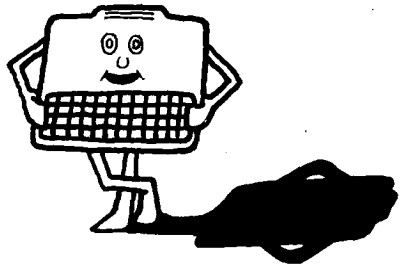
$10.95

MAKING THE MOST OF YOUR ZX 81

TIM HARTNELL

Making the most of your ZX81



by

Tim Hartnell

Reston Publishing Co, Inc.
A Prentice-Hall Company
Reston, Virginia.

This book was set in Oracle by
The Drawing Room and printed by
Westcreté Ltd of Ashford, Middx.
England.

0-8359-4189-2
0-8359-4188-4 (Pbk)

Published in the United States and Canada by:
Reston Publishing Co, Inc.
A Prentice-Hall Company
Reston, Virginia.

Sales restricted to the United States, its territories and possessions, and Canada.

Contents

NOTE: Some ZX81's have a NEWLINE key to enter commands. On most later machines, the key is labelled ENTER. Use ENTER where NEWLINE or N/L is referred to.

Foreword

Getting hold of a ZX81 is the first experience many people have had with a computer. And even a computer which helps the user as much as a ZX81 does, can be bewildering to a first time user.

A computer is not like a stereo record player or a video recorder. You can't just turn it on and expect it to do something. A computer must be treated like a very stupid child. It must be told what to do, in precise terms. It will then follow the instructions it has been given exactly.

But you have to know how to enter those instructions.

This is where Tim's book can be a great asset. If you turn on your ZX81, and enter each program as you come to it, you'll have a lot of fun and will — almost without knowing it — end up knowing quite a bit about programming.

MARK CHARLTON
Author "THE GATEWAY GUIDE TO THE ZX81 AND ZX80".

Introduction

I well remember the day in April, 1980, when my ZX80 arrived in the post. I rushed home from work and started entering the programs given in the manual. And I was not too impressed. I knew nothing about computers at that stage, and only learned with a fair degree of difficulty how to program, and what could be done with a ZX80.

By the time Clive Sinclair released the ZX81 in March 1981, I had developed some proficiency in programming, and eagerly turned to this new, more flexible computer, to see what could be done with it.

And I was not disappointed. The ZX81 is a remarkable piece of technology, compressing an awesome amount of computing power into a very small space. The range of applications is practically infinite, and the programming skill you'll acquire on your ZX81 will stand you in good stead no matter which computer system you come across in the future.

I've written this book to help you get started, to help you progress rapidly and painlessly from the stage of knowing practically nothing about programming — as I did when I began — to the point where you'll soon be writing your own games and other programs, doubtlessly impressing your family and friends in the process.

And even if you do know a bit about programming, you're sure to find things of interest in this book. The more BASIC you know, the further you may wish to read before turning your ZX81 on.

Thank you to Colin Hughes, Toni Baker, Trevor Sharples and Ian Beardsmore for helping me with the contents of this book. Thanks also to members of the National ZX80 and ZX81 Users' Club (44-46 Earls Court Road, London, W8 6EJ) who've shown such enthusiasm for the ZX81 and friendliness in the months I've been associated with the club.

I sincerely hope this book does, indeed, help you in MAKING THE MOST OF YOUR ZX81.

Tim Hartnell
London, December 1981

ZX81 Graphics and Symbols

If you look at the keyboard of your ZX81 you'll notice that some of the keys have strange symbols in squares printed on them -the numbers 1 to 8 for example, or the keys Q to Y and A to H. These are GRAPHICS symbols, and we shall be using these in many of the programs in this book. They are obtained by using the GRAPHICS key (shift 9) and then pressing the required key with SHIFT.

To save us drawing pretty pictures of squares throughout the book we shall simply write what we mean in small letters, so for example wherever you see the phrase graphic T we mean the graphics symbol on the T key. Graphic 4 will mean the graphics symbol on the 4 key, and so on. Notice that the digit 4 was used rather than spelling out four in letters.

When graphics symbols are repeated we shall simply write this out in small letters as well, so that five graphic A means the graphics symbol adorning the A key is to be repeated five times.

Another use of the GRAPHICS key is to produce inverse characters. We shall usually write the word inverse in small letters when these symbols are needed, so that inverse A is a white letter A on a black background. Note that this is not the same thing as graphic A. The SHIFT symbols such as $ and ? can also be inverse - again we will simply write inverse $ or inverse ?. Sometimes it will be necessary to produce whole words or phrases out of inverse characters. When this happens the appropriate letters or symbols will be UNDERLINED, so "ZX81" means the same thing as "inverse Z inverse X inverse 8 inverse 1".

If we were to write "CAT graphic A DOG" we are talking about a string containing the letters C,A and T, the graphics symbol on key A, and the inverse letters D,O and G. There is no space between the T and the graphic A, nor between the graphic A and the inverse D. If spaces were needed there we would write "CAT space graphic A space DOG" Unlike some other books we shall not be using an underlined asterisk where we mean space, since underlined asterisk(*) means inverse asterisk. The word space will always be written out in full if a space is needed where it is not immediately obvious.

The "quote-token" symbol on key Q shall be written as shift Q. Whenever you see an ordinary ", or the word quote, we are referring to the symbol on key P. This cannot be included in the middle of a string, ie you cannot write PRINT """", although you

CAN write PRINT "shift Q" which will have precisely the same effect. Note that PRINT "inverse quote" is the graphics symbol on key P, not on key Q.

Inverse space will generally be written out in full, however where it is more convenient we shall just use an UNDERLINE eg PRINT "THE_CAT_SAT_ON_THE_MAT" includes inverse spaces.

