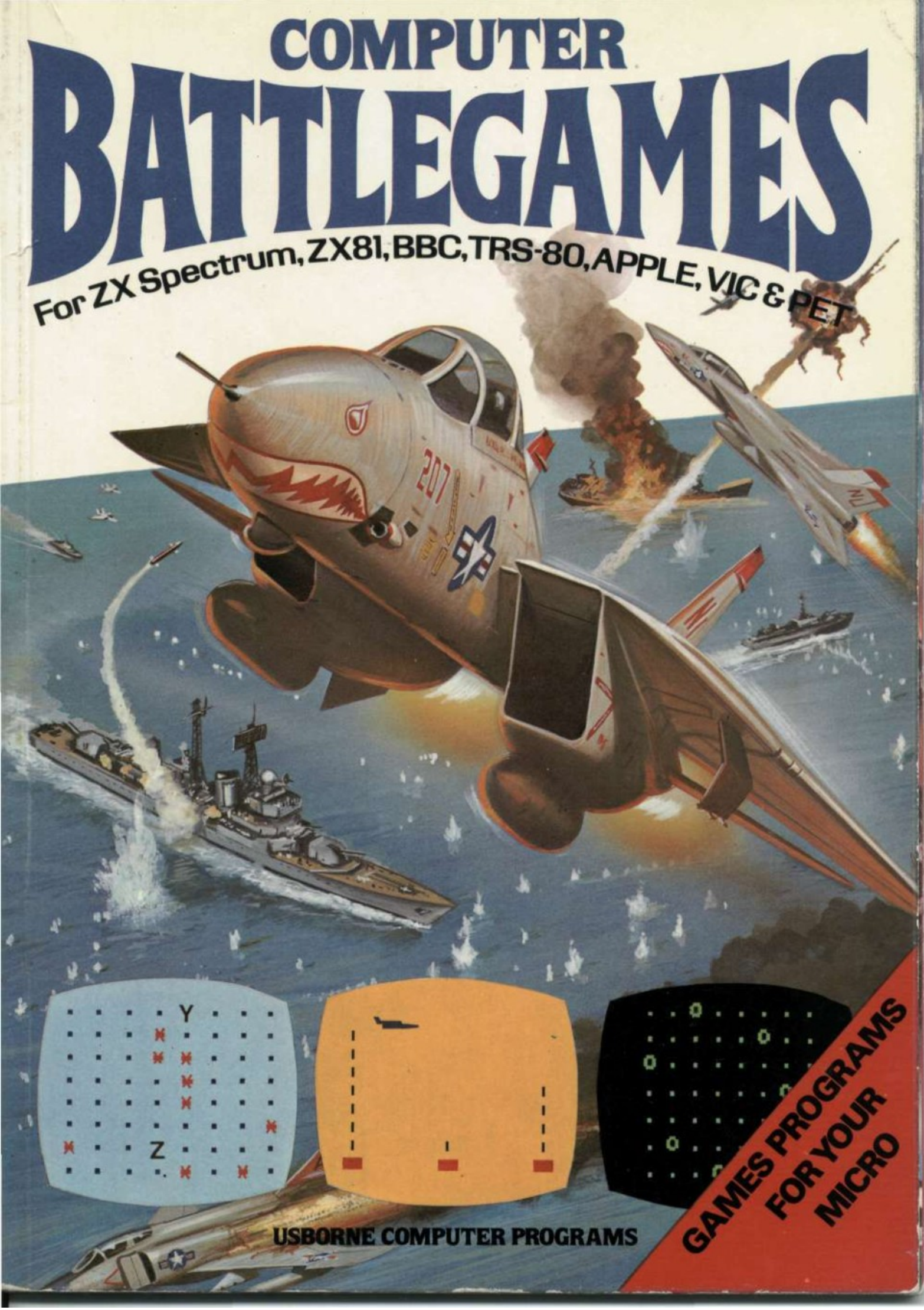
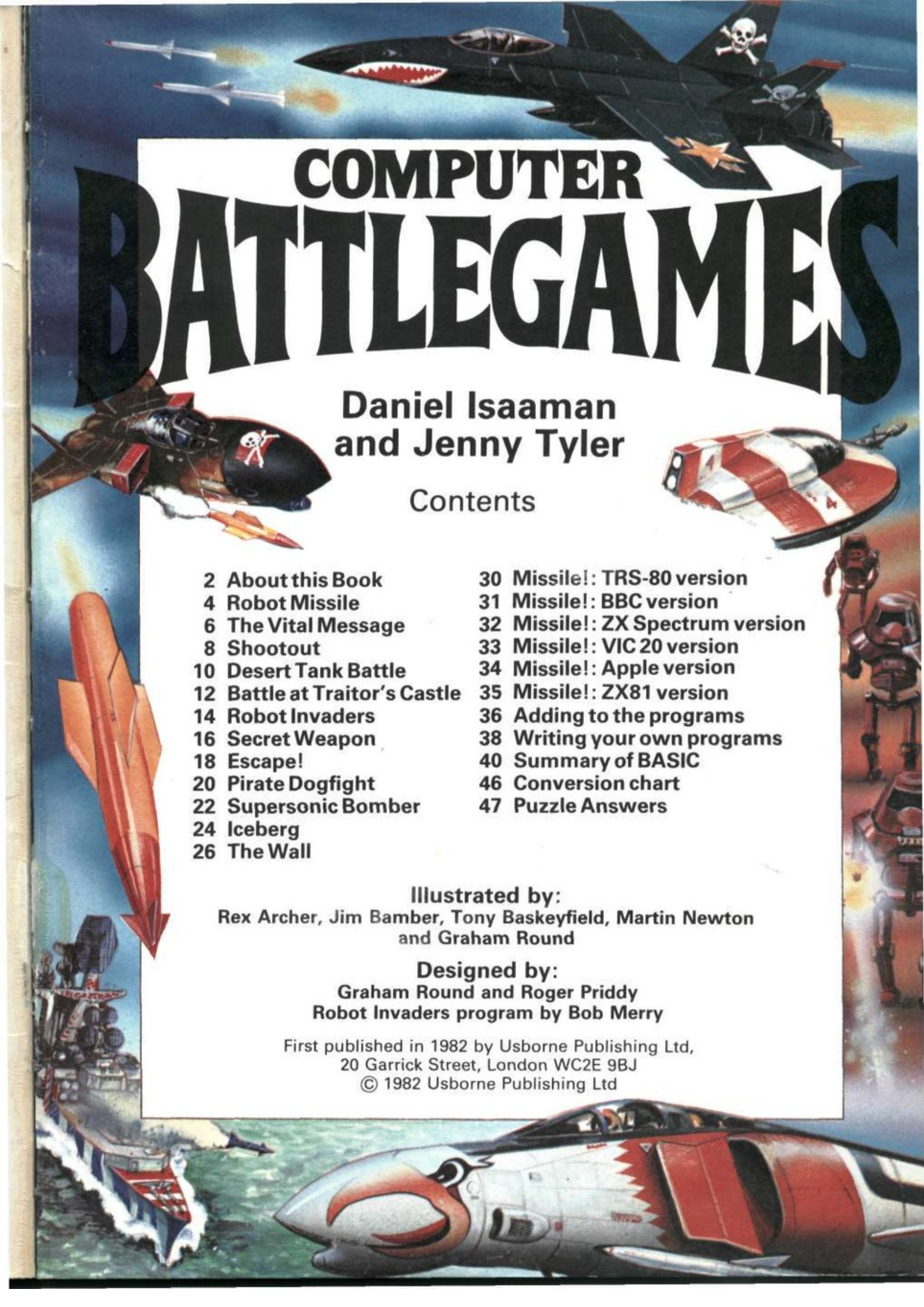
COMPUTER BATTLEGAMES

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



GAMES PROGRAMS FOR YOUR MICRO

USBORNE COMPUTER PROGRAMS



COMPUTER BATTLEGAMES

Daniel Isaaman and Jenny Tyler

Contents

2 About this Book
4 Robot Missile
6 The Vital Message
8 Shootout
10 Desert Tank Battle
12 Battle at Traitor's Castle
14 Robot Invaders
16 Secret Weapon
18 Escape!
20 Pirate Dogfight
22 Supersonic Bomber
24 Iceberg
26 The Wall
30 Missile!: TRS-80 version
31 Missile!: BBC version
32 Missile!: ZX Spectrum version
33 Missile!: VIC 20 version
34 Missile!: Apple version
35 Missile!: ZX81 version
36 Adding to the programs
38 Writing your own programs
40 Summary of BASIC
46 Conversion chart
47 Puzzle Answers

Illustrated by:
Rex Archer, Jim Bamber, Tony Baskeyfield, Martin Newton and Graham Round

Designed by:
Graham Round and Roger Priddy
Robot Invaders program by Bob Merry

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

© 1982 Usborne Publishing Ltd

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.

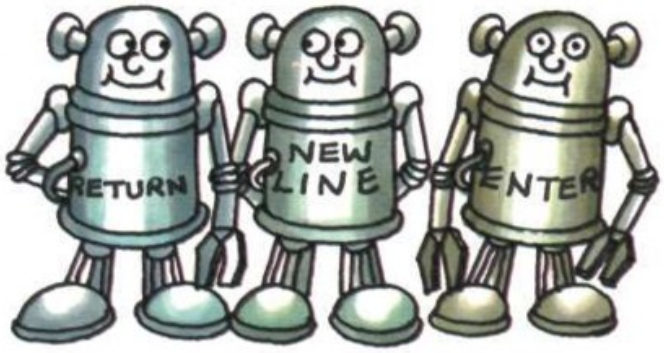
Typing in the programs

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.)







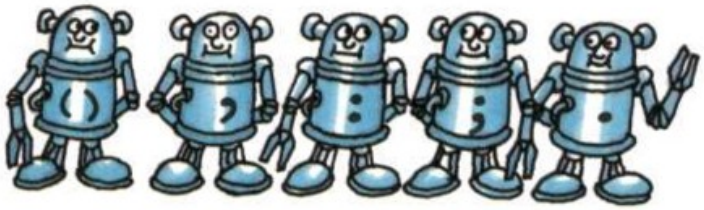
Check you have typed everything correctly. It is easy to make mistakes, so expect to find some. Use your manual to find out how to make changes to the program once it is typed in. If in doubt, you can always type the line again. All the computers will replace an existing line with a new one with the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

Robot Missile

The year is 2582 and the people of Earth are in the midst of battle against the Robots. A lethal Robot Missile has just landed and everyone is depending on you to find the secret code which unlocks its defuse mechanism. If you fail, the entire Earth Command Headquarters will be blown up.

Your computer knows what the code letter is. You must type in your guess and it will tell you whether the code letter is earlier or later in the alphabet. You have four chances to find the correct letter before the missile blows up.

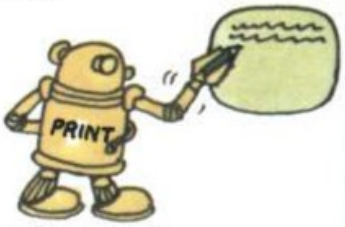




How the program works

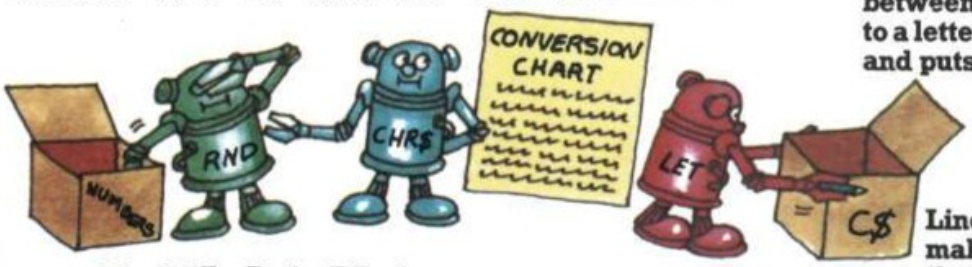
```
10 PRINT "ROBOT MISSILE"
20 PRINT
30 PRINT "TYPE THE CORRECT CODE"
40 PRINT "LETTER (A-Z) TO"
50 PRINT "DEFUSE THE MISSILE."
60 PRINT "YOU HAVE 4 CHANCES"
70 PRINT
```

These lines print the title and instructions.




```
s★■▲●80 LET C$=CHR$(37+INT(RND*26+1))
```

This selects a number between 0 and 26, converts it to a letter (the "secret code") and puts it in C$.




```
90 FOR G=1 TO 4
```

Line 90 begins a loop which makes lines 100-140 repeat 4 times.

```
100 INPUT G$
```

Puts your guess in G$.

```
110 IF G$=C$ THEN GOTO 210
```

Checks if your letter is the same as the "secret code" in C$. If so computer jumps to 210.

```
120 IF G$<C$ THEN PRINT "LATER";
130 IF G$>C$ THEN PRINT "EARLIER";
140 PRINT " THAN ";G$
```

Checks if secret code letter is earlier or later in the alphabet than yours and prints an appropriate message.




```
150 NEXT G
```

End of loop. Goes back for next turn.

```
160 PRINT
170 PRINT "BOOOOOOOOMMM..."
180 PRINT "YOU BLEW IT."
190 PRINT "THE CORRECT CODE WAS";C$
200 STOP
```

This prints if all your guesses were wrong.

```
210 PRINT "TICK...FZZZZ...CLICK..."
220 PRINT "YOU DID IT"
230 STOP
```

This prints if you guessed right.

