programs for it.

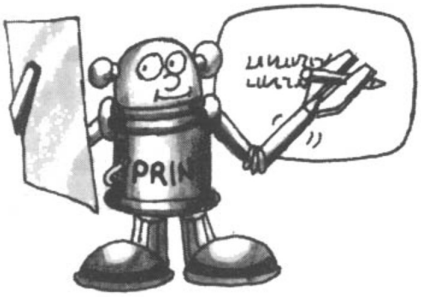
Summary of BASIC

This section lists some common BASIC words and describes what they make the computer do and how they are used. Most of them have been used in the programs in this book, so you can check back through the book to see how they work in a game. Not all the words can be used on all the computers mentioned in this book. The conversion chart on page 46 shows what you can use instead.



LET tells the computer to label a section of its memory and put a particular value in it e.g. LET A=6 means label a section of memory "A" and put the value 6 in it. "A" is called a "variable" and putting something in it is called "assigning a value to a variable".

Some variable labels are followed by a dollar sign e.g. A$. This means they are for "strings", which can contain any number of characters, including letters, numbers and symbols.





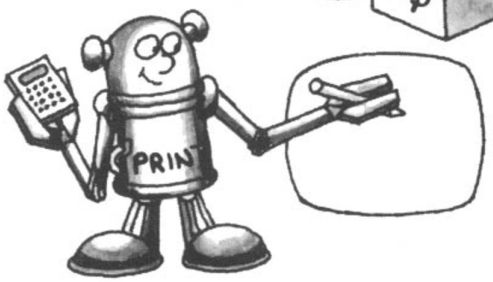
PRINT tells the computer to display things on the screen and you can use it in several ways:

A message enclosed in quotation marks with PRINT in front of it will be displayed on the screen exactly as you typed it. The section inside quotes does not have to be in BASIC, it can be anything you like.

PRINT followed by a variable label e.g. PRINT A or PRINT A$ tells the computer to display the contents of that variable on the screen.

PRINT can also do calculations and then display the results e.g. PRINT 6*4 will make the computer display 24.

You can use PRINT by itself to leave an empty line.





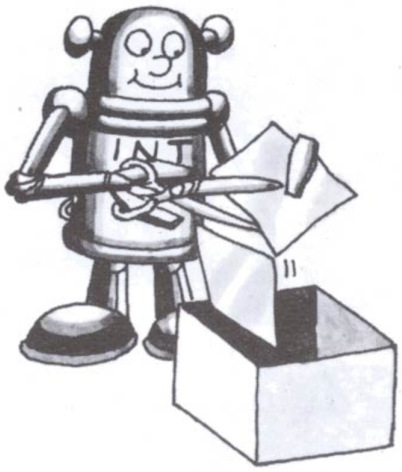
RND tells the computer to choose a number at random. Different computers use different forms of RND and you can see what these are in the conversion chart on page 46. On Sinclair computers RND by itself produces a number between 0 and 0.99999999. You can vary the limits of the number it chooses by multiplying RND and adding to it. E.g. RND*20 produces a number between 0 and 19.99999999, while RND*20+1 produces a number between 1 and 20.99999999.

See INT for how to produce only whole numbers.

See CHR$ for how to produce letters and other keyboard characters at random.

INT is short for integer, which means whole number. For positive numbers, it tells the computer to ignore everything to the right of the decimal point. E.g. INT(20.999) is 20. For negative numbers, it ignores everything to the right of the decimal point and "increases" the number to the left of it by one e.g. INT(−3.6) is −4.

INT is often used with RND, like this:
INT(RND*20+1) which tells the computer you want it to choose a whole number between 1 and 20.