Finally a brief mention of KEYWORDS. It will generally save space in your programs if in PRINT statements you use keywords like AND instead of typing out space letter A letter N letter D space. We shall indicate this by writing "keyword AND" or "keyword AT" etc. Some of the keywords like NEXT can't be obtained in a single key. If you see PRINT "keyword NEXT" you may obtain the word NEXT by typing shift 3 (THEN). Now press NEXT and the word will appear. Delete the word THEN by typing shift 5 (cursor left) shift Ø (rubout) shift 8 (cursor right) and continue typing in the rest of the line.

Random Numbers

Turn on your TV, plug in the ugly black power converter, and get the cursor (that's the white K on a black square) down in the left hand corner of the screen. Our first program uses one of the most useful programming aids for games — the random number generator.

Actually, as the manual points out, it is not a true random number generator, but it is close enough for our purposes.

RND is a random number between Ø and Ø.99999999, so to generate a random number between zero and, say, 1Ø, you just input LET J=1Ø*RND. Actually if you think about it this will give a number between Ø amd 9.9999999, but we can get round that by using the function INT. INT of a number is the largest integer not greater than that number, so for positive numbers this has the effect of chopping off all the decimal places. If RND turned out to be Ø.395Ø3486 then 1Ø*RND would be 3.95Ø3486, and INT (1Ø*RND) would be 3 - all of the decimal places were wiped out by the function INT. Now J will be at random either Ø,1,2,3,4,5,6,7,8,or 9. If we wished J to be a random integer between one and ten we would have to input LET J=INT

(1Ø*RND)+1. If you follow up this line with the command PRINT J, your ZX81 will display the number (between 1 and 1Ø) which it has generated. Notice that when you write a program, each line must be numbered (with numbers between 1 and 9999). The computer executes each line from the lowest number to the highest.

Try this program first:

```
1Ø  PRINT "NUMBER
    GENERATOR"
2Ø  LET J = INT
    (1Ø*RND)+1
3Ø  PRINT J ;" "; (just leave
    a couple of spaces
    between the quote
    marks)
4Ø  GOTO 2Ø
```

When you run this, you'll find the screen fills up after a very long delay with numbers, between 1 and 10. The program stops when it runs out of memory.

There are five things you can learn from this program:

1Ø PRINT "NUMBER GENERATOR"

This line starts with a line number (as I mentioned before). You can use any number you like (between 1 and 9999), but you'll find working in multiples of 1Ø or 2Ø will give you enough room to add new lines in between others if you need to later. Line numbers sort themselves automatically into the correct order. When you ran the program, the ZX81 acted on the lowest numbered line (in this case 1Ø) and obeyed the instruction PRINT and printed what was between the quote marks. Pretty simple, basic stuff. So from this line you have learned the use of line numbers, and of the command PRINT.

The next line:

2Ø LET J = RND (1Ø*RND)+1

This tells the computer to assign the integer value generated to the variable J (a letter, a number of letters, or a letter followed by a combination of letters and numbers can be a variable). Then when, in the next line, you ask the ZX81 to PRINT J, it prints the number which has been assigned to J. Don't worry if you can't understand this, it will become clear in due course.

The next line:

3Ø PRINT J;" ";

This line tells the computer to print the number which it has assigned to J. The semi-colon after the J, and after the second set of quote marks, ensures that the ZX81 will print the values

for J one after another, instead of starting a new line for each one.

Try typing 3Ø PRINT J, Using the NEWLINE, put this into the program in place of the original line 3Ø. Run the program, and you'll notice the numbers printed in neat little columns. The comma divides the screen up into two parts, and when it reads a comma, automatically goes to the start of the next half of the screen. (The space between the quote marks in the original program ensured there was a space between each number). Try running the program without the space, i.e. with line 3Ø PRINT J; you'll find the screen fills up with solid numbers, all running into each other. The semi-colon tells the ZX81 to go on and print the next J, without leaving a space, and without starting a new line.

The fourth line of our original program:

```
4Ø  GOTO 2Ø
```

The computer runs through the program in order, carries out the instructions in each line, and then stops. In this case, when it gets to the end of the program, line 4Ø tells it to go back to line 2Ø. It does so, then runs through the program again, and again finds the instructions to go back to line 2Ø. It stops only when you run out of screen space.

Now, you've learned that a letter, such as J, can be assigned a numerical value. What is known as a 'string' (a letter followed by a $ sign, like A$) can be assigned to a word or words (or any combination of letters, symbols and numbers, but we'll stick to words for the moment).

Clear the RAM with the instruction NEW, and input the following program:

```
1Ø  PRINT "NUMBER
    GENERATOR"
2Ø  PRINT "WHAT IS
    YOUR NAME?"
3Ø  INPUT A$
4Ø  PRINT "OK,";A$"I AM
    GOING TO WORK OUT
5Ø  PRINT "SOME
    RANDOM NUMBERS
    BETWEEN ONE
6Ø  PRINT "AND TEN."
7Ø  PAUSE 100
8Ø  LET J = INT
    (1Ø*RND)+1
9Ø  PRINT J;" ";
1ØØ GOTO 8Ø
```

Try running this program. You'll find the line 2Ø asks your name, and then accepts it in line 3Ø, assigning the string A$ to your name. The ZX81 then uses your name (A$) in the next line, line 4Ø.

The other thing to note about this program is the line 7Ø which tells the ZX81 to PAUSE for a short while before it continues with the rest of the program. (Try adjusting the number after the word PAUSE to see what happens.)This feature is a very useful one in games, and we will be using it time and time again.

You've probably cursed the fact that your ZX81 doesn't automatically'scroll' the contents of the screen upwards. It just shuts down when the screen is full, and gives you an error code.

The clear screen command (CLS) is very useful here. Looking at the output of the program you've just run, you have probably realised that all the lines spent asking WHAT IS YOUR NAME? and so on are wasted once the instruction has been followed. Add the line: 75 CLS to the program, and run it again.

The CLS, you will find, cleared the screen of everything that preceded it on the program, but didn't (as will become important later) 'forget' what had been assigned to the string, A$. That is the computer still remembers your name (A$), so long as you don't wipe the program with NEW or turn off the power, or go back into the 'command mode' (when you can see the whole program listed on the screen, numbers and all).