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

```
★▲●80 LET C$=CHR$(64+INT(RND(1)*26+1))
  ■80 LET C$=CHR$(64+INT(RND(0)*26+1))
  s80 LET C$=CHR$(64+INT(RND*26+1))
```



Adding to the program

You can make the computer print an extra message for a correct guess on the last go. Change line 220 by adding a semicolon to it, like this:

220 PRINT "YOU DID IT";

and add a new line 230:

230 IF G=5 THEN PRINT "(JUST)"

Puzzle corner

See if you can work out how to change the program to give you more or less chances of guessing the code letter.

The Vital Message



You are a laser communications operator. Your job is to intercept robot messages and relay them to Command HQ. A vital code message is expected. If you relay it correctly, the Robot attack will be crushed.

This game tests your skill at remembering a group of letters which you see for only a very short time. The computer will ask you for a difficulty from 4 to 10. When you have typed in your answer, letters will appear top left of your screen and disappear again fairly quickly. Memorize them and then type them into the computer and see if you were right.

1K

How the program works

```
▲●10 CLS
  20 PRINT "VITAL MESSAGE"
  30 PRINT
  40 PRINT "HOW DIFFICULT? (4-10)"
  50 INPUT D
  60 IF D<4 OR D>10 THEN GOTO 40
```

Clears screen before game starts.

Asks you for a "difficulty number" and puts it in D.

Checks your number isn't less than 4 or more than 10.

```
  70 LET M$=""
```



Sets up an empty ("null" in computer language) string labelled M$ in which the computer can store the secret message.

```
  80 FOR I=1 TO D
s★■▲●90 LET M$=M$+CHR$(INT(RND*26+38))
 100 NEXT I
```

Computer loops round D times. Each loop it chooses a letter and adds it to the string of letters it has already chosen and put in M$.

```
▲●110 CLS
 120 PRINT "SEND THIS MESSAGE:"
 130 PRINT
 140 PRINT M$
```

Clears screen and then prints the message.

```
★■▲●150 FOR I=1 TO D*8
 160 NEXT I
```

Message stays on screen while computer loops round, doing nothing, for a number of times depending on D.

```
▲●170 CLS
 180 INPUT N$
 190 IF N$=M$ THEN GOTO 240
```

Clears screen when loop has finished.

Puts your version of the message in N$.

Checks if your message is the same as the message in M$ and jumps to 240 if it is.

```
 200 PRINT "YOU GOT IT WRONG"
 210 PRINT "YOU SHOULD HAVE SENT:"
 220 PRINT M$
 230 GOTO 260
 240 PRINT "MESSAGE CORRECT"
 250 PRINT "THE WAR IS OVER"
 260 STOP
```

Prints if you are wrong, telling you what the message should have been.

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

```
   ●10,110,170 HOME
   ▲10,110,170 PRINT CHR$(147)
★▲●90 LET M$=M$+CHR$(INT(RND(1)*26+65))
   ■90 LET M$=M$+CHR$(INT(RND(0)*26+65))
   s90 LET M$=M$+CHR$(INT(RND*26+65))
■▲●150 FOR I=1 TO D*180
   ★150 FOR I=1 TO D*400
```



Notice that the ZX81 uses a different code for the keyboard characters. All the other computers use the "ASCII" code.

How to make the game harder

You can change the program to include numbers and punctuation marks in the secret message. Do this by changing line 90 as follows:

```
 zx90 LET M$=M$+CHR$(INT(RND*43+21))
★▲●90 LET M$=M$+CHR$(INT(RND(1)*43+48))
   ■90 LET M$=M$+CHR$(INT(RND(0)*43+48))
    s90 LET M$=M$+CHR$(INT(RND*43+48))
```



Puzzle corner

Can you work out how to make the message stay on the screen longer?

Shootout

You are standing back to back. You take 10 paces, turn and reach for your gun. How quick are you? Can you shoot first?

Your computer prints the numbers 1 to 10 to represent the 10 paces, pauses and then prints HE DRAWS... You must be ready to press a key (any one will do) the instant these words come on the screen. If you are quick enough, you will win. Don't press a key before HE DRAWS comes up or you will automatically lose.

1K

```
▲●10 CLS
   20 PRINT "COWBOY SHOOTOUT -"
   30 PRINT "YOU ARE BACK TO BACK"
   40 PRINT "TAKE 10 PACES..."
   50 FOR I=1 TO 10
★■▲●→60 PRINT I;"..";
   70 NEXT I
   80 PRINT
★■▲●90 FOR I=1 TO RND*200
  100 NEXT I
 ★▲●110 IF INKEY$<>"" THEN GOTO 160
  120 PRINT "HE DRAWS..... ";
★■▲●130 FOR I=1 TO 5
 ★▲●140 IF INKEY$<>"" THEN GOTO 190
   ★150 NEXT I
```




```
  160 PRINT "AND SHOOTS."
  170 PRINT "YOU ARE DEAD."
  180 GOTO 210
  190 PRINT "BUT YOU SHOOT FIRST."
  200 PRINT "YOU KILLED HIM."
  210 STOP
```

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

```
   ●10 HOME
   ▲10 PRINT CHR$(147)
   ★60 PRINT ;I;"..";
★■▲●65 FOR J=1 TO 300 : NEXT J
 ★▲●90 FOR I=1 TO RND(1)*1000
   ■90 FOR I=1 TO RND(0)*1000
   ●110 IF PEEK(-16384)>127 THEN GOTO 160
   ▲110 GET I$ : IF I$<>"" THEN GOTO 160
   ★110 IF INKEY$(1)<>"" THEN GOTO 160
   ●130 FOR I=1 TO 20
  ■▲130 FOR I=1 TO 50
   ★DELETE 130,150
   ●140 IF PEEK(-16384)>127 THEN GOTO 190
   ▲140 GET I$ : IF I$<>"" THEN GOTO 190
   ★140 IF INKEY$(40)<>"" THEN GOTO 190
```

How to change the speed of the game

You can adjust the time you have to react to the message and press a key by changing the last number in line 130. A smaller number will give you less time. (For the BBC, change the number in brackets in line 140.)

Making the game harder

If you change the program as follows, you will add the possibility of you missing sometimes:
1) In line 140, change 190 to 220.
2) Add these lines:

```
s ZX 220 IF RND>.3 THEN GOTO 190
★▲●220 IF RND(1)>.3 THEN GOTO 190
  ■220 IF RND(0)>.3 THEN GOTO 190
  230 PRINT "BUT YOU MISSED"
  240 GOTO 90
```

How the program works

This is a loop which sends the computer round 10 times to print a number and two dots each time.

Another loop – this time to make the computer delay. The computer loops round a number of times depending on the value of RND, doing nothing.

Checks you're not cheating by pressing a key before HE DRAWS comes on the screen.

Prints the signal for you to press a key.

Checks the keyboard to see if you are pressing a key, and jumps to 190 if you are. (Notice that line 140 is in the middle of a FOR . . . NEXT loop. This makes the computer check the keyboard a number of times to give you a reasonable chance of pressing a key.)

Prints if you lose. (Either you ran out of time or were cheating.)

Prints if you win.

Delay loops can be written in one line, as in line 65, for all but the ZX81.



An extra delay loop for the faster computers.



Notice how the different computers check the keyboard: Vic uses GET, Apple has to PEEK into its memory, the others use INKEY$.



Puzzle corner

See if you can work out how to make it possible for the computer to miss too.

Desert Tank Battle

The last major stronghold of Robot forces outside the U.R.S* is hidden in ancient castle ruins in the middle of the desert. A fleet of desert hovertanks has been sent to destroy it and you are the commander. Your tank controls the five remaining missiles.

You must assess carefully the direction and elevation before you launch each one. Your computer will ask you for a direction angle between −90° (extreme left) and +90° (extreme right) and an elevation angle between 0° (along the ground) and 90° (straight up in the air). The elevation determines the distance the missile will travel.

Is your aim good enough to destroy the robot stronghold?



How the program works

```
    10 PRINT "DESERT TANK BATTLE"
★■▲●20 LET T=INT(RND*181)-90
★■▲●30 LET D=RND
    40 FOR G=1 TO 5
    50 PRINT "DIRECTION (-90 TO 90) ?"
    60 INPUT T1
    70 PRINT "ELEVATION (0 TO 90) ?"
    80 INPUT B
    90 LET D1=SIN(2*(B/180*3.1416))
   100 IF ABS(T-T1)<2 AND ABS(D-D1)<.05 THEN GOTO 220
   110 PRINT "MISSILE LANDED ";
   120 IF T1<T THEN PRINT "TO THE LEFT ";
   130 IF T1>T THEN PRINT "TO THE RIGHT ";
   140 IF ABS(D1-D)>.05 AND T1<>T THEN PRINT "AND ";
   150 IF D-D1>.05 THEN PRINT "NOT FAR ENOUGH";
   160 IF D1-D>.05 THEN PRINT "TOO FAR";
   170 PRINT
   180 NEXT G
   190 PRINT "DISASTER - YOU FAILED"
   200 PRINT "RETREAT IN DISGRACE"
   210 STOP
   220 PRINT "*KABOOOMMM*"
   230 PRINT "YOU'VE DONE IT"
   240 STOP
```

This selects a whole number between −90 and 90 for the direction.

This selects a number between 0 and 1 for the distance of the castle from you.

Get your guesses and puts them in A and B.

Uses your elevation angle to calculate the distance your missile went. (Answer will be between 0 and 1).

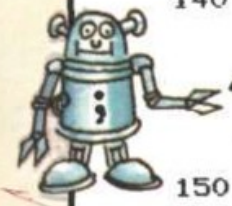
If your direction was within 2 degrees and your distance within 0.05, then you have hit the castle. Program jumps to line 220 to tell you so.



ABS takes the "absolute" value of a number, which means it ignores + or − signs.

Compares your direction angle with the number chosen in line 20 and prints an appropriate message.

Decides whether to print "and" by comparing the distance your missile travelled with the number chosen in line 30.



A semi-colon at the end of a print statement tells the computer *not* to go to a new line for the next item to be printed.

Prints a message if your shot was too long or short.

Prints if you lose.

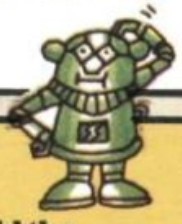
Prints if you win.

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

```
★▲●20 LET T=INT(RND(1)*181)-90
   ■20 LET T=INT(RND(0)*181)-90
★▲●30 LET D=RND(1)
   ■30 LET D=RND(0)
```



Only the ZX81 and Spectrum need LET. You can leave it out for the other computers.



Puzzle corner

Can you work out how to add the possibility of the robots seeing you and shooting at you before your five goes are up?

*United Robot States

Battle at Traitor's Castle



1K

```
10 PRINT "TRAITOR'S CASTLE"
20 LET S=0
30 FOR G=1 TO 10

40 LET R$=""
★■▲●50 LET T=INT(RND*9+1)

60 FOR L=1 TO 9
70 IF L=T THEN LET R$=R$+"O"
80 IF L<>T THEN LET R$=R$+"."
90 NEXT L

100 PRINT R$,

★■▲●110 FOR I=1 TO 12
 ★▲●120 LET I$=INKEY$

130 IF VAL("0"+I$)=T THEN GOTO 170
140 NEXT I

150 PRINT "MISSED"
160 GOTO 190
170 PRINT "GOOD SHOT"
180 LET S=S+1
190 NEXT G

200 PRINT "YOU HIT ";S;"OUT OF 10"
210 STOP
```

The King is waging a fierce and bloody battle against his deadliest enemy – the Traitor Baron. You are one of the King's crack bowmen and at this very moment you are crouching behind the bushes outside the Baron's Castle, shooting at his men as they lift their heads above the battlements.

Your computer will print a row containing eight dots and an O. The number keys 1 to 9 correspond to the position of the O in the row. You have a short time to press the correct key, and hit the O, before it disappears.

How many of the Baron's men can you hit?