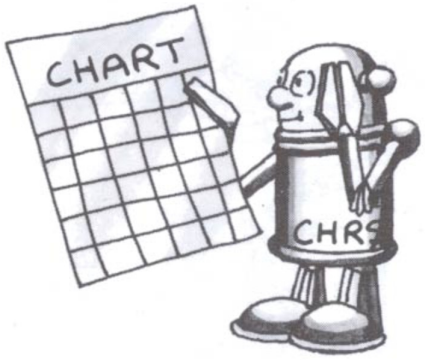


CHR$ converts numbers into letters. Apart from the ZX81, all the computers in this book use the ASCII* set of keyboard characters in which each character corresponds to a certain number. E.g. letter A has the code number 65 and PRINT CHR$(65) will display an A on the screen.

You can use CHR$ with INT and RND to make the computer select random letters, like this:

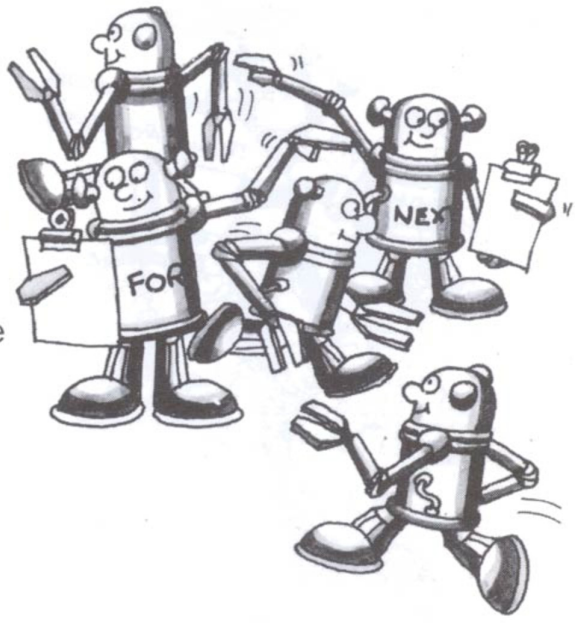
CHR$(INT(RND*26+65))

This line will produce random letters on a ZX Spectrum (see conversion chart for other computers).



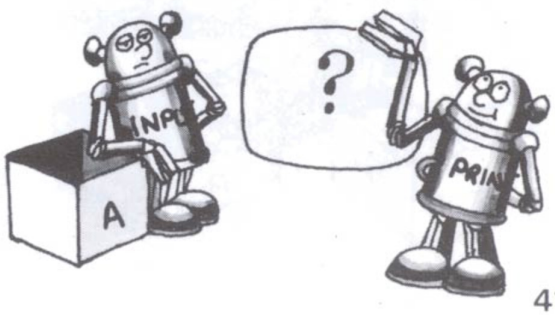
FOR is used to start a "loop" which will make the computer repeat part of a program a certain number of times. It must be followed by a variable (such as G to stand for the number of goes allowed in a game), and the variable must be given start and end values (such as 1 TO 10.)

The end of the loop is marked by a NEXT line (NEXT G in this example) which increases the value of the variable by 1 each time and then sends the computer back to the FOR line again. When the variable reaches its end value, the computer ignores the NEXT line and carries on to the line which follows it. Every FOR must have a NEXT or you will get a bug.

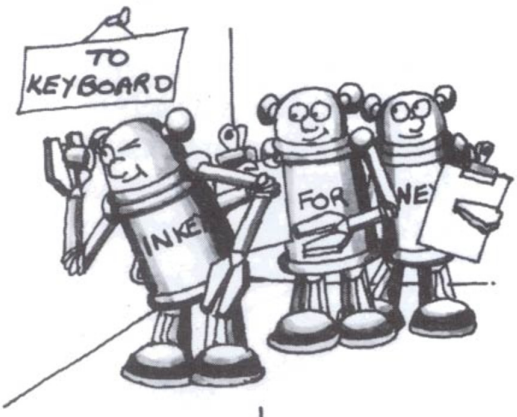


INPUT labels a space in the computer's memory, prints a question mark and then waits for you to type something which it can put in this memory space. It will not carry on with the rest of the program until you press RETURN, ENTER or NEWLINE.