It is a bit butchery to just run the program until it fills the screen and jams off, giving you an error message. We can let the ZX81 count how many times it goes around the loop (that is, how many times it executes the lines 8Ø, 9Ø and 1ØØ). Add the following:

```
76  LET K = Ø
85  LET K = K + 1
95  IF K > 3Ø THEN STOP
```

Now run it, you should find the program prints 31 numbers between Ø and 1Ø and then stops. It does this by assigning the value of zero to the number variable in line 76, then adds one to K each time round (so the first time it is, in effect, LET K = Ø + 1, the second time LET K = 1 + 1, the third time LET K = 2 + 1, and so on) until K is bigger than 3Ø, when the computer STOPs. This counting and terminating feature is a very useful one.

IF/THEN: In line 95 (IF K > 3Ø THEN STOP) we used one of the ZX81's facilities to make a decision, and act on it. This IF/THEN is a very useful statement in BASIC. The form of an IF/THEN line is line number, IF something (such as IF Z = 30, or IF A = B) followed by THEN (shift 3) and another command. In ZX81 *basic* the other command can be anything (such as GOTO, LET S = 2, or STOP).

Let's get to our first real program.

Simple Decision Makers

Input the following:

```
1Ø  PRINT ,"DECISION
    MAKER"
2Ø  PRINT
3Ø  PRINT
4Ø  PRINT "THINK OF A
    QUESTION AND"
5Ø  PRINT "PRESS
    NEWLINE FOR A"
6Ø  PRINT "DECISION ON
    IT"
7Ø  INPUT Q$
8Ø  LET J = INT (3*RND)
9Ø  IF J = Ø THEN PRINT,
    "YES"
1ØØ IF J = 1 THEN
    PRINT,"NO"
11Ø IF J = 2 THEN
    PRINT,"MAYBE"
```

Run this program a few times. You'll see how the comma in lines 1Ø, 9Ø, 1ØØ and 11Ø sets the words away from the left hand side of the screen, and how lines 2Ø and 3Ø (with just the command PRINT, with nothing following it) put a space between the printout of lines 1Ø and 4Ø.

Now comes the creative bit. To dress up this program, you can do at least three things: (1) Let the ZX81 ask for, and use your name; (2) Use CLS to make the presentation cleaner; and (3) Allow you more than one go before the screen flips back into the command mode. Try and amend the program (by slipping things in between the numbered lines, and/or by changing some lines) until you can do these three things — before reading on. Cover up the following program until you've had a go at altering the original program.

This is just an example of how to modify the program. There are many, many other ways to achieve similar ends.

```
1Ø  PRINT ,"DECISION
    MAKER"
2Ø  PRINT
25  PRINT "WHAT IS
    YOUR NAME?"
27  INPUT A$
3Ø  CLS
4Ø  PRINT "THINK OF A
    QUESTION,";A$
5Ø  PRINT "PRESS
    NEWLINE, AND I
    WILL"
6Ø  PRINT "MAKE A
    DECISION ON IT"
7Ø  INPUT Q$
8Ø  LET J = INT (3*RND)
9Ø  IF J = Ø THEN PRINT
    "YES"
1ØØ IF J = 1 THEN PRINT
    "NO"
11Ø IF J = 2 THEN PRINT
    "MAYBE"
12Ø PRINT
13Ø PRINT "ANOTHER
    GO,";A$;"?"
```

```
14Ø INPUT B$
15Ø IF B$ = "YES" THEN
     GOTO 3Ø
16Ø CLS
17Ø PRINT "OK,";A$;",BYE
     BYE"
```

Note the way the computer is asked to check if B$ = "YES" (in line 15Ø). If it finds that B$ *is* equal to YES, control goes back to line 3Ø. There is no need to put in a line IF B$ = "NO" THEN GOTO 160 because the ZX81 moves on until it comes to another instruction.

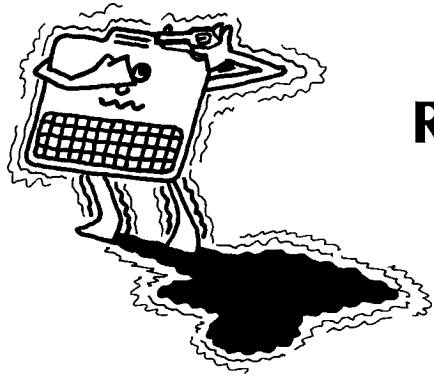
There is another refinement we could add to this program. Try and work out how to make the amendment before you read the program listing. If you decided you wanted the computer to give you two "NO" and two "YES" to every one "MAYBE", how would you amend it? Try to work this out (changes will be needed between lines 8Ø and 12Ø) before you read on.

If you need five alternatives (two NO, two YES, and one MAYBE), you will need line 8Ø to generate five random numbers. For two of these the computer must print NO, and so on. To save writing IF J = 1 THEN PRINT "YES", and IF J = 2 THEN PRINT "YES", and IF J = 3 THEN PRINT "NO" and so on, we can use the line IF J < 3 THEN PRINT "YES".

But what about the "NO" answer? If you write, for line 1ØØ, IF J < 5 THEN PRINT "NO", the computer will print a YES with a NO underneath if the number 3 or 4 is generated. You can overcome this with an AND statement, as follows:

```
8Ø  LET J = INT (5*RND)
9Ø  IF J < 2 THEN PRINT
    "YES"
1ØØ IF J > 1 AND J < 4
    THEN PRINT "NO"
11Ø IF J = 4 THEN PRINT
    "MAYBE"
```

A little later on we'll work out a more sophisticated DECISION MAKER program, but for now let's do something a little more interesting — play Russian roulette.