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

```
   ■50 LET T=INT(RND(0)*9+1)
★▲●50 LET T=INT(RND(1)*9+1)
   ▲110 FOR I=1 TO 200
   ■110 FOR I=1 TO 60
 ★●110 FOR I=1 TO 90
   ●115 LET I$=""
★120 LET I$=INKEY$(1)
●120 IF PEEK(-16384)>127 THEN GET I$
▲120 GET I$
```

Making the game faster or slower

You may find that the computer works too quickly or too slowly for you in this game. You can adjust this by changing the last number in line 110. A lower number will make the game faster.

Puzzle corner

Can you change the program so that you get two kinds of target – either O (one of the Baron's Ordinary soldiers) which is worth 1 point, or S (one of his Special branch) which is worth 5 points?

How the program works

Sets your score to zero for start of game.

Beginning of loop which gives you 10 goes.

Sets up an empty string, R$.

Selects a number between 1 and 9 and puts it in T.

Loops round 9 times, adding a character to the string, R$, each time. The character is O for the position corresponding to the number in T and a dot for all the others.

Prints the string.

Loops round a number of times to see if you are pressing a key. (End of loop at line 140)

Checks if you are pressing the right key. Jumps to 170 if so.

Prints if you pressed the wrong key or didn't press in time, then jumps to 190.

Increases your score by 1.

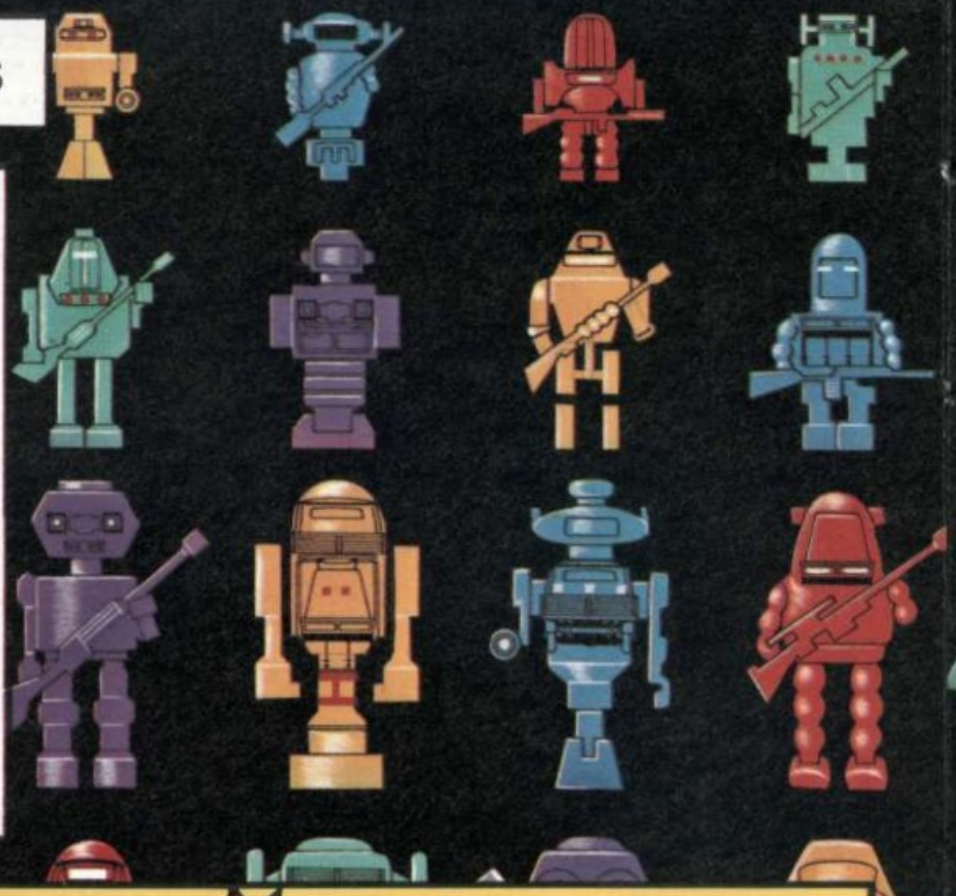
Sends computer back to line 30 for another go.

Prints your score when 10 goes are up.

Notice how the scoring works in this program (lines 20 and 180). You could try adding this to other programs.



Robot Invaders



You must act quickly. Robot invaders of all kinds are approaching. You have plenty of weapons, but for each type of Robot you must select exactly the right one for it to have any effect.

Code symbols for each Robot will flash up on your screen. Quickly press the key with that symbol on it – beware, some need the shift key too – and see how many Robot invaders you can destroy.

1K

```
    10 PRINT "ROBOT INVADERS"
    20 LET H=0
    30 FOR T=1 TO 25

★■▲●40 FOR I=1 TO INT(RND*30+20)
    50 NEXT I

★■▲●60 LET A=INT(RND*20)
★■▲●70 LET D=INT(RND*15)

s★■▲●80 LET P$=CHR$(INT(RND*53+11))
  ▲●90 CLS

   100 FOR J=0 TO D
   110 PRINT
   120 NEXT J
   130 PRINT TAB(A);P$

★■▲●140 FOR I=1 TO 15
●→★▲●150 LET R$=INKEY$
   160 IF R$=P$ THEN GOTO 210
   170 IF R$<>"" THEN GOTO 190
   180 NEXT I
   190 PRINT "MISSED"
   200 GOTO 230
   210 PRINT "A HIT"
   220 LET H=H+1
   230 NEXT T
  ▲●240 CLS
   250 PRINT "YOU SCORED ";H;"/25"
   260 STOP
```

How the program works

Sets score to zero for start.

Beginning of loop which gives you 25 goes.

Random delay.

Selects numbers for across and down positions on screen.

Chooses a keyboard character.

Clears screen

Moves cursor down the screen, one line at a time, until it reaches line number D (which was chosen in line 70).

Moves cursor A spaces across the screen and prints the character chosen in line 80 there.

Checks keyboard a number of times to see if you are pressing a key, checks if this is the right key and then jumps to the appropriate line to print hit or miss.

Increases score by 1.

Goes back for another turn.

Prints score after 25 goes.

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

```
★▲●40 FOR I=1 TO INT(RND(1)*300+200)
   ■40 FOR I=1 TO INT(RND(0)*300+200)
★▲●60,70 change RND to RND(1)
   ■60,70 change RND to RND(0)
★▲●80 LET P$=CHR$(INT(RND(1)*58+33))
   ■80 LET P$=CHR$(INT(RND(0)*58+33))
   s80 LET P$=CHR$(INT(RND*58+33))
   ●90,240 HOME
   ▲90,240 PRINT CHR$(147)
★■▲●140 FOR I=1 TO 150
   ●145 R$=""
   ★150 R$=INKEY$(1)
   ▲150 GET R$
   ●150 IF PEEK(-16384)>127 THEN GET R$
```

Speeding up the game

As you get used to playing this game, you will probably find you want to speed it up. Do this by changing the last number in line 140 to a lower one.

Moving the cursor

Your computer may have a simpler way of moving the cursor to a particular point on the screen (see lines 100-130). Check your manual and see.

Puzzle corner



These are the most dangerous of the robots.



U Mechans — V Tintroids — W Scrapions — X Wireheads — Y Steeleyes

Can you adjust the program so that you get 100 points for each of these and only 10 points for any of the others?

Secret Weapon

If you could destroy the main Robot Spare Parts Store, which lies underground somewhere in the eastern wastes of the U.R.S., you could cripple the robot attack quite severely.

You have a new secret weapon, as yet unknown to the robots, which can cut silently through solid rock, vapourizing everything in its path. The Store is very cleverly concealed though. All you can do is aim your weapon blindly and hope you get somewhere near the target.

Your computer will ask you for a difficulty number (the smallest number allowed is 4) and then ask for your guesses for the X and Y coordinates of the target. (Enter these separately, pressing RETURN, NEWLINE or ENTER after each one.)

For a clue to the possible values of X and Y, look carefully at the program listing.



1K

How the program works

```
▲●10 CLS
  20 PRINT "SECRET WEAPON"
  30 PRINT "ENTER DIFFICULTY "
  40 INPUT D
  50 IF D<4 THEN GOTO 30
★■▲●60 LET X=INT(RND*D+1)
★■▲●70 LET Y=INT(RND*D+1)
  80 FOR G=1 TO D+5

  90 PRINT "GUESSES FOR X AND Y"
 100 INPUT X1
 110 INPUT Y1
 120 LET Z=SQR((X-X1)*(X-X1)+(Y-Y1)*(Y-Y1))

 130 IF Z=0 THEN GOTO 200
 140 IF Z<=3 THEN PRINT "CLOSE"
 150 IF Z>3 THEN PRINT "NOT EVEN CLOSE"
 160 NEXT G

 170 PRINT "THE ROBOTS HAVE SEEN"
 180 PRINT "YOU - AGGHHHHH....."
 190 STOP

 200 PRINT "YOU DESTROYED IT IN "
 210 PRINT G;" GOES"
 220 STOP
```

Gets a difficulty number from you, puts it in D and checks it is not less than 4.

Selects numbers for X and Y.

Beginning of loop which gives you a number of goes depending on the difficulty you chose.

Gets your guesses and puts them in X1 and Y1.

Works out the distance between your shot and the target and puts the answer in Z.



SQR takes square roots.

Checks Z to see how close you were and prints a message. (If Z = 0 you've won.)

End of loop. Goes back for next turn.

Prints when you've used all your goes.

Prints if you were successful.



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

```
  ●10 HOME
  ▲10 PRINT CHR$(147)
★▲●60 LET X=INT(RND(1)*D+1)
  ■60 LET X=INT(RND(0)*D+1)
★▲●70 LET Y=INT(RND(1)*D+1)
  ■70 LET Y=INT(RND(0)*D+1)
```

Puzzle corner

Can you work out how to add the following scoring system?

Score 1 point for each time you are close.

Score 10 points for a win.

Escape!

The Robots have caught you, taken your weapons and locked you up. Suddenly you remember you still have your sonar wristwatch, which can be tuned to produce sounds of any frequency. If you can only find the resonant frequency of your Robot guards, they should vibrate so much they fall apart.

You must be careful not to use frequencies that are too low or the building will vibrate and collapse on top of you. If you go too high, you will get such a terrible headache you will have to give up.

Can *you* escape the horrors of the Robot prison? (Look carefully at the program for a clue to the range of frequencies to try.)




```
▲●10 CLS
  20 PRINT "ESCAPE"
★■▲●30 LET F=INT(RND*100+1)
  40 LET L=1
  50 LET H=1
  60 FOR G=1 TO 5
  70 PRINT "GUESS? ";
  80 INPUT F1
  90 IF ABS(F-F1)<5 THEN GOTO 290
 100 IF F-F1>40 THEN GOTO 170
 110 IF F1-F>40 THEN GOTO 230
 120 PRINT "NO VISIBLE EFFECT"
 130 NEXT G
 140 PRINT "YOU TOOK TOO LONG."
 150 PRINT "THE FREQ. WAS ";F
 160 STOP
 170 IF L=2 THEN GOTO 210
 180 PRINT "TOO LOW...CAREFUL"
 190 LET L=2
 200 GOTO 130
 210 PRINT "THE BUILDING COLLAPSED"
 220 STOP
 230 IF H=2 THEN GOTO 270
 240 PRINT "TOO HIGH...OUCH"
 250 LET H=2
 260 GOTO 130
 270 PRINT "YOUR HEAD ACHES - GIVE UP"
 280 STOP
 290 PRINT "YOU'VE DONE IT"
 300 STOP
```

How the program works

Chooses a number between 1 and 100 for frequency of robots and puts it in F.

Puts 1 in L and H. These are used if you go too low or too high – see lines 170-190 and 230-250.

Beginning of loop which allows you to have 5 turns.

Gets a guess from you and puts it in F1.

Checks if your guess is within 5 of F. If it is, jumps to 290 to print YOU'VE DONE IT.

Jumps to 170 if your guess is so low it is less than F by more than 40.