You can use number or string variables with INPUT, but if you use a number variable the computer will not accept letters from you.

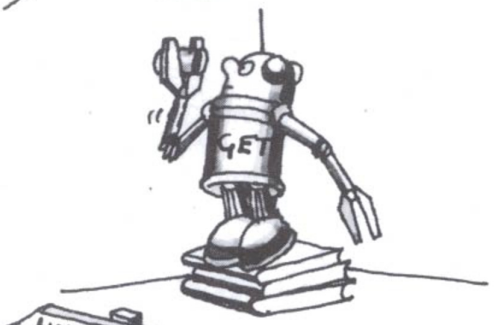


*American Standard Code for Information Interchange (see page 45)



INKEY$ checks the keyboard to see if a key is being pressed and if so which one. It does not wait for you to press a key like INPUT does. It is usually used in a loop which makes the computer go round checking the keyboard lots of times. This is because computers work so quickly, you wouldn't have a chance of pressing a key in the time it takes the computer to do one check.

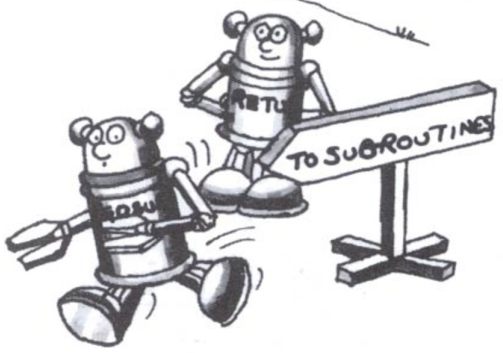
If you haven't pressed a key before the loop finishes, the computer carries on with a string containing nothing (called a "null" string).
NB Apple and VIC do not use INKEY$.



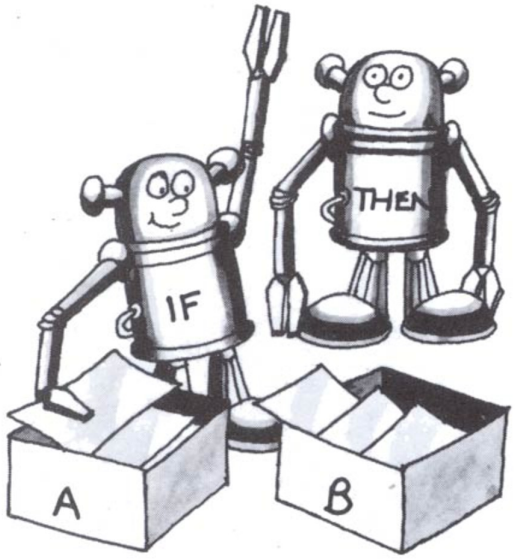
GET is used instead of INKEY$ on VIC and Pet computers.



GOTO makes the computer jump up or down the program ignoring the lines in between. You must put the number of the line you want it to jump to after the GOTO instruction.



GOSUB tells the computer to leave the main program and go to a sub-routine. GOSUB must be folowed by the number of the first line of the sub-routine. At the end of the sub-routine you must have a RETURN line. This sends the computer back to the main program to the line immediately following the GOSUB line. A GOSUB without a RETURN in a program will give a bug.

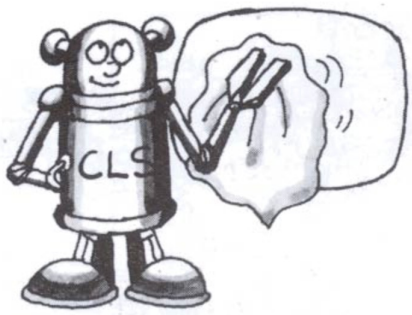


IF . . . THEN tells the computer to decide if an expression is true or false, and do different things depending on the answer. It is used with the following signs, and also with AND or OR:

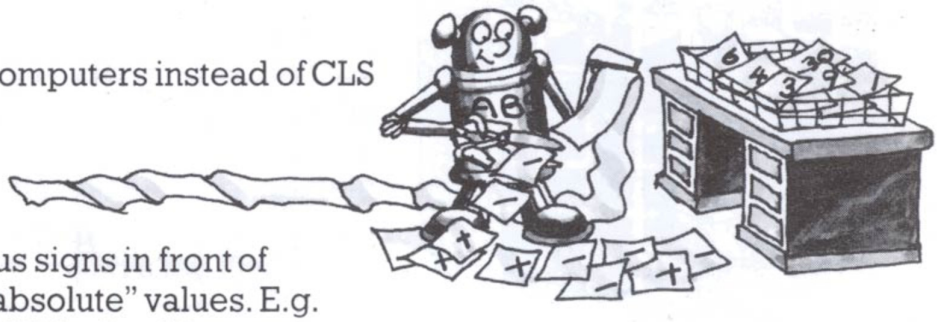
=the same as
<less than
>greater than
<=less than or the same as
>=greater than or the same as
<>not the same as

If the computer decides an expression is true, it carries on to do the instruction which follows THEN. If it decides it is false, it ignores the rest of that line and goes on to the next one.

CLS is used to clear everything off the screen without removing or changing anything in the memory. It is useful for removing the listing from the screen at the beginning of a RUN or in games when you want the player to react to something seen for a limited amount of time. (NB Apple and VIC do not use CLS – see conversion chart).

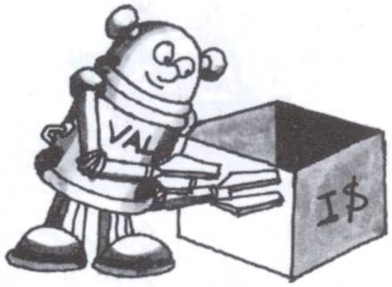


HOME is used by Apple computers instead of CLS to clear the screen.

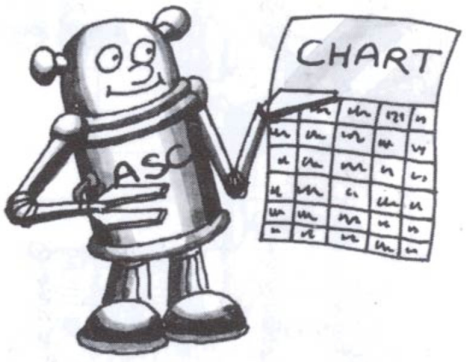


ABS ignores plus and minus signs in front of numbers and takes their "absolute" values. E.g. ABS(−10) is 10 and ABS(+10) is also 10.

VAL takes the numeric value of numbers written as strings. In effect, it tells the computer to ignore the dollar sign and treat the string as an ordinary number variable. E.g. if I$="60" then VAL(I$) is the number 60.



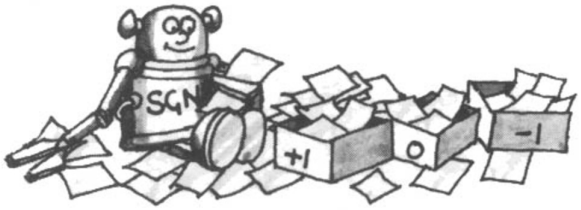
ASC converts a character into its ASCII code number e.g. ASC("3") gives 51. The expression in brackets must be a string e.g. ASC(A$) or ASC("20").
NB ZX81 and ZX Spectrum do not use ASC, though the Spectrum does use the ASCII code.