Russian Roulette

The principle of the game is simple. You have a pistol with six chambers, only one of which contains a bullet. You spin the chamber, pull the trigger, and...either bang, or click. Already you are thinking "Aha, the random number generator, one in six." The only major decision to make is how many shots you are going to have before ending the game. Input the following program into your ZX81, run it a few times, then come back to this book for a discussion on some of the features of it.

```
1Ø  PRINT,"RUSSIAN
    ROULETTE"
2Ø  PRINT,"(C) HARTNELL
    1980"
3Ø  PRINT
4Ø  LET J = Ø
5Ø  PRINT "PRESS
    NEWLINE TO FIRE"
6Ø  INPUT T$
7Ø  CLS
8Ø  LET J = J + 1
9Ø  LET G = RND
1ØØ IF G < 5/6 THEN PRINT
    "CLICK"
11Ø IF G> = 5/6 THEN
    GOTO 14Ø
12Ø IF J = 1Ø THEN GOTO
    16Ø
13Ø IF J <1Ø THEN GOTO
    5Ø
14Ø PRINT "BANG...";
15Ø GOTO 14Ø
16Ø PRINT "YOU HAVE
    SURVIVED"
```

Notice that in line 9Ø instead of using INT (6*RND)+1 we just used RND on its own. RND if you remember is a random number between Ø and 1, so, five times out of six it will be less than 5/6, and once out of every six times it should be greater than 5/6. This is how the decision inlines 1ØØ to 11Ø is made.

This program uses the GOTO (in lines 11Ø, 12Ø, 13Ø and 15Ø) command to either give you another 'shot', print BANG... to fill the screen, or to print YOU HAVE SURVIVED. Line 4Ø sets J = Ø at the beginning, and line 8Ø adds one to it each time you run through until (unless you've shot yourself) you've been through ten times, that is when J = 1Ø. Lines 12Ø and 13Ø check the value of J and either send you back to 5Ø or advance to 16Ø. Now, add the following line to the program and run it a few times:

```
95  PRINT,G,J
```

This added line displays the number that line 9Ø has generated and, to the right, the number of the run (i.e. J) that you are on. After you've run it a few times, erase line 95. Before you read on, try to amend the program (just by adding lines between the present ones) to allow the program to give you (a) the option not to play at all; and (b) the chance for subsequent games if you manage to survive.

```
3Ø  PRINT "DO YOU
    WANT TO PLAY?"
32  INPUT A$
34  IF A$ = "NO" THEN
    STOP
```

and further down:

```
162 PRINT "DO YOU
    WANT ANOTHER
    GO?"
164 INPUT B$
166 IF B$ = "YES" THEN
    GOTO 4Ø
```

You may well have modified the program differently, but as long as it works, it doesn't matter exactly how you do it.

Now, the following is a very important point for the development of games' programs. The best way I've found to work out games is to set up the 'core' of the game first (in the case of RUSSIAN ROULETTE, the core would be the first version of the game). Having set up this core, and made sure it works, I then proceed to elaborate the game to make it more fun to play, while making sure I don't overload the ZX81's tiny RAM.

You may have felt that the running of the game in its present form is a little unsatisfactory, and that when line 95 was included (PRINT,G,J) it was a bit more interesting. So add the following line:

```
95  PRINT "THAT WAS SHOT ";J
```

and run the program. That certainly makes it a bit more interesting. Before reading on, I want you to try and write a new version of line 95 which will tell how many shots you've left to

reach 1Ø, rather than just telling you the number of the current shot. One way of doing it:

```
95  PRINT 1Ø — J;"SHOTS TO GO"
```

Now, let us have a second look at line 34. Instead of just stopping when the player decides not to play, why not let the ZX80 respond more definitely, say by printing the word CHICKEN to fill the screen. Try and work out how to do this. For a start, you'll have to change the STOP statement in line 34 into a GOTO command.

One way of doing it:

```
34  IF A$ = "NO" THEN
    GOTO 18Ø
18Ø PRINT "CHICKEN...";
181 GOTO 18Ø
```

This version prints a most interesting pattern of the word CHICKEN. We'll use this pattern-effect to produce a new program shortly, but for now, add a few more lines to the program so that it uses the player's name. Once you've done that, and before you read my version below, try to write in a line so the ZX81 won't print ...SHOTS TO GO on the shot which fills the screen with BANG... This is my final version of the program:

```
1Ø  PRINT ,"RUSSIAN
    ROULETTE"
2Ø  PRINT ,"(C) HARTNELL
    1980"
22  PRINT
24  PRINT "WHAT IS
    YOUR NAME?"
26  INPUT A$
27  PRINT
28  PRINT
3Ø  PRINT "DO YOU
    WANT TO PLAY,
    ";A$;"?"
32  INPUT B$
33  CLS
34  IF B$ = "NO" THEN
    GOTO 18Ø
4Ø  LET J = Ø
5Ø  PRINT "PRESS
    NEWLINE TO FIRE"
6Ø  INPUT T$
7Ø  CLS
8Ø  LET J = J + 1
9Ø  LET G = RND
95  PRINT A$;", ";10 —
    J;"SHOTS TO GO"
96  PRINT
97  PRINT
1ØØ IF G < 5/6 THEN PRINT
    " Graphic A CLICK
    graphic A"
1Ø5 PRINT
11Ø IF G> = 5/6 THEN
    GOTO 139 (you'll see
    why 139, rather than
    14Ø, by reading 139)
12Ø IF J = 1Ø THEN GOTO
    16Ø
13Ø IF J <1Ø THEN GOTO
    5Ø
139 CLS (this clears the
    screen for-the big
    BANG)
14Ø PRINT "BANG...";
15Ø GOTO 14Ø
```

```
160 PRINT "YOU HAVE
    SURVIVED, ";A$
162 PRINT "DO YOU
    WANT ANOTHER
    GO?"
164 INPUT C$
165 CLS
166 IF C$ = "YES" THEN
    GOTO 40
```

```
180 PRINT "CHICKEN...";  (I
    removed the STOP
    statement here so the
    following line, 190,
    could serve for a NO
    response to 162, as well
    as for the NO response
    to 34.)
190 GOTO 180
```