Jumps to 230 if your guess is so high it is more than F by more than 40

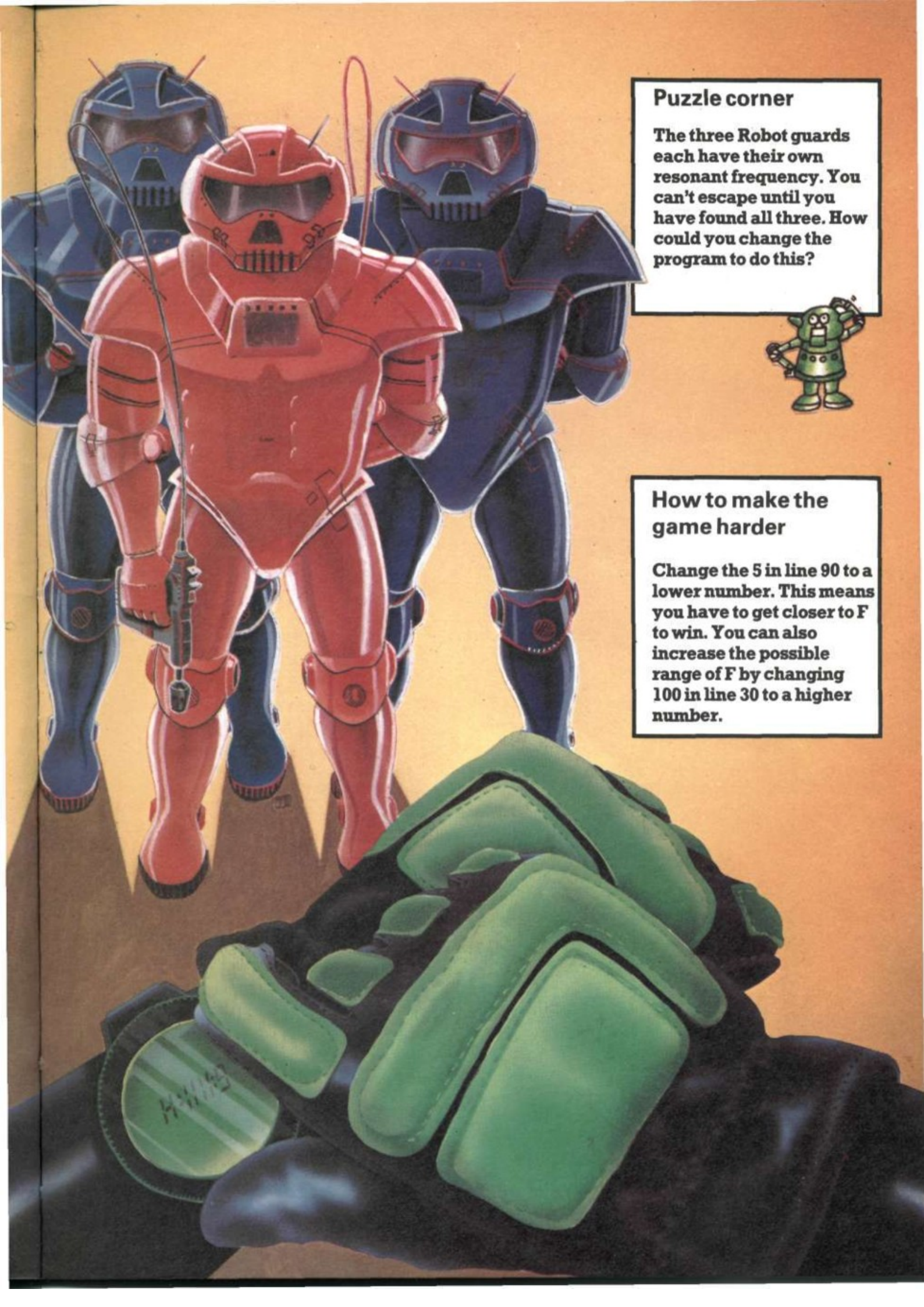
Prints if your guess was within 40 of F and goes back for next turn. If all turns have been used, it prints the answer.

Checks the value of L. The first time this part of the program is reached, L is 1. So the computer moves down the program to print a warning, change L to 2 and go back for another turn. Next time line 170 comes into operation the program jumps straight to 210 to tell you you've lost.

These lines check H in the same way to give you a warning first time you go too high and tell you you've lost the second time.

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

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



Puzzle corner

The three Robot guards each have their own resonant frequency. You can't escape until you have found all three. How could you change the program to do this?

How to make the game harder

Change the 5 in line 90 to a lower number. This means you have to get closer to F to win. You can also increase the possible range of F by changing 100 in line 30 to a higher number.

Pirate Dogfight

It's you against the Sky Pirate. He moves ahead, you accelerate – He drops behind, you slow down. You must try to get level with him and then you can fire, hoping that he won't be able to fire and hit you first.

Use the letter keys A to accelerate, D to decelerate and F to fire. Your computer will tell you your speed and position relative to the pirate. You will need to be ready to press the appropriate keys as soon as you press RUN. Keep pressing A and D until you get level and then fire.



1K

```
▲●10 CLS
  20 PRINT "PIRATE DOGFIGHT"
★■▲●30 LET V=INT(RND*11-5)

★■▲●40 LET S=-INT(RND*3+1)

  50 IF ABS(S)>20 THEN GOTO 230
▲●60 CLS
  70 PRINT "YOU ARE ";
  80 IF S<0 THEN PRINT "BEHIND"
  90 IF S>0 THEN PRINT "AHEAD"
 100 IF S=0 THEN PRINT "LEVEL"
 110 PRINT "YOU ARE GOING ";
 120 IF V>0 THEN PRINT "FASTER"
 130 IF V<0 THEN PRINT "SLOWER"
 140 IF V=0 THEN PRINT "SAME"
  ●→
★▲●150 LET I$=INKEY$
 160 IF I$="A" THEN LET V=V+1
 170 IF I$="D" THEN LET V=V-1
 180 IF I$="F" AND S=0 THEN GOTO 250
```

How the program works

Chooses a number between −5 and +5 for your speed relative to pirate and puts it in V.

Chooses a number for your distance from the pirate and puts it in S. This is negative at the start, which means you are behind him.

Checks if the distance between you is more than 20. If so, computer jumps to 230 to say you've lost sight of him.

Checks the values of S and V and prints your position and speed in relation to the pirate.

Looks to see if you are pressing a key and, if so, which one. If it is A, your speed is increased by 1. If D, your speed is decreased by 1. If you are pressing F *and* S=0 (i.e. you're level) then it jumps to 250. (If S is not 0 when you press F then nothing happens.)

```
    190 LET S=S+V
★■▲●200 FOR I=1 TO 20
    210 NEXT I
    220 GOTO 50
    230 PRINT "YOU LOST SIGHT OF HIM"
    240 GOTO 330

    250 IF ABS(V)<2 THEN GOTO 290

    260 PRINT "YOU ONLY MANAGED TO"
    270 PRINT "SCARE HIM."
    280 GOTO 40
★■▲●290 IF RND>.7 THEN GOTO 320

    300 PRINT "YOU SHOT HIM DOWN"
    310 GOTO 330
    320 PRINT "HE SHOT FIRST"
    330 STOP
```

Works out the new distance between you.

Delay loop.

Goes back to 50 to repeat.

Comes into operation if you fired when level. Checks speed is less than 2 (ignoring + and − signs) and if so, jumps to 290.

Prints a message if speed was more than 2 when you fired. Returns to 40 to start again with a new value for S.

Applies a random test to see if you fired before the pirate.

Prints a message depending on the result of the test in 290.

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

```
  ●10,60 HOME
  ▲10,60 PRINT CHR$(147)
★▲●30 LET V=INT(RND(1)*11-5)
  ■30 LET V=INT(RND(0)*11-5)
★▲●40 LET S=-INT(RND(1)*3+1)
  ■40 LET S=-INT(RND(0)*3+1)
```

```
  ●145 I$=""
  ★150 I$=INKEY$(1)
  ●150 IF PEEK(-16384)>127 THEN GET I$
  ▲150 GET I$
★■▲●200 FOR I=1 TO 200
  ■290 IF RND(0)>.7 THEN GOTO 320
★▲●290 IF RND(1)>.7 THEN GOTO 320
```



How to make the game easier

You may find this game quite difficult. To make it slightly easier, add these two lines. You will then be able to see the relative positions of the two planes printed on the screen.

```
195 IF ABS(S)>10 THEN GOTO 200
196 PRINT TAB(w/2);"HIM"
197 PRINT TAB(S+w/2);"YOU"
```

Replace w with the width of your screen.

Puzzle corner



The random test in line 290 is loaded in your favour. How could you change it so your chances of winning are equal?

Supersonic Bomber



You are on a lone supersonic bombing mission over the U.R.S. Your computer shows graphs of Robot population based on infrared photographs relayed to it by satellite. You only have time to attack one target in five, so you must quickly select the one with the highest population of Robots and release one of your "Corrodarobe" bombs on it. (These contain a substance so corrosive it can dissolve a Robot's body in seconds.)

To release a bomb, press the number key which corresponds to the number next to the graph of highest Robot population. If there are two the same, choose the one with the lowest number.

Will you be a hero when you return to base?

1K

```
▲●10 CLS
   20 PRINT "SUPERSONIC BOMBER"
   30 DIM B(5)
   40 LET S=0
```



This sets up B as an "array variable". It can hold five different values at a time, identifying them as B(1), B(2), B(3), B(4) and B(5).

Sets opening score at zero.

```
   50 FOR G=10 TO 1 STEP -1
```

Start of loop for 10 goes. Done backwards in this game (i.e. G=10 for first go, 9 for the second and so on). This allows the delay loop in line 190 to give you more time for the earlier goes.




```
   60 LET M=1
```

Sets number of highest population graph at 1 to start with.

```
   70 FOR I=1 TO 5
★■▲●80 LET B(I)=INT(RND*10+1)
   90 IF B(I)>B(M) THEN LET M=I
  100 NEXT I
```

Chooses 5 numbers and puts them in the array B(1) to B(5). Checks which is biggest and changes M to correspond with it.

```
 ▲●110 CLS
  120 FOR I=1 TO 5
  130 PRINT I;
  140 FOR J=1 TO B(I)
  150 PRINT "**";
  160 NEXT J
  170 PRINT
  180 NEXT I
```

Prints the 5 numbers on the screen in the form of rows of stars.

```
★■▲●190 FOR I=1 TO G*3
 ★▲●200 LET I$=INKEY$
  210 IF I$<>"" THEN GOTO 270
  220 NEXT I
```

Checks if you are pressing a key and goes to 270 if you are.

```
  230 PRINT "TOO LATE"
★■▲●240 FOR J=1 TO 10
  250 NEXT J
  260 GOTO 280
```

If you didn't press in time, prints TOO LATE and goes back for next go.

```
  270 IF VAL(I$)=M THEN LET S=S+1
```

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

```
  280 NEXT G
```

Goes back for another go.

```
  290 PRINT "YOU HIT ";S;" OUT OF 10"
  300 PRINT "HIGH DENSITY TARGETS"
  310 IF S=10 THEN PRINT "YOU'RE A HERO"
  320 IF S<10 THEN PRINT "TOUGH - YOU FAILED"
  330 STOP
```

Prints score after 10 goes.

How the program works

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

```
   ●10,110 HOME
   ▲10,110 PRINT CHR$(147)
 ★▲●80 LET B(I)=INT(RND(1)*10+1)
    ■80 LET B(I)=INT(RND(0)*10+1)
★■▲●190 FOR I=1 TO G*30
   ●195 I$=""
 ★200 I$=INKEY$(1)
 ▲200 GET I$
 ●200 IF PEEK(-16384)>127 THEN GET I$
★■▲●240 FOR J=1 TO 400
```

Changing the speed of the game

To give yourself more chance of pressing a key each time, change the last number in line 190 to a higher one.

As your skill improves, keep lowering the number in line 190. How low can you go and still win?

Puzzle corner



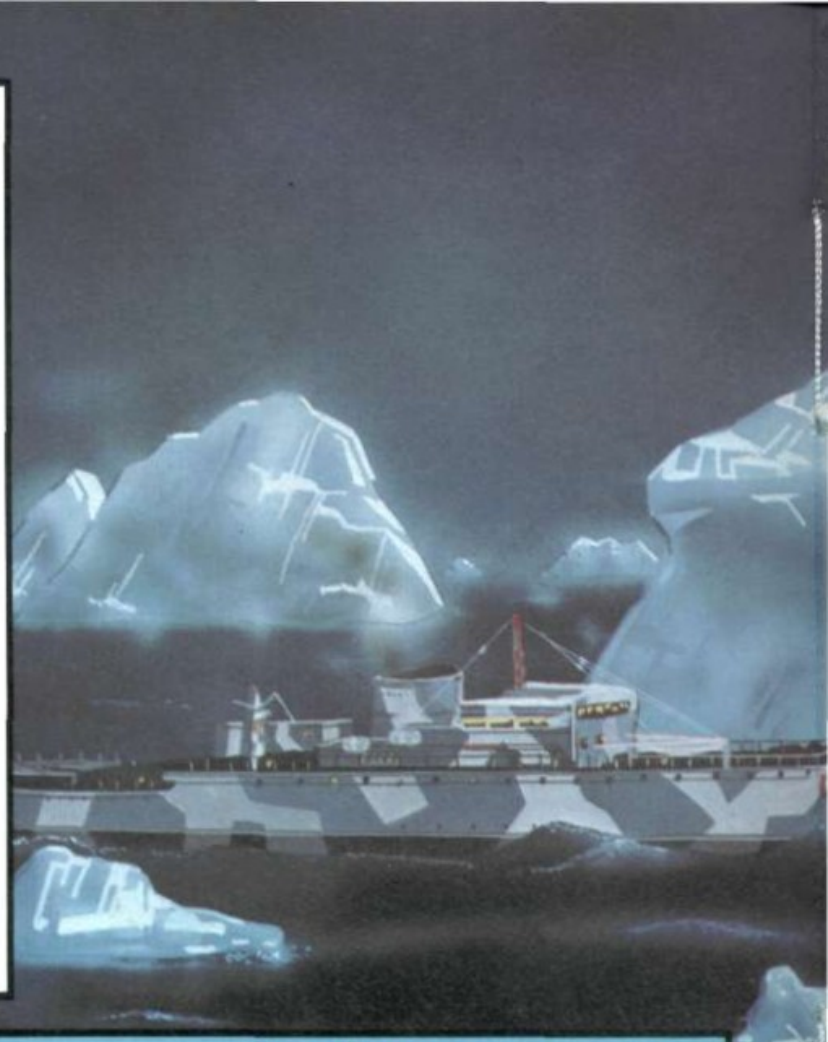
Can you work out how to make the computer give you more than 5 targets to choose from each time?