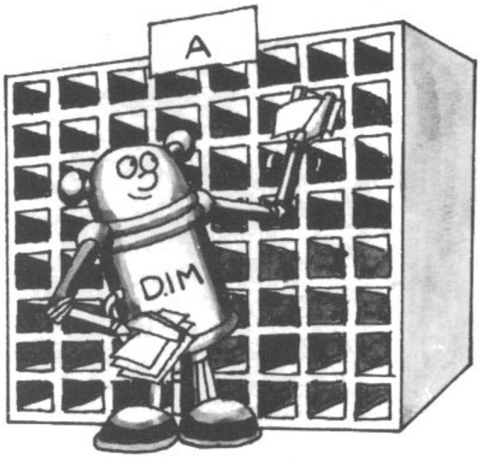
CODE is used by ZX81 and Spectrum in place of ASC. Like ASC it must always be followed by a string. Remember that the ZX81 uses different code numbers from the other computers.

TAB moves the cursor across the screen to a specified column number. It is usually used with PRINT to display something in the middle of the screen. The number of spaces you want the cursor moved is put in brackets after TAB. The maximum number you can use depends on the screen width of your computer.





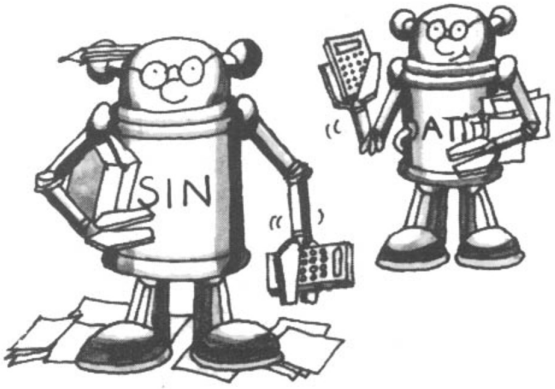
SGN tells the computer to find out the sign of a number. It produces −1 for a negative number, 0 for zero and +1 for positive numbers. E.g. SGN(−30) is −1, SGN(7) is +1 and SGN(0) is 0.



DIM tells the computer how much memory space will be needed for an "array" (a row or a grid). E.g. DIM X(6) tells the computer to set aside an area large enough to contain a row of 6 elements and labelled X. DIM A(8,8) means a memory space labelled A and big enough to take 8 elements across and 8 down is needed. The number of elements of data used in the program must correspond to the numbers in brackets after DIM or you will get a bug.

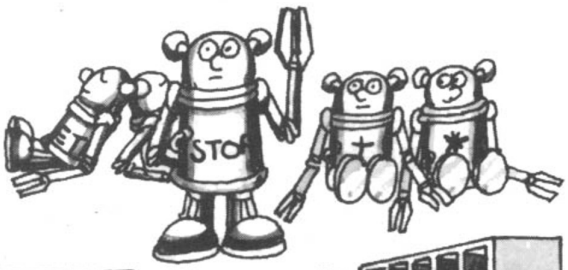


SQR takes square roots of numbers. E.g. SQR(16) gives the answer 4.

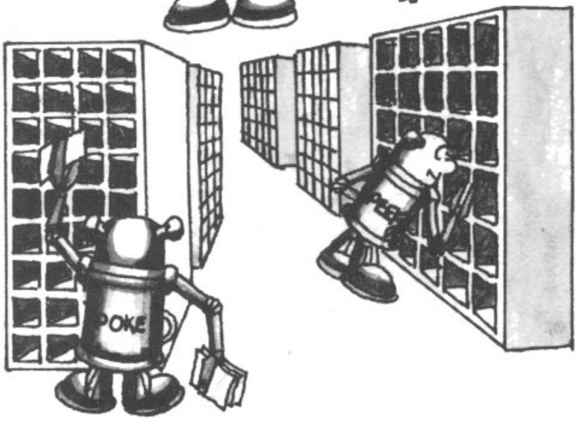


SIN calculates the sine of an angle. In a right-angled triangle the length of the side opposite an angle, divided by the length of the hypotenuse (the side opposite the right angle) is the sine of that angle. When you use SIN in a program, the angle you are using it with must be measured in radians, not degrees.

ATN is one of the trig. functions which computers can calculate (see also SIN above). It stands for arctangent and it is important to remember that it gives an answer in radians, not degrees. You will need to use a maths book to find out how this works if you do not already know about it.