Now, before we leave Russian roulette games, we'll look at one final version that introduces you to a new command, and shows a slightly different conversational approach. SAVE your final version of RUSSIAN ROULETTE on cassette, then clear the memory, and input the following program:

```
10  PRINT ,"RUSSIAN
    ROULETTE (3)"
15  PRINT ,"(C) HARTNELL
    1980"
20  PRINT
25  PRINT "HI GUN-
    TOTER. YOU HAVE"
30  PRINT "A PISTOL
    WITH ONE BULLET IN
    IT."
35  PRINT "YOU WILL
    SPIN THE CHAMBER"
40  PRINT "AND FIRE 10
    TIMES."
45  PRINT "ARE YOU
    GAME TO PLAY?"
50  INPUT A$
55  LET J = 0
60  PRINT
65  IF A$ = "NO" THEN
    GOTO 185
70  PRINT "PRESS
    NEWLINE TO FIRE"
75  INPUT T$
80  CLS
85  PRINT
90  PRINT
```

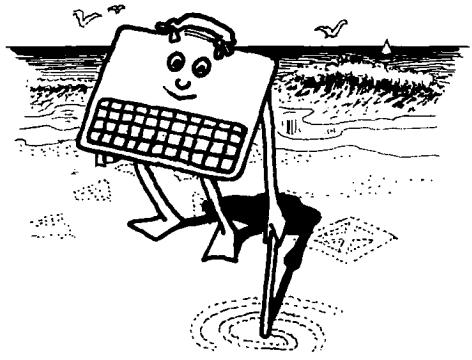
```
95  PRINT
100 LET J = J + 1
105 LET G = RND
110 PRINT ,"SHOT
    NUMBER ";J
115 IF G < 5/6 THEN
    GOSUB 195
120 IF G>= 5/6 THEN
    GOTO 220
125 IF J = 10 THEN GOTO
    140
130 IF J < 10 THEN GOTO
    70
135 CLS
140 PRINT
145 PRINT
150 PRINT ,"YOUVE
    SURVIVED."
155 PRINT "IF YOU WANT
    TO RISK DEATH
    AGAIN"
160 PRINT "PRESS G FOR
    GO...OR S FOR STOP"
165 INPUT B$
170 CLS
180 IF B$ = "G" THEN
    GOTO 55
```

```
185 PRINT "COWARD...";
190 GOTO 185
195 LET H = INT
    (2*RND)+1
200 IF H = 1 THEN PRINT
    "$ CLICK $" (use a
    'graphic T' in place of
    the dollar signs)
210 IF H = 2 THEN PRINT
    "EMPTY CHAMBER"
215 RETURN
220 PRINT "BANG...";
225 GOTO 220
```

You'll notice as you play this that an empty chamber (i.e. G 6) gives you either CLICK or EMPTY CHAMBER, which, as you can see, is generated by the lines 195 to 215. The ZX81 leaps to 195 from 115 on the command GOSUB 195.

Lines 195 through to 215 make up a "subroutine". Whenever the computer comes across a GOSUB command, it goes to the line specified, and follows on until it comes to command RETURN. The computer then returns to the line *after* the GOSUB command. The GOSUB/RETURN is a very useful feature. We'll be using it again.

You will recall that when you first worked out how to get the computer to fill the screen with the word CHICKEN that it made quite an attractive pattern. We are going to look now at a slightly more developed form of pattern-making program, which uses the FOR/NEXT loop to limit the number of times the sequence of letters or spaces is repeated. We will be looking at the FOR/NEXT in detail a little later.



Pattern Makers

The core of this program is simple. Feed it into your ZX81 and after pressing RUN, input any six of the graphical symbols, then press NEWLINE.

```
1Ø  INPUT A$
2Ø  FOR J = 1 TO 75
3Ø  PRINT A$;
4Ø  NEXT J
```

You can run this program, which is surprisingly effective, using any combination of letters, numbers and symbols you like. You'll find that one or two spaces, instead of letters, enhance the pattern produced. Try a few more patterns, using spaces, letters like M and W, the $ sign and the numbers 6 and 9.

Before reading on to see how I have done it, try to write around the core program you have to include the following features: (a) a title; (b) instructions; and (c) the chance to form another pattern without having to go back into the command mode and press RUN again. Cover up my version until you've tried your own.

One way (and there is an infinite number of ways you could have done it) is as follows:

```
1   PRINT ,"PATTERNS"
2   PRINT , "(C) HARTNELL
    1980"
3   PRINT
6   PRINT "PRESS ANY
    COMBINATION OF 6"
7   PRINT "LETTERS,
    SYMBOLS AND
    SPACES..."
8   PRINT
9   PRINT "...AND I WILL
    PRINT A PATTERN"
1Ø  INPUT A$
15  CLS
2Ø  FOR J = 1 TO 75
3Ø  PRINT A$;
4Ø  NEXT J
5Ø  PRINT "twenty four
    spaces"
6Ø  PRINT "DO YOU
    WANT ANOTHER
    GO?"
7Ø  INPUT B$
75  CLS
```

```
80  IF B$ = "YES" THEN
    GOTO 6
90  PRINT "OK. SEE
    YOU LATER, ARTIST"
```

This is quite an addictive program. The simplest combination of symbols (like three of graphic 6, two of graphic 5 and a space) produce almost three-dimensional patterns. If you set this program up for one of your friends to try, be ready to wait a long, long time before you next have a go.

Building a Library

I hope you've been SAVEing the programs you create as you go along. There is little point in having to input the whole program by hand every time you want to run it.

Resist the temptation to put all your programs on one cassette, one program after another. The frustration you'll experience searching through the tape (even if you've identified each program with a voice label) to find a particular program is just not worth the trouble. Go to your computer shop, or buy by mail from one of the many companies that advertise in the personal computing magazines, and get a set of C-12 cassettes. Put just one program on each cassette, with two copies of each program on each side, just in case something happens to one of the copies and you find it difficult, or impossible, to LOAD.

Write the name of the program on the cassette, and on the cardboard insert. You'll find this makes it very easy to find the program you want, and as your library builds, it will give you quite a feeling of accomplishment to see all those programs ranked side by side. It will impress your friends as well, who will believe after visiting you, and playing with your ZX81, that you're some kind of natural computer genius (which you probably are).