Iceberg

Your hull is badly damaged and you've no weapons to speak of. As you limp slowly home through treacherous iceberg-strewn waters, you become aware that an enemy ship is tailing you. Strangely it can detect you but not the icebergs, so your best chance is to lure it into hitting one.

Your computer will print a grid showing the position of your ship (Y), the enemy (Z) and the icebergs (*). You can move one space North, South, East or West each go. The enemy moves towards you by the most direct route (it can move diagonally too). If you move into any of the 8 positions surrounding the enemy, you will be captured, and if you hit an iceberg you will sink.

Can you escape?



```
      10 PRINT "ICEBERG"
      20 DIM B(8,8)
★■▲●30 LET N=INT(RND*8+4)
      40 FOR I=1 TO N
s★■▲●50 LET B(INT(RND*8+1),INT(RND*8+1))=23
      60 NEXT I
★■▲●70 LET SX=INT(RND*8+1)
★■▲●80 LET SY=INT(RND*8+1)
      90 IF B(SX,SY)<>0 THEN GOTO 70
s★■▲●100 LET B(SX,SY)=63
★■▲●110 LET YX=INT(RND*8+1)
★■▲●120 LET YY=INT(RND*8+1)
      130 IF B(YX,YY)<>0 THEN GOTO 110
s★■▲●140 LET B(YX,YY)=62
   ▲●150 CLS
      160 FOR Y=1 TO 8
      170 FOR X=1 TO 8
      180 IF B(X,Y)=0 THEN GOTO 210
      190 PRINT CHR$(B(X,Y));
      200 GOTO 220
      210 PRINT ".";
      220 PRINT " ";
      230 NEXT X
      240 PRINT
      250 NEXT Y
```

How the program works

This sets up the grid. B is an "array" with the DIMensions 8 by 8.

N is the number of icebergs. It varies from 4 to 11 depending on the value of RND.

23 on ZX81 and 42 on the other computers is the code for *. Computer loops round N times putting * into random positions in the grid.

This puts the enemy ship on the grid for the start of the game. 63 (ZX81) and 90 (others) are the codes for Z. Checks the position has not already been given to an iceberg and finds another if it has.

These lines do the same as above to put you on the grid. (62 and 89 are the codes for Y.)

Prints out grid with current positions of ships and icebergs.

Puzzle corner



Can you work out how to make the grid bigger?

Now work out how to add more icebergs too.


```
      260 LET B(YX,YY)=0
      270 PRINT "DIRECTION (N,S,E,W) ";
      280 INPUT D$
★■▲   290 LET YY=YY+(D$="S" AND YY<>8)
★■▲   300 LET YY=YY-(D$="N" AND YY<>1)
★■▲   310 LET YX=YX+(D$="E" AND YX<>8)
★■▲   320 LET YX=YX-(D$="W" AND YX<>1)
s★■▲● 330 IF B(YX,YY)=63 THEN GOTO 500
s★■▲● 340 IF B(YX,YY)=23 THEN GOTO 600
s★■▲● 350 LET B(YX,YY)=62
      360 LET B(SX,SY)=0
      370 LET SX=SX+SGN(YX-SX)
      380 LET SY=SY+SGN(YY-SY)
s★■▲● 390 IF B(SX,SY)=62 THEN GOTO 500
s★■▲● 400 IF B(SX,SY)=23 THEN GOTO 700
s★■▲● 410 LET B(SX,SY)=63
      420 GOTO 150
      500 PRINT "YOU'VE BEEN CAUGHT"
      510 GOTO 800
      600 PRINT "YOU'VE HIT AN ICEBERG"
      610 GOTO 800
      700 PRINT "YOU'RE SAFE - HE'S HIT ONE"
      800 STOP
```

Clears your current position from the grid so you can specify a new one.

Puts your direction in D$.

Calculates your new position, checking that you don't go over the edge of the grid.

Checks your new position to see if you have bumped into the enemy ship or an iceberg. If you have, it jumps down the program to tell you.

You've moved to a safe position and the code for your ship is put there.

Calculates the enemy's new position.

Checks enemy's new position to see if he has caught you or hit an iceberg. Jumps down program to tell you if he has.

Enemy's code is put in the new position.

End of a turn. Loops back to 150 to start another turn.

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

```
★▲●  30,50,70,80,110,120 change RND to RND(1)
  ■  30,50,70,80,110,120 change RND to RND(0)
s★■▲●50,340,400 change 23 to 42
s★■▲●100,330,410 change 63 to 90
s★■▲●140,350,390 change 62 to 89
  ▲  150 PRINT CHR$(147);
  ●  150 HOME
★■▲   290 LET YY=YY-(D$="S" AND YY<>8)
★■▲   300 LET YY=YY+(D$="N" AND YY<>1)
★■▲   310 LET YX=YX-(D$="E" AND YX<>8)
★■▲   320 LET YX=YX+(D$="W" AND YX<>1)
```

Computers can decide whether an expression is true or false. If the answer is false they give it the value 0. If it is true, some computers give the value +1, others −1. Lines 290-320 have to be changed for some computers because of this.



The Wizard's Wall

The wall the Wizard built to surround his secret stronghold is no ordinary wall. The stones in it are people, petrified by the Wizard's angry stare, and, what's more, it can move. If you can break a hole through it with your trusty catapults which fling gigantic boulders, you will destroy the Wizard's magic powers and turn the stones back into people again. There are problems though – the Wizard is working on a spell which stops missiles in mid-air and sends them back where they came from. Sometimes he can make it work . . .

If you don't succeed, you will become just another stone in the Wizard's Wall!!!!

```
▲●10 CLS
  20 PRINT "WIZARD'S WALL"
  30 PRINT
  40 PRINT "DO YOU WANT ANY HELP?"
  50 INPUT I$
★■▲●60 IF I$(1)="Y" THEN GOSUB 740
  70 PRINT "DIFFICULTY?"
  80 PRINT "(5=EASY, 1=DIFFICULT)"
  90 INPUT Q
  100 DIM W(8,8)
  110 DIM E(8)
  120 FOR Y=1 TO 8
  130 FOR X=1 TO 4
★■▲●140 LET W(X,Y)=20
  150 NEXT X
  160 NEXT Y
  170 LET Z=0
  180 FOR C=1 TO 3
  190 LET D=INT(RND*80+21)
  200 GOSUB 430
  210 IF Z=1 THEN GOTO 1150
  220 PRINT "YOU ARE ";D;" YARDS AWAY"
★■▲●230 LET W=INT(RND*41)-20
  240 IF W=0 THEN PRINT "NO WIND"
  250 IF W<0 THEN PRINT "WIND TO RIGHT"
  260 IF W>0 THEN PRINT "WIND TO LEFT":
  270 PRINT "GIVE ELEVATION (1-90)"
  280 INPUT A
  290 PRINT "GIVE SPEED"
```

```
    300 INPUT V
    310 LET A=A/180*3.1416
    320 LET H=TAN(A)*(D-W)-(5*(D-W)^2)/(V*COS(A))^2
    330 LET H=INT(H/Q)
    340 IF H>0 AND H<9 THEN GOSUB 540
    350 IF H<1 THEN PRINT "SHOT WAS TOO NEAR"
    360 IF H>8 THEN PRINT "SHOT WAS TOO FAR"
★■▲●370 IF RND>.2 THEN GOTO 200
    380 PRINT "THE WALL HAS MOVED..."
    390 NEXT C
    400 PRINT "YOU HAVE BEEN TURNED"
    410 PRINT "INTO STONE"
    420 STOP
★■▲●430 FOR I=1 TO 30
    440 NEXT I
  ▲●450 CLS
    460 PRINT
    470 FOR Y=8 TO 1 STEP -1
    480 FOR X=1 TO 8
★■▲●490 PRINT CHR$(W(X,Y));
    500 NEXT X
    510 PRINT
    520 NEXT Y
    530 RETURN
    540 GOSUB 650
    550 LET W(E(H)-1,H)=0
    560 IF V*COS(A)>50 THEN LET W(E(H)-2,H)=0
★■▲●570 IF RND>.5 AND H<>1 THEN LET W(E(1),1)=29
★■▲●580 IF RND>.5 AND H>5 AND H<8 THEN LET
        W(E(H+1)-1,H+1)=0
    590 IF H>1 THEN GOTO 630
    600 FOR Y=2 TO 8
★■▲●610 IF RND<.5 THEN LET W(E(Y)-1,Y)=0
    620 NEXT Y
    630 GOSUB 650
    640 RETURN
    650 FOR Y=1 TO 8
    660 LET X=1
    670 IF W(X,Y)=0 OR X=8 THEN GOTO 700
    680 LET X=X+1
    690 GOTO 670
    700 LET E(Y)=X
    710 IF X=1 THEN LET Z=1
```

Program continued on next page.

The Wizard's Wall continued

```
720 NEXT Y
730 RETURN
740 PRINT "YOU ARE ATTACKING THE"
750 PRINT "LAST STRONGHOLD OF THE"
760 PRINT "NOTORIOUS WIZARD, WHO"
770 PRINT "IS HIDDEN BEHIND AN"
780 PRINT "ENDLESS STONE WALL,"
790 PRINT "EACH STONE BEING ONE"
800 PRINT "OF HIS FORMER VICTIMS."
810 PRINT "ONLY YOU CAN ATTACK,"
820 PRINT "AND FREE THEM FROM"
830 PRINT "HIS MAGIC."
840 PRINT "YOU MUST DESTROY THE"
850 PRINT "WALL USING CATAPULTS,"
860 PRINT "BUT BEWARE, THE WIZARD"
870 PRINT "HAS THE POWER TO MOVE"
880 PRINT "THE WALL BACK AND"
890 PRINT "FORTH, AND OCCASIONALLY,"
900 PRINT "TO DEFLECT YOUR SHOTS"
910 PRINT "BACK AT YOU."
920 GOSUB 1110
930 PRINT "AFTER EACH SHOT, YOU ARE"
940 PRINT "SHOWN A CROSS-SECTION"
950 PRINT "OF THE WALL, SHOWING"
960 PRINT "HOW MUCH DAMAGE THERE"
970 PRINT "IS."
980 PRINT "NOTE THERE ARE CERTAIN"
990 PRINT "KEY STONES THAT PRODUCE"
1000 PRINT "LOTS OF DAMAGE, AND"
1010 PRINT "ALSO, THE FASTER THE"
1020 PRINT "BOULDER IS MOVING"
1030 PRINT "HORIZONTALLY, THE MORE"
1040 PRINT "DAMAGE IT WILL CAUSE."
1050 PRINT "CAN YOU DEFEAT THE"
1060 PRINT "WIZARD IN TIME TO SAVE"
1070 PRINT "THE THOUSANDS OF TRAPPED"
1080 PRINT "SOULS....."
1090 GOSUB 1110
```

```
   1100 RETURN
   1110 PRINT "PRESS A KEY ";
★▲●1120 IF INKEY$="" THEN GOTO 1120
 ▲●1130 CLS
   1140 RETURN
   1150 PRINT "YOU MANAGED TO BREAK"
   1160 PRINT "A HOLE IN THE WIZARD'S"
   1170 PRINT "WALL - YOU HAVE BEATEN"
   1180 PRINT "HIS MAGIC POWERS, AND"
   1190 PRINT "FREED HIS VICTIMS."
   1200 STOP
```