STOP tells the computer not to go any further in a program. Computers other than the ZX81 can use END instead.



PEEK is a way of finding out what is in a specific area of the computer's memory. You need to use it with a number which specifies an "address" in the memory.
NB not used on BBC.

POKE is a special way of putting information in the computer's memory by using a memory "address".
NB not used on BBC.

ASCII chart

Code number	ASCII character	Code number	ASCII character
32	space	62	>
33	!	63	?
34	"	64	@
35	#	65	A
36	$	66	B
37	%	67	C
38	&	68	D
39	'	69	E
40	(	70	F
41	)	71	G
42	*	72	H
43	+	73	I
44	,	74	J
45	-	75	K
46	.	76	L
47	/	77	M
48	0	78	N
49	1	79	O
50	2	80	P
51	3	81	Q
52	4	82	R
53	5	83	S
54	6	84	T
55	7	85	U
56	8	86	V
57	9	87	W
58	:	88	X
59	;	89	Y
60	<	90	Z
61	=		

ZX81 code chart

Code number	ZX81 character	Code number	ZX81 character
11	"	41	D
12	£	42	E
13	$	43	F
14	:	44	G
15	?	45	H
16	(	46	I
17	)	47	J
18	>	48	K
19	<	49	L
20	=	50	M
21	+	51	N
22	-	52	O
23	*	53	P
24	/	54	Q
25	;	55	R
26	,	56	S
27	.	57	T
28	0	58	U
29	1	59	V
30	2	60	W
31	3	61	X
32	4	62	Y
33	5	63	Z
34	6		
35	7		
36	8		
37	9		
38	A		
39	B		
40	C		

Chart of screen sizes

	Max. number of characters across (or number of columns)	Max. number of lines down (or number of rows)
VIC 20	22	23
TRS-80	64	16
BBC	20/40/80	16/24/32
ZX81	32	22
ZX Spectrum	32	22
Apple	40	25

Conversion chart

This quick reference chart shows some of the variations in the BASIC used by the machines in this book. It does not include instructions for graphics, sound or colour as these vary so enormously from machine to machine. Note also that although most computers (except the BBC) use PEEK and POKE, they do not use the same system of memory addresses, so the numbers used with PEEK and POKE must be changed for each computer.

	BBC	VIC/Pet	Apple	TRS-80	ZX Spectrum	ZX81
Select random number between 0 and 0.99999999	RND(1)	RND(1)	RND(1)	RND(0)	RND	RND
Select random number between 1 and N	RND(N)	RND(1)*N+1	RND(1)*N+1	RND(N)	RND*N+1	RND*N+1
Select random letter between A and Z	CHR$(RND(26)+64)	CHR$(INT(RND(1) *26+65))	CHR$(INT(RND(1) *26+65))	CHR$(RND(26)+64)	CHR$(INT(RND*26+65))	CHR$(INT(RND*26+38))
Clear screen	CLS	PRINT CHR$(147)	HOME	CLS	CLS	CLS
Check keyboard to see if key being pressed	INKEY$(N)	GET X$	X$=" " IF PEEK(−16384) >127 THEN GET X$	INKEY$	INKEY$	INKEY$
Convert characters into code numbers	ASC("X") (using ASCII code)	ASC("X") (using ASCII code)	ASC("X") (using ASCII code)	ASC("X") (using ASCII code)	CODE("X") (using ASCII code)	CODE("X") (using ZX81 code)
Move cursor up	PRINT CHR$(11)	PRINT CHR$(145)	CALL −998	PRINT CHR$(27)	PRINT CHR$(11)	PRINT CHR$(112)
Move cursor down	PRINT CHR$(10)	PRINT CHR$(17)	PRINT CHR$(10)	PRINT CHR$(26)	PRINT CHR$(10)	PRINT CHR$(113)
Move cursor left	PRINT CHR$(8)	PRINT CHR$(157)	PRINT CHR$(8)	PRINT CHR$(24)	PRINT CHR$(8)	PRINT CHR$(114)
Move cursor right	PRINT CHR$(9)	PRINT CHR$(29)	PRINT CHR$(21)	PRINT CHR$(25)	PRINT CHR$(9)	PRINT CHR$(115)
Take 1st N characters of string	LEFT$(A$,N)	LEFT$(A$,N)	LEFT$(A$,N)	LEFT$(A$,N)	A$(1 TO N)	A$(1 TO N)
Take last N characters of string	RIGHT$(A$,N)	RIGHT$(A$,N)	RIGHT$(A$,N)	RIGHT$(A$,N)	A$(N TO)	A$(N TO)
Take middle N characters of string	MID$(A$,N1,N2)	MID$(A$,N1,N2)	MID$(A$,N1,N2)	MID$(A$,N1,N2)	A$(N1 TO N2)	A$(N1 TO N2)