Review

Let's review the ground we've covered so far;

- The use of the random number generator LET J = RND or LET J = INT (10*RND)+1
- The output statement PRINT
- The assignment statement LET

- The use of , and ; in PRINT statements
- The PAUSE statement
- The control statement IF...THEN...
- The use of strings (A$), including the use of a string to stop a program until NEWLINE is pressed
- The statement CLS
- The symbols < >,<= and >=
- The statement AND
- The control statement STOP
- The control statement FOR...TO/NEXT
- The form and use of a "counter" to limit the number of times part of a program is run through
- How to develop a game program from the core of it, to make it more elaborate and interesting
- Subroutines (with the commands GOSUB and RETURN)
- How the string, FOR...TO.../NEXT, and PRINT can be used to generate patterns.

We've come quite a long way in a short time. Make sure you understand all the preceding material before going on.

For/Next

You'll remember when we wrote the first PATTERN-MAKER program, we used (in line 2Ø) FOR J = 1 TO 75, and (in line 4Ø) NEXT J. We will now have a look at the use of FOR/NEXT loops in some detail.

First, input the following program:

```
1Ø  LET J = Ø
2Ø  PRINT "J = ";J
3Ø  IF J = 1Ø THEN STOP
4Ø  LET J = J + 1
5Ø  GOTO 2Ø
```

Run it a few times, then clear the RAM and input the following program, which uses the FOR/NEXT loop:

```
1Ø  FOR J = 1 TO 1Ø
2Ø  PRINT "J = ";J
3Ø  NEXT J
```

You can see that running this program produces exactly the same result as running the preceding one, although this second program is only three lines long, two shorter than the first program. It is not always possible or desirable to use a FOR/NEXT loop as a "counter", but because it uses less memory

than the LET J = Ø/LET J = J + 1 approach, it should always be investigated.

When you run a program with a FOR/NEXT loop, the from line to line after the FOR statement until it finds the NEXT command, when it reverts to the FOR line. You can place one (or more) FOR/NEXT loops inside another. Add the following lines to the current program:

```
22  FOR K = 2 TO 6 STEP 2
25  PRINT "K =";K
27  NEXT K
```

Run this. Notice what happens to K - it runs 2,4,6. This is the effect of putting STEP 2 after FOR K=2 TO 6. K is incremented by 2 each time round the loop instead of just by one.It is important to make sure that each loop nests neatly within each other loop. That is, if you placed the NEXT K command **after** the NEXT J, so the loops cross, you get a rather unsatisfactory result. Try it, by renumbering line 27 as line 35 (making sure you delete the "old" line 27).

Finally, here is a remarkably effective double FOR/NEXT loop program:

```
5   PRINT "YOUR NAME
    IN LIGHTS"
1Ø  FOR J = 1 TO 6
2Ø  PRINT ,"(put your first
    name here)";
3Ø  FOR S = 1 TO 99
4Ø  PRINT "graphic T";
5Ø  NEXT S
6Ø  NEXT J
```

This is a good program to run for your more egocentric friends.

Flipping a Coin

The computer can be used to simulate a huge variety of things, from the growth of plants to the effects of bumps on the road on motorcycle suspension. We are going to look at a more homely example of simulation: flipping a coin.

Input the following program, and run it:

```
1Ø  PRINT ,"COIN FLIP —
    1"
2Ø  LET J = RND
3Ø  IF J > .5 THEN PRINT
    "HEADS"
4Ø  IF J< = .5 THEN
    PRINT "TAILS"
```

Once you've run this a few times, amend the program as follows:

```
1Ø  PRINT , "COIN FLIP —
    2"
2Ø  LET J = RND
3Ø  IF J> .5 THEN PRINT
    "HEADS"
4Ø  IF J<= .5 THEN PRINT
    "TAILS"
5Ø  PRINT "TO FLIP
    AGAIN PRESS F"
6Ø  PAUSE 4ØØØØ
7Ø  IF INKEY$ = "F" THEN
    GOTO 2Ø
```

You'll need to use the function key (shift newline) to get the word INKEY$ in line 7Ø. You might think that after using PAUSE 1ØØ and then played around with the number 1ØØ you'd have a pretty good idea of what PAUSE was all about. So what's this PAUSE 4ØØØØ? It seems an awfully long time to wait for. In actual fact PAUSE will always stop working as soon as you press a key - any key will do. When this happens the program will just carry on as normal, and this is where INKEY$ comes in. INKEY$ (or"In Key") will tell the computer which key you've got your finger on. If you're not pressing any key, or you're pressing two keys at once, INKEY$ will equal the empty string " ". If you're

pressing the A key then INKEY$ will equal "A". If you're pressing the F key then INKEY$ will equal "F". You can even make INKEY$ = "keyword STOP" by pressing SHIFT and A at the same time! IF INKEY$="F" means IF you're pressing key F. You could have used an INPUT statement instead - see if you can work out how to do this.

You can run this until the screen is full. Although there will probably be an imbalance in the numbers of HEADS and TAILS, the longer you run the program, the more the ratio of each should approach 1:1. If we knew how many times we had flipped the coin, it would make it easier to work out how many HEADS and how many TAILS the program had printed. It is fairly simple to amend the program to do this. Try it yourself, before looking at my version.

```
10  PRINT "COIN FLIP — 3"
20  LET K = 1
30  LET J = RND
40  PRINT "FLIP NUMBER ";K;" IS ";
50  IF J> .5 THEN PRINT "HEADS"
60  IF J< = .5 THEN PRINT "TAILS"
70  LET K = K + 1
80  PRINT "TO FLIP AGAIN PRESS F"
90  PAUSE 40000
100 IF INKEY$= "F" THEN GOTO 30
```