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)
   ●10,450,1130 HOME
   ▲10,450,1130 PRINT CHR$(147);
★■▲●60 IF LEFT$(I$,1)="Y" THEN GOSUB 740
★■▲●140 LET W(X,Y)=29
 ■▲●430 FOR I=1 TO 200
   ★430 FOR I=1 TO 1000
★■▲●490 PRINT CHR$(W(X,Y)+32);
   ★1120 I$=GET$
   ●1120 GET I$
   ▲1120 GET I$ : IF I$="" THEN GOTO 1120
```

Changes you can make to the game

You can make the wall thicker by changing the following lines:

Line 100: change the first number to a higher one, say, 10.

Line 130: subtract 4 from the number you used in line 100 and put the answer at the end of line 120. (i.e. in this case use 6.)

Line 480: change the last number to the number you used in line 100 (10 again in this case).

If you make the wall much thicker, you will probably need more time. Change the last number in line 180 to a higher number to do this.

Missile!

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 Missile!

You have three missile bases, each capable of launching one missile. When you see a plane approaching, you must judge its height and speed and fire your missiles at it one by one.

Your missiles are launched by pressing any key. The first time you press launches the left-hand one, second time the middle one and third time the right-hand one.

See how many enemy planes you can shoot down.

Missile!: TRS-80 version

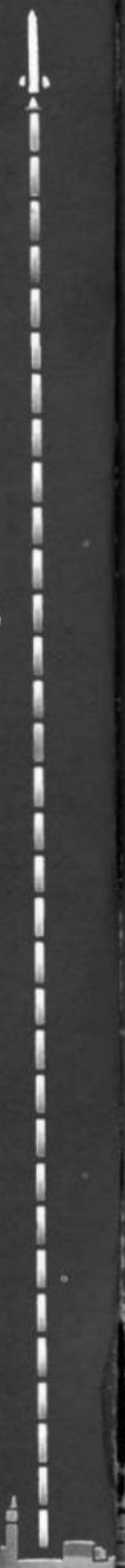
```
10 CLS
20 DIM Y(3),F(3)
30 N=1
40 PS=INT(RND(0)*3+1)
50 P=INT(RND(0)*36+5)
60 GOSUB 400
70 FOR I=PS TO 100 STEP PS
80 GOSUB 300
90 F$=INKEY$
100 IF F$="" OR N>3 THEN 130
110 F(N)=1
120 N=N+1
130 FOR J=1 TO 3
140 RESET(32*J,46-Y(J))
150 IF F(J)=0 OR Y(J)>45 THEN 190
160 Y(J)=Y(J)+1
170 IF POINT(32*J,46-Y(J)) THEN 230
180 SET (32*J,46-Y(J))
190 NEXT
200 NEXT
210 PRINT @0,"MISSED"
220 END
230 PRINT @0,"HIT!!!"
240 END
300 RESET (I-PS,P) : RESET (I-PS+1,P)
310 SET (I,P) : SET (I+1,P)
320 RETURN
400 FOR J=1 TO 3
410 SET (J*32,47)
420 SET (J*32+1,47)
430 NEXT
440 RETURN
```

Missile!: BBC version

```
10 MODE 5
20 VDU 23,224,224,160,144,144,143,128,128,255
30 VDU 23,225,0,0,0,0,240,12,2,255
40 VDU 23,226,16,56,84,16,16,16,0,0
50 VDU 23,227,0,0,0,8,8,8,8.60
60 DIM Y(3),F(3)
70 N=1:MS=16
80 PS=RND(20)+10
90 P=RND(500)+400
100 PROCDISPLAY
110 FOR I=PS TO 1100 STEP PS
120 PROCPLANE(I-PS,P,0) : PROCPLANE(I,P,3)
130 F$=INKEY$(0)
140 IF F$="" OR N>3 THEN 170
150 F(N)=TRUE
160 N=N+1
170 FOR J=1 TO 3
180 IF NOT F(J) THEN 240
190 PROCMISSILE(J,0)
200 Y(J)=Y(J)+MS
210 IF Y(J)<1024 THEN PROCMISSILE(J,3)
220 X=J*320-I : Y=Y(J)+32-P
230 IF X<128 AND X>-40 AND Y>-32 AND Y<2 THEN 280
240 NEXT
250 NEXT
260 CLS : PRINT "MISSED!!!!"
270 END
280 PROCPLANE(I,P,1) : SOUND 0,-15,5,20
290 END
300 DEF PROCPLANE(X,Y,C)
310 GCOL 0,C
320 MOVE X,Y
330 VDU 5,224,225,4
340 ENDPROC
350 DEF PROCMISSILE(N,C)
360 GCOL 0,C
370 MOVE 320*N,32+Y(N)
380 VDU 5,226,4
390 ENDPROC
400 DEF PROCDISPLAY
410 FOR I=1 TO 3
420 MOVE I*320,32
430 VDU 5,227,8,226,4
440 NEXT
450 ENDPROC
```

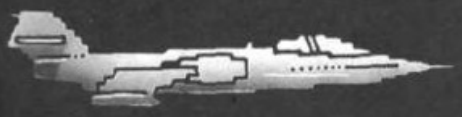
Missile!: ZX Spectrum version

```
10 CLS
15 INVERSE 0
20 DIM y(3)
30 DIM f(3)
40 LET n=1
50 LET ms=8
60 LET ps=INT(RND*6+5)
70 LET p=INT(RND*140+20)
80 GOSUB 400
90 FOR i=ps TO 240 STEP ps
100 LET c=1 : LET x=i-ps
110 GOSUB 300
120 LET c=0 : LET x=i
130 GOSUB 300
140 LET f$=INKEY$
150 IF f$="" OR n>3 THEN GOTO 170
160 LET f(n)=1 : LET n=n+1
170 FOR j=1 TO 3
180 LET c=1 : GOSUB 350
190 IF f(j)=0 OR y(j)>148 THEN GOTO 240
200 LET y(j)=y(j)+ms
210 LET c=0 : GOSUB 350
220 LET x=j*64-i : LET y=p-y(j)
230 IF x>-1 AND x<12 AND y<10 AND y>-5 THEN GOTO 280
240 NEXT j
250 NEXT i
260 PRINT AT 0,0;"Missed"
270 STOP
280 PRINT AT 0,0;"Hit!!!"
290 STOP
300 INVERSE c
310 PLOT x,p
320 DRAW 0,8 : DRAW 3,-6
330 DRAW 8,0 : DRAW 2,-2
340 DRAW -13,0 : RETURN
350 INVERSE c
360 PLOT 64*j+4,Y(j)+4
370 DRAW 0,6 : DRAW -2,-2
380 DRAW 2,2 : DRAW 2,-2
390 RETURN
400 FOR a=65 TO 66
410 FOR b=0 TO 7
420 READ c
430 POKE USR CHR$(a)+b,c
440 NEXT b
450 NEXT a
460 FOR j=1 TO 3
470 PRINT AT 21,8*j;"[AB]"
480 NEXT j
490 RETURN
500 DATA 0,0,0,0,255,255,
    127,63
510 DATA 0,252,252,252,
    255,255,254,252
```



Missile!: VIC 20 version

```
10 PRINT CHR$(147)CHR$(5);
20 POKE 36879,8
60 DIM Y(3),F(3)
70 N=1:MS=2
90 P=INT(RND(1)*9+2)*2
110 FOR I=1 TO 21 STEP RND(1)/2+.5
120 GOSUB 300
130 GET F$
140 IF F$="" OR N>3 THEN 170
150 F(N)=-1
160 N=N+1
```



```
170 FOR J=1 TO 3
180 IF F(J)=0 THEN 220
190 POKE 8164+J*5-Y(J)*22,32
200 IF Y(J)=22 THEN 240
210 Y(J)=Y(J)+1
220 POKE 8164+J*5-Y(J)*22,30
230 IF ABS(I-J*5)<=1 AND P=Y(J) THEN 280
240 NEXT
250 NEXT
260 PRINT CHR$(147);"MISSED!!!!"
270 END
280 PRINT CHR$(147);"HIT!!!!"
282 POKE 36877,220 :  POKE 36878,15
284 FOR K=1 TO 500 : NEXT
286 POKE 36877,0 : POKE 36878,0
290 END
300 POKE 8163+I-P*22,32
310 POKE 8164+I-P*22,121
320 RETURN
```

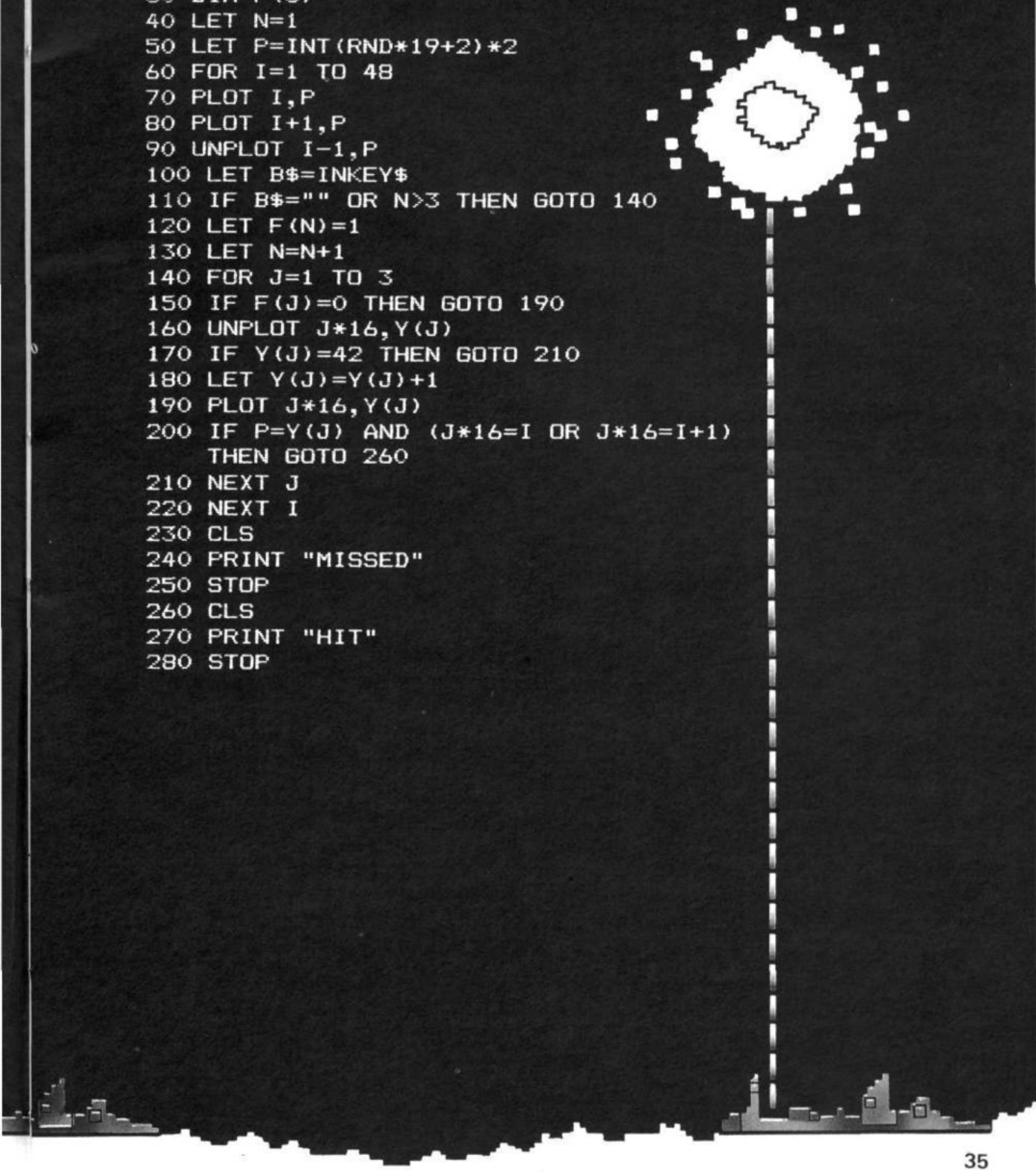
Special note for Spectrum users:
The convention used for showing graphics characters in a PRINT statement is as follows: The character [means press the GRAPHICS key once. Then type the following letter keys (shifted for the ZX81). Press the GRAPHICS key again when] is shown.

Missile!: Apple version

```
10 HOME
20 HGR
30 HCOLOR=3
40 DIM Y(3),F(3)
50 N=1 : MS=5
60 PS=INT(RND(1)*6+4)
70 P=INT(RND(1)*135+11)
80 GOSUB 400
90 FOR I=PS TO 265 STEP PS
100 X=I-PS : Y=159-P : C=0 : GOSUB 300
110 X=I : C=3 : GOSUB 300
120 F$="" : IF PEEK(-16384)>127 THEN GET F$
130 IF F$="" OR N>3 THEN 160
140 F(N)=1
150 N=N+1
160 FOR J=1 TO 3
170 C=0 : GOSUB 350
180 IF F(J)=0 OR Y(J)>145 THEN 230
190 Y(J)=Y(J)+MS
200 C=3 : GOSUB 350
210 X=J*70-I : Y=P-Y(J)
220 IF X>-1 AND X<15 AND Y>-9 AND Y<5 THEN 270
230 NEXT
240 NEXT
250 VTAB 22 : PRINT "MISSED"
260 END
270 VTAB 22 : PRINT "HIT!!!"
280 END
300 HCOLOR=C
310 HPLOT X,Y TO X,Y-8
320 HPLOT TO X+3,Y-2 : HPLOT TO X+12,Y-2
330 HPLOT TO X+14,Y : HPLOT TO X,Y
340 RETURN
350 HCOLOR=C
360 HPLOT 70*J,158-Y(J) TO 70*J,154-Y(J)
370 RETURN
400 FOR J=1 TO 3
410 HPLOT 70*J-5,159 TO 70*J+5,159
420 NEXT
430 RETURN
```

Missile!: ZX81 version

```
10 CLS
20 DIM Y(3)
30 DIM F(3)
40 LET N=1
50 LET P=INT(RND*19+2)*2
60 FOR I=1 TO 48
70 PLOT I,P
80 PLOT I+1,P
90 UNPLOT I-1,P
100 LET B$=INKEY$
110 IF B$="" OR N>3 THEN GOTO 140
120 LET F(N)=1
130 LET N=N+1
140 FOR J=1 TO 3
150 IF F(J)=0 THEN GOTO 190
160 UNPLOT J*16,Y(J)
170 IF Y(J)=42 THEN GOTO 210
180 LET Y(J)=Y(J)+1
190 PLOT J*16,Y(J)
200 IF P=Y(J) AND (J*16=I OR J*16=I+1)
    THEN GOTO 260
210 NEXT J
220 NEXT I
230 CLS
240 PRINT "MISSED"
250 STOP
260 CLS
270 PRINT "HIT"
280 STOP
```

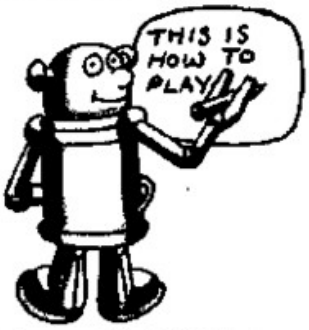


Adding to the programs

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
```

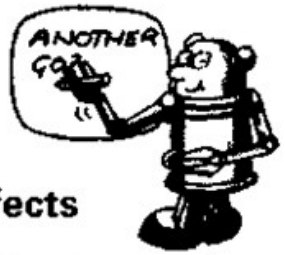
You could combine this with the instruction sub-routine by taking lines 11 to 17 from the instructions section on this page and putting them at lines 20 to 26 of this program. You can then start the main program at line 30 and add the instruction sub-routine at the end.

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.



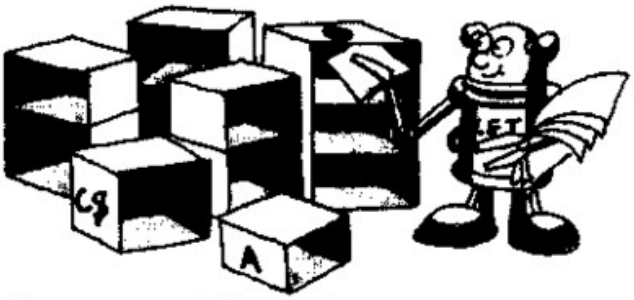
Special note for BBC and Spectrum users

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.

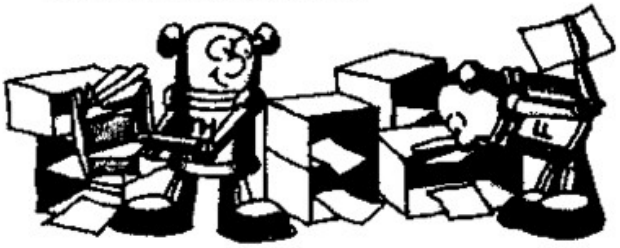
Before you start, it is a good idea to stop and think about what your computer can and cannot do.



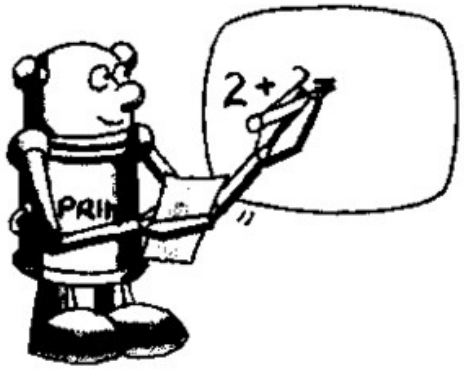
*It can store information



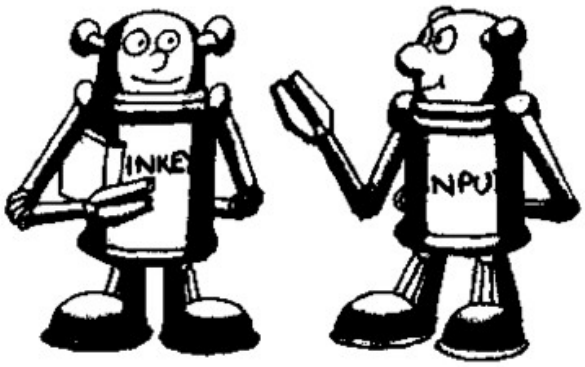
*It can do calculations.



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



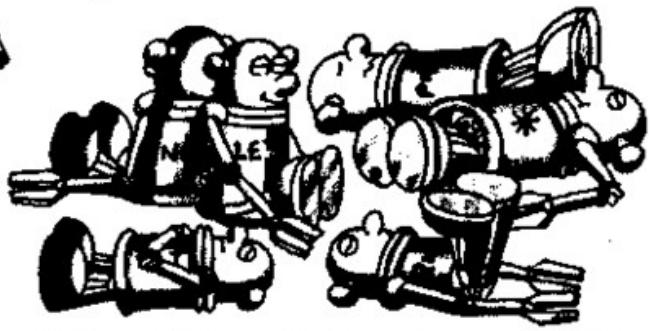
*It can tell you the results of its calculations and decisions and also what is stored in its memory.



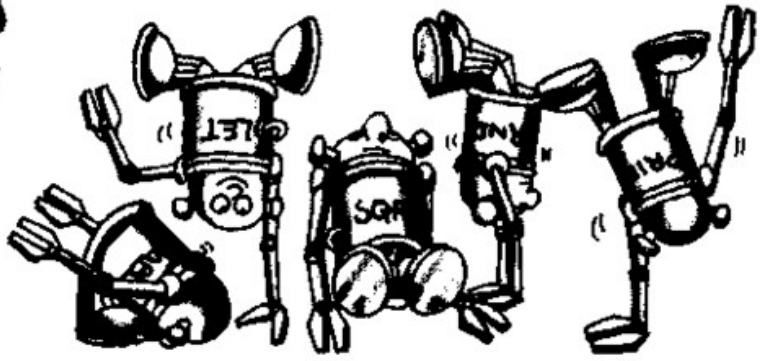
*It can ask you for information.



*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, notto 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

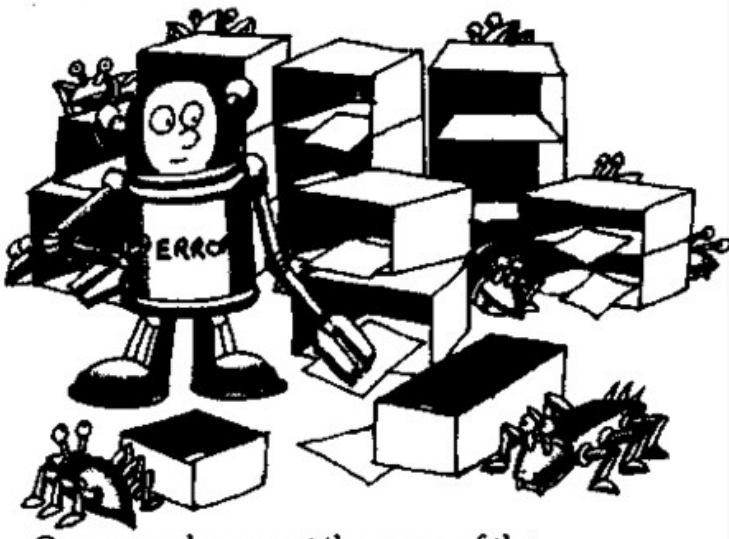
6) PRINT MESSAGE DEPENDING ON ACCURACY OF SHOT

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.

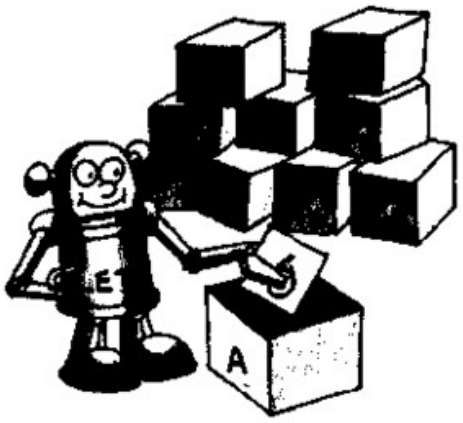


Once you have got the core of the program working, you can add to it. Scoring, extra comments, more targets etc. can all be incorporated later. You could add sections from the programs in this book to your games.

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.

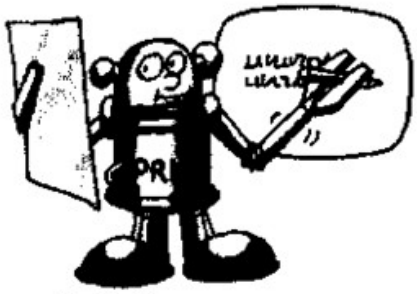
Summary of BASIC

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.





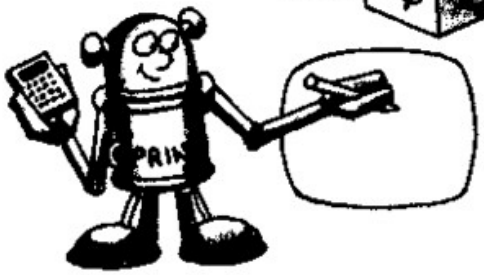
PRINT tells the computer to display things on the screen and you can use it in several ways:

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.

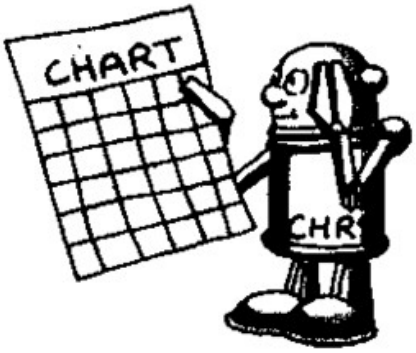


CHR$ converts numbers into letters. Apart from the ZX81, all the computers in this book use the ASCII* set of keyboard characters in which each character corresponds to a certain number. E.g. letter A has the code number 65 and PRINT CHR$(65) will display an A on the screen.

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

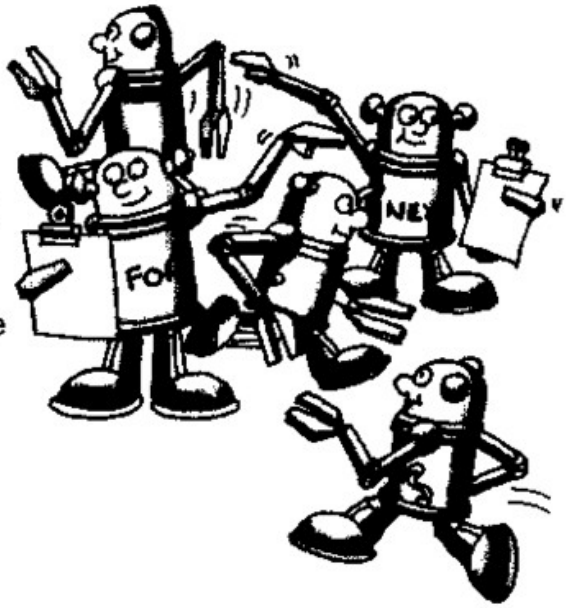
CHR$(INT(RND*26+65))

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.

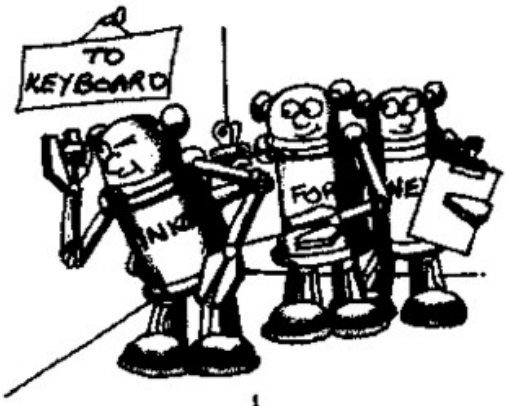


INPUT labels a space in the computer's memory, prints a question mark and then waits for you to type something which it can put in this memory space. It will not carry on with the rest of the program until you press RETURN, ENTER or NEWLINE.

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

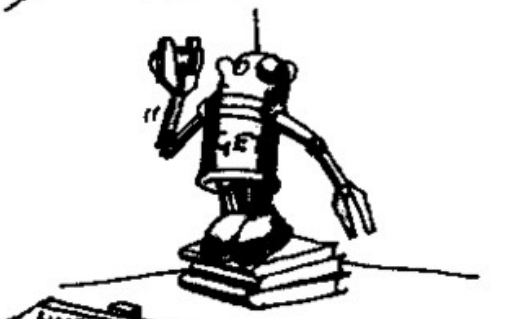


*American Standard Code for Information Interchange (see page 45)



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.

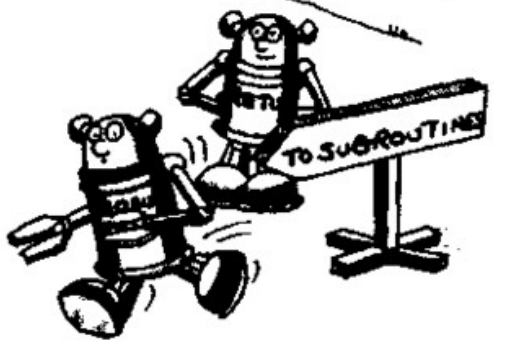
If you haven't pressed a key before the loop finishes, the computer carries on with a string containing nothing (called a "null" string).
NB Apple and VIC do not use INKEY$.



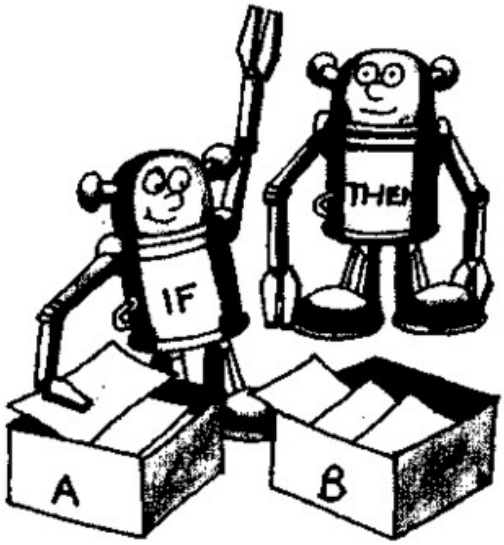
GET is used instead of INKEY$ on VIC and Pet computers.



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.

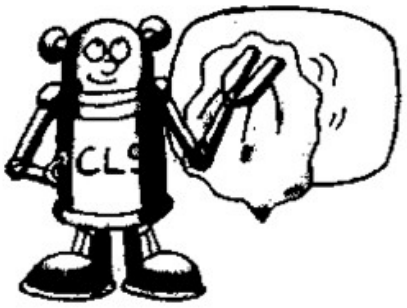


IF . . . THEN tells the computer to decide if an expression is true or false, and do different things depending on the answer. It is used with the following signs, and also with AND or OR:

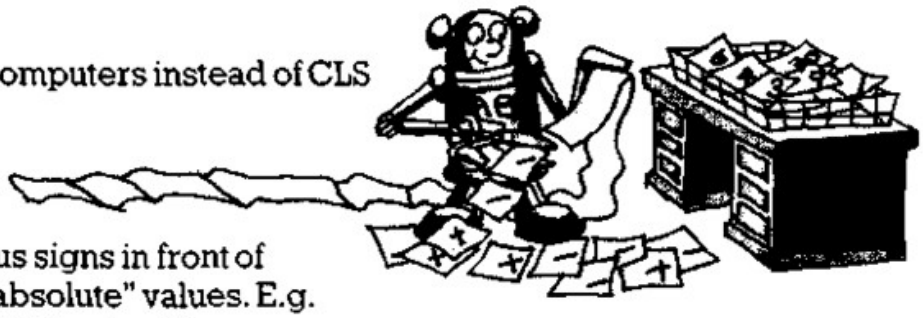
=the same as
<less than
>greater than
<=less than or the same as
>=greater than or the same as
<>not the same as

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).

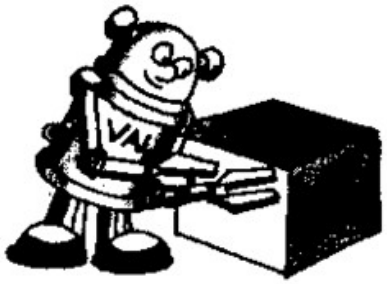


HOME is used by Apple computers instead of CLS to clear the screen.

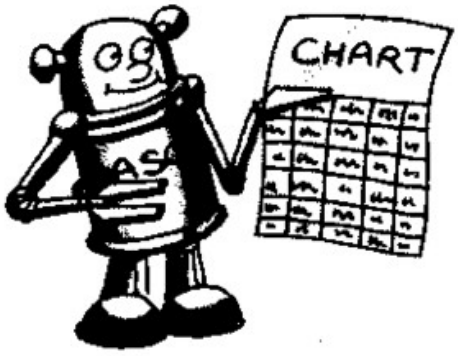


ABS ignores plus and minus signs in front of numbers and takes their "absolute" values. E.g. ABS(−10) is 10 and ABS(+10) is also 10.

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.

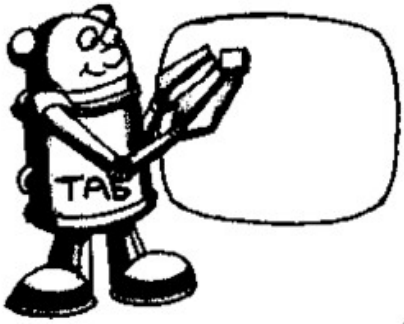


ASC converts a character into its ASCII code number e.g. ASC("3") gives 51. The expression in brackets must be a string e.g. ASC(A$) or ASC("20").
NB ZX81 and ZX Spectrum do not use ASC, though the Spectrum does use the ASCII code.



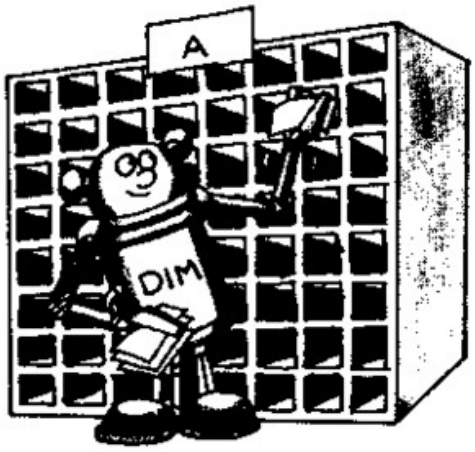
CODE is used by ZX81 and Spectrum in place of ASC. Like ASC it must always be followed by a string. Remember that the ZX81 uses different code numbers from the other computers.

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.





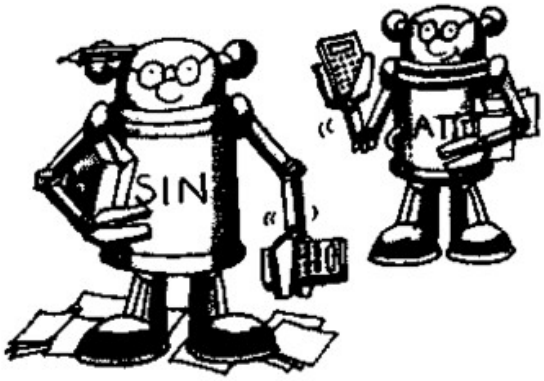
SGN tells the computer to find out the sign of a number. It produces −1 for a negative number, 0 for zero and +1 for positive numbers. E.g. SGN(−30) is −1, SGN(7) is +1 and SGN(0) is 0.



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.

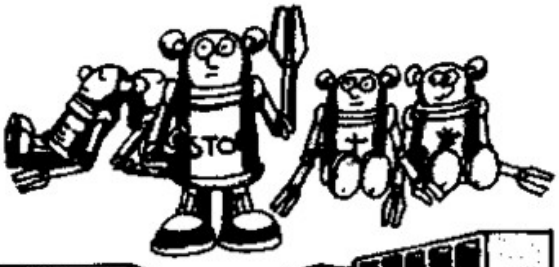


SQR takes square roots of numbers. E.g. SQR(16) gives the answer 4.



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 Hypoteneuse (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
Robot Missile

Line 90 tells the computer how many times to loop round and get a guess from you. So, for more chances of guessing the secret code letter, change the last number in line 90 to a higher one. For less chances, change it to a lower one.

Page 7
The Vital Message

In this program, lines 150 and 160 are a "delay" loop. They make the computer do nothing for a certain length of time before going on to the next instruction in the program which is to clear the message off the screen. To make the message stay on the screen longer, you need to make the computer loop round more times. You can do this by changing the last number in line 150 to a higher one.

Page 9
Shootout

To make it possible for the computer to miss too, add the following lines in addition to those in the 'Making the game harder" box.

```
155 IF RND<.1 THEN GOTO 250
250 PRINT "HE SHOOTS BUT MISSES"
260 GOTO 90
```

Line 155 may need RND(1) or RND(0), depending on your computer.

Page 11
Desert Tank Battle

Add the following lines to the program to add the possibility of the robots seeing you:

```
175 IF RND<.05 THEN GOTO 250
250 PRINT "THEY'VE SEEN YOU-AABBGHHHH"
260 STOP
```

Change the RND in line 175 to the version your computer needs. You can change .05 to any number between 0 and 0.999, but remember that the higher the number you put here the greater the chance of the robots seeing you.

Page 13
Battle at Traitor's Castle

This is a difficult puzzle, so don't worry if you couldn't do it. Try out the answer anyway. Add these lines to get the two targets.

```
 55 LET P=INT(RND+.5)*4+1
 57 LET P$=CHR$(P+51)
 57 LET P$=CHR$(P+78)
 70 IF L=T THEN LET R$=R$+P$
180 LET S=S+P
```

Page 15
Robot Invaders

You can get 100 points for U, V, W, X or Y and 10 points for the others by making these changes.

```
220 LET H=H+10
225 IF P$>"T" AND P$<"Z" THEN LET
      H=H+90
```

Page 17
Secret Weapon

You can add the scoring system in the

puzzle by putting these extra lines into the program.

```
15 LET S=0
145 IF Z<=3 THEN LET S=S+1
190 GOTO 217
215 LET S=S+10
217 PRINT "YOU SCORED ";S;" POINTS"
```

Page 19
Escape!

This is the simplest way to add three robots to the game, though it makes the game rather difficult to win.

```
22 FOR R=1 TO 3
25 PRINT "ROBOT ";R
300 NEXT R
310 STOP
```

Page 21
Pirate Dogfight

To make your chance of hitting the pirate equal to his chance of hitting you, change the .7 in line 290 to .5.

Page 23
Supersonic Bomber

To get more targets each time, change the 5 in lines 30, 70 and 120 to a higher number. You must put the same number in all three lines.

Page 25
Iceberg

You can make the grid bigger by changing all the 8s in lines 20, 50, 70, 80, 110, 120, 160, 170, 290 and 310 to a bigger number. (Use the same number each time.)

To add more icebergs change the 4 in line 30 to a higher number.

Going further

Here is a list of books you should find useful if you want to find out more about computers or writing programs.

Usborne Guide to Computers by Brian Reffin Smith, Usborne
Understanding the Micro by Judy Tatchell and Bill Bennett, Usborne
Introduction to Computer Programming by Brian Reffin Smith, Usborne
Illustrating BASIC by Donald Alcock, Cambridge University Press
Fred Learns about Computers, Macdonald & Evans
The BASIC Handbook by David A. Lien, Compusoft Publishing
The Computer Book by Robin Bradbeer et al, BBC Publications

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 Spain by
ELEXPURU, S. A. L. – Bilbao

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.**

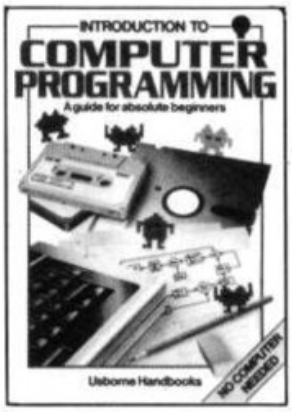




Other Computer Titles



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 685 8

Printed in Spain by ELEXPURU, S. A. L.—Bilbao