Answers

You may find that your answers to some of the puzzles are different to the ones given here. As long as they work on your computer then this doesn't really matter, but check to see if they are as neat and simple as the answers in the book.

Page 5
Starship Takeoff

Lines 30 and 40 select the numbers which determine what the force will be. To increase the range of possible forces, you can increase either the 20 in line 30, or the 40 in line 40, or both of these numbers. Increasing the range of forces will obviously make the game more difficult.

Page 7
Intergalactic Games

Change lines 222 and 230 as follows:

```
222 LET B=B+INT(1000/G)
230 GOTO 20
```

and add a new line 15:

```
15 LET B=0
```

Page 9
Evil Alien

Change lines 20 and 30 and add a new line 25 as follows:

```
20 PRINT "HOW DIFFICULT? (6 TO 30)"
25 INPUT S
30 LET G=INT(S/3)
```

Page 11
Beat the Bug Eyes

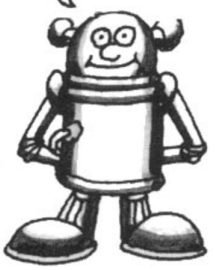
To make the bugs appear in more than four places on the screen, you need to put a higher number than 4 in the middle of line 70, change line 80 and add more sub-routines at the end of the program – one for each extra position.

Here are the changes to make the bugs appear in 5 places:

```
70 LET R=INT(RND*5+1)
S ZX80 GOSUB 220+20*R
★■▲●80 ON R GOSUB 240,260,280,300,320
```

```
240 LET D=5
245 LET A=1
250 GOTO 350
260 LET D=1
265 LET A=9
270 GOTO 350
280 LET D=5
285 LET A=18
290 GOTO 350
300 LET D=10
305 LET A=7
310 GOTO 350
320 LET D=15
325 LET A=15
330 GOTO 350
```

You can use any numbers you like for A and D provided they will fit on your screen.



To add more bugs, change the 10 in lines 30 and 220 to a higher number. (Make sure you use the same number for both lines.)

Page 13
Moonlander

To increase the speed allowed for a safe landing, you need to make changes to lines 230, 240 and 250. You can use any numbers you like – the higher they are the easier the game will be. In this example, you are allowed a speed of 2 for a good landing and 7 for an OK landing:

```
230 IF V1>7 THEN PRINT "YOU CRASHED
      - ALL DEAD"
240 IF V1>2 AND V1<=7 THEN PRINT "OK
     -BUT SOME INJURIES"
250 IF V1<=2 THEN PRINT "GOOD LANDING"
```

Page 15
Monsters of Galacticon

Four ways of making this game harder are:

1 Start the game with less people in the group by putting a smaller number than 5 in line 40.

2 Increase the number of monsters by changing the 4 in lines 20 and 30. Add

extra monster names at lines 81 to 89 using M$(5) and M$(6).