Once you've run this a few times, and added up the HEADS and TAILS, and then worked out what the ratio of HEADS to TAILS is, you will probably have realised the ZX81 can do all the work — flip the coin, count the heads and tails, and then manipulate this result as you choose. Let's try a more complex program:

```
10  PRINT ,"COIN FLIP — 4"
20  LET H = 0
30  LET T = 0
40  LET K = 1
50  LET J = RND
60  PRINT ,"FLIP NUMBER ";K;" IS ";
70  IF J> .5 THEN PRINT "HEADS"
80  IF J< = .5 THEN PRINT "TAILS"
90  IF J> .5 = 1 THEN LET H = H + 1
100 IF J<= .5 THEN LET T = T + 1
110 PRINT "OUT OF ";K;" FLIPS YOU HAVE"
120 PRINT H; "HEADS AND ";T;" TAILS"
130  PRINT "THIS IS A RATIO OF";H/T
140 LET K = K + 1
150 PRINT "PRESS F TO FLIP AGAIN"
160 PAUSE 40000
170 CLS
```

```
18Ø IF INKEY$= "F" THEN
     GOTO 5Ø
19Ø PRINT "THANKS FOR
     PLAYING. BYE"
2ØØ STOP
```

You will find that adding PRINT statements for lines 1Ø5, 1Ø6 and 1Ø7 will make the program, when running, easier to read. Now, we could use a FOR/NEXT loop to tell the ZX81 in advance how many times we wanted to flip the coin, and then we could leave it to do the work. For practice, I'd like you to produce such a program, which also asks the player at the beginning how many times he or she wishes to flip the coin. I am not going to give you a sample version of this because I think by now you should be able to write such a program easily. What I am going to give you though is another version of the program, that gives percentage breakdowns of heads and tails. First write your program using the FOR/NEXT loop, SAVE it, and then have a look at my final COIN FLIP program.

```
1Ø   PRINT ,"FLIP-A-COIN"
15   PRINT ,"(C) HARTNELL
     1980"
2Ø   LET H = Ø
3Ø   LET T = Ø
4Ø   LET K = 1
5Ø   LET J = RND
55   PRINT
57   PRINT
6Ø   PRINT "FLIP
     NUMBER ";K;" IS ";
7Ø   IF J>.5  THEN PRINT
     "HEADS"
8Ø   IF J<= .5 THEN PRINT
     "TAILS"
9Ø   IF J>.5 THEN LET H =
     H + 1
1ØØ  IF J<= .5 THEN LET T
     = T + 1
1Ø5  PRINT
1Ø6  PRINT
1Ø7  PRINT
11Ø  IF K = 1 THEN GOTO
     15Ø
111  LET A = 1ØØ*H/K
112  LET B = 1ØØ*T/K
12Ø  PRINT "OUT OF ";
     K;" FLIPS YOU HAVE"
13Ø  PRINT B;" PER CENT
     TAILS"
14Ø  PRINT "AND ";A;" PER
     CENT HEADS"
142  PRINT
147  PRINT
15Ø  PRINT ,"PRESS F TO
     FLIP AGAIN"
155  LET K = K + 1
16Ø  PAUSE 4ØØØØ
17Ø  CLS
175  IF INKEY$= "F" THEN
     GOTO 5Ø
18Ø  IF K = 2 THEN GOTO
     195
181  PRINT
182  PRINT
185  PRINT "YOU FLIPPED
     THE COIN ";K-1;
     " TIMES:-"
186  PRINT
187  PRINT,"HAD ";A;" PER
     CENT HEADS"
188  PRINT
```

```
189 PRINT ,"AND ";B;" PER
    CENT TAILS"
190 PRINT
191 PRINT
192 PRINT
193 PRINT
194 PRINT
195 PRINT "THANKS FOR
    PLAYING. BYE"
200 STOP
```

There are a few points you can learn from examining this program. Up to line 100, there is nothing new. Line 110 (IF K = 1 THEN GOTO 150) bypasses all the percentage computation, because there is no need for it if you have only flipped once. Lines 111 and 112 multiply the ratio of heads and tails by the total number of flips by 100.

Line 180 (IF K = 2 THEN GOTO 195) is used if the player decides to stop after two flips. The information given by lines 185 to 189 is only of interest if the coin has been tossed three or more times. Line 195 is just a friendly way of ending. Many times, when you write a program, you'll use nearly all the RAM just for the program, so you will not have room for niceties like goodbyes. However, whenever you have room in the memory, it is a good idea to make the program more "user-friendly" and make the ZX81 "conversation" as natural as possible.

Decision Maker — Mark 11

One of the first programs we developed was a simple decision maker. Let us look now at the more complex program to make decisions. This program will show how strings can be used by the ZX81 to form complete sentences, and will also demonstrate a memory-saving way of using the random number generator. Input the following program:

```
10 PRINT "DECISION
   MAKER — 2"
11 PRINT "(C) HARTNELL
   1980"
12 PRINT
15 PRINT "HI, WHAT IS
   YOUR NAME?"
20 INPUT A$
22 CLS
25 PRINT A$;", I AM
   HERE TO HELP YOU"
26 PRINT "REACH A
   DECISION"
27 PRINT
28 PRINT "JUST
   COMPLETE THE
   FOLLOWING"
29 PRINT "SENTENCE
   (WITHOUT USING
   THE"
```

```
3Ø   PRINT "WORDS I, ME
     OR MINE)"
31   PRINT "COMPLETE
     THIS: SHOULD I..."
32   INPUT B$
33   CLS
35   LET J = INT
     (4*RND)+1
36   GOTO 1ØØ*J (this is the
     memory-saving trick)
1ØØ  PRINT "IF I WERE
     YOU, ";A$; ", I WOULD
     NOT"B$
1Ø5  GOTO 6ØØ
2ØØ  PRINT "SURE, ";A$;
     ", ";B$
2Ø5  GOTO 6ØØ
3ØØ  PRINT "YOU MUST BE
     CRAZY, ";A$;", TO
     THINK YOU MIGHT
     ";B$
3Ø5  GOTO 6ØØ
4ØØ  PRINT "GEE, ";A$;",
     ITS A TOSSUP
     WHETHER YOU ";B$;"
     OR NOT"
4Ø5  GOTO 6ØØ
6ØØ  PRINT
61Ø  PRINT
62Ø  PRINT "ANOTHER
     DECISION, ";A$;"?"
63Ø  INPUT C$
64Ø  CLS
65Ø  IF C$ = "YES" THEN
     GOTO 27
66Ø  PRINT "THANKS FOR
     LETTING ME"
67Ø  PRINT "HELP YOU,
     ";A$
68Ø  STOP
```

Number Crunching

The programs we've worked on so far have ignored the great, and fundamental ability the ZX81 has to work with numbers.

Input this program into your computer:

```
1Ø   PRINT "MATHS
     DEMONSTRATION"
15   PRINT "GIVE ME A
     NUMBER"
```

```
20  INPUT A
25  PRINT "A SECOND
    NUMBER, PLEASE"
30  INPUT B
35  PRINT "AND ONE
    MORE"
40  INPUT C
45  LET K = A*B/C
50  PRINT A;" MULTIPLIED
    BY ";B;", DIVIDED BY"
55  PRINT C;" EQUALS ";K
```

Run this through a few times, with different values for A, B and C. Notice how, in lines 50 and 55, when using PRINT, the ZX81 substitutes the values assigned to A, B and C. It prints the number rather than the letter.

Now, change lines 45, 50 and 55 as follows:

```
45  LET K = (A +
    C)/(B+C-A)
50  PRINT K
55  STOP
```

Run this a few times and see what happens.

We can use a program of this type to do practically any maths problem, no matter how complex. The ZX81 can handle any numbers up to 1.7014118E38, and negative numbers down to −1.7014118E38. The E is a rather special symbol used by the ZX81 and it means "move the decimal point". E38 then means "with the decimal point moved 38 places to the right" so the biggest number the ZX81 can use in actually 17014118000000000000000000000000000000000, which is much larger than most of us will ever need. The ZX81 can just as easily handle functions like SINes, COSines and so on.

We'll use the basic mathematical ability of the ZX81 to produce a rather interesting demonstration program you can run next time you have a couple of friends around.

```
10  PRINT "NUMBER
    JUGGLE"
30  PRINT "NAME OF
    PLAYER?"
40  INPUT A$
65  CLS
70  PRINT A$;", THINK OF
    A NUMBER AND"
75  PRINT "DOUBLE IT"
95  PAUSE 40000
100 CLS
120 PRINT A$;", ADD
    FIVE"
125 PAUSE 40000
130 CLS
135 PRINT A$; " SUBTRACT
    SEVEN"
140 PRINT "AND TYPE IN
    THE RESULT"
145 INPUT A
150 CLS
175 LET K = (A + 2)/2
185 PRINT "YOUR
    NUMBER, ";
    A$,", WAS ";K
```

Now that did not, perhaps, seem very startling, although it is fairly impressive the first time you try it on someone. Note how the PAUSE 40000 was used twice to stop the program until you pressed any key, how CLS was used to clear the screen after each instruction, and how the final line (185) used the name string (A$) and the assigned variable (K) to print out your answer.

Before you read on, work out a way of letting the program give you the chance to have another go. Type your version in at the end of the program, and see if it works. One way would be as follows:

```
200 PRINT "ANOTHER
    GO?"
205 INPUT Z$
215 IF Z$ = "YES" THEN
    GOTO 65
230 STOP
```

Once you've run through this program a few more times, you're probably pretty bored with it. Remember that a computer can, within the limits of its memory, store a great deal more information than just one simple mathematical manipulation. This game program becomes a great deal more interesting when you interweave a second set of instructions into the first, so a second player can "think of a number" and so on, at the same time as the first person is doing so. I'll list the whole program. Apart from a word to be added to line 30, and some possible differences in the lines 200 and beyond (the lines that offered you a second run), you should be able to add the lines from the following program in between those you already have in the ZX81's memory.

```
10  PRINT "NUMBER
    JUGGLE"
20  PRINT ,"(C) HARTNELL
    1980"
25  PRINT
30  PRINT "NAME OF
    FIRST PLAYER?"
40  INPUT A$
45  PRINT
50  PRINT "NAME OF
    SECOND PLAYER?"
60  INPUT B$
65  CLS
70  PRINT A$;", THINK OF
    A NUMBER AND"
75  PRINT "DOUBLE IT"
80  PRINT
85  PRINT B$;", THINK OF
    A NUMBER AS WELL"
90  PRINT "AND ADD SIX
    TO IT"
95  PAUSE 40000
100 CLS
105 PRINT B$;", I WANT
    YOU TO MULTIPLY
    YOUR"
110 PRINT "NUMBER BY
    THREE"
115 PRINT
120 PRINT A$;", ADD
    FIVE"
```

```
125 PAUSE 40000
130 CLS
135 PRINT A$;", SUBTRACT
    SEVEN"
140 PRINT "AND TYPE IN
    THE RESULT"
145 INPUT A
150 CLS
155 PRINT B$;", I WANT
    YOU TO SUBTRACT"
160 PRINT "FOUR THEN
    TYPE IN YOUR
    RESULT"
165 INPUT B
170 CLS
175 LET K = (A + 2)/2
180 LET L = ((B + 4)/3) -
    6
185 PRINT "YOUR
    NUMBER, ";A$;
    ", WAS ";K
190 PRINT
195 PRINT "AND
    YOURS, ";
    B$;", WAS ";L
200 PRINT ,"ANOTHER
    GO?"
205 INPUT Z$
210 CLS
215 IF Z$ = "YES" THEN
    GOTO 65
220 PRINT "OK, ";A$;"
    AND "; B$;", THANKS"
225 PRINT "FOR PLAYING.
    BYE BYE."
230 STOP
```

Try this out with a couple of friends, and watch their reaction.

I want to introduce you now to a little trick which can add a lot to games programs (even though it will probably horrify computer purists). The ZX81 works very, very quickly. So quickly, in fact, that its responses appear to be instantaneous. Although this is fine for "serious" use of a computer, it detracts in some measure from games. You'll find that games are more interesting if