3 Reduce the number of goes allowed by altering the 8 in line 160.

4 Increase the chance of the monster being angered in line 330 by increasing .4 a little.

Page 17
Alien Snipers

In this game, N is the code number. To change the scoring to fit the code number you need to increase the score by N each time instead of 1. So, change line 190 as follows:

```
S ZX190 IF I$=CHR$(CODE(L$)+N) THEN
          LET S=S+N
★■▲●190 IF I$=CHR$(ASC(L$)+N) THEN
          LET S=S+N
```

Page 19
Asteroid Belt

You need to change line 260 so that the computer adds the number of stars to your score instead of 1. The number of stars is controlled by the value chosen for N in line 70, so, as in the puzzle above, you need to add N to the score. You also need to change line 320.

```
260 LET S=S+N
320 PRINT "YOU SCORED ";S;" POINTS"
```

Page 21
Trip into the Future

1 To increase the range of years which must elapse before you return to Earth, change the 100 in line 30 to a higher number, e.g. 150, like this:

```
30 LET T=INT(RND*150+25)
```

2 To increase the accuracy from 5 to 2 years, change the 5s in lines 180 and 190 to 2, like this:

```
180 IF ABS(T-T2)<=2 THEN PRINT "YOU
      ARRIVED ON TIME"
190 IF ABS(T-T2)>2 THEN PRINT "NOT
      EVEN CLOSE"
```

3 Line 170 contains the number which determines the length of your lifetime. Change the 50 to a higher number for a longer lifetime.

Page 23
Death Valley

You can make the valley longer by changing the number in line 30 to something higher than 200.

Page 25
Space Mines

Add these lines to make the computer ask if you would like another game:

```
645 PRINT "ANOTHER GAME? (TYPE Y OR N)"
646 INPUT A$
647 IF A$="Y" THEN GOTO 10
```

You must then a new line at 5 and change line 30 to add the money you ended up with at the end of the game to the money allowed for the new game:

```
5 LET M=0
30 LET M=M+INT(RND*50+10)*P
```

(Make sure you use the correct version of RND for your computer.)

First published in 1982 by Usborne Publishing Ltd, 20 Garrick Street, London WC2E 9BJ, England.
© 1982 Usborne Publishing Ltd
The name Usborne and the device are Trade marks of Usborne Publishing Ltd.

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

PRINTED IN BELGIUM BY proost INTERNATIONAL BOOK PRODUCTION

Usborne Computer Programs

Each of these colourful new books contains 14 simple games programs to play on a microcomputer.* Alongside the programs there are explanations of how they work and puzzles and suggestions for ways of changing them. Through playing these games even complete beginners will quickly begin to understand how a simple program works and be itching to write their own. There are tips and hints on writing programs and a summary of BASIC at the back of each book and also a chart which will help you convert programs in magazines and other books to work on your micro.

***The programs in these books are suitable for use on the following micros: ZX81, BBC, TRS-80, VIC 20, Pet, Apples which use Palsoft BASIC and ZX Spectrum.**

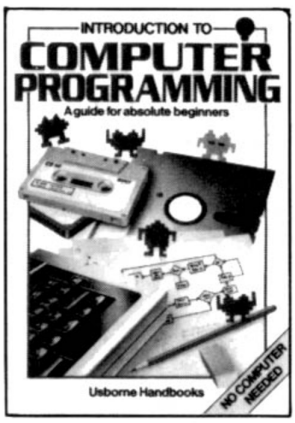




Other Computer Titles



A colourful guide to microcomputers, how they work and what they can do, with lots of ideas for things you can do with a micro.



A step-by-step guide to programming in BASIC for absolute beginners. With lots of programs to run on any microcomputer.



A colourful look at how computers play Space Invaders, chess and other games, with lots of tips on how to beat the computer.

Published in Canada by Hayes Publishing Ltd, 3312 Mainway, Burlington, Ontario, Canada, L7M1A7.
Published in the USA by Hayes Books, 4235 South Memorial Drive, Tulsa, Oklahoma, USA.

ISBN 0 86020 683 1

12/82 £1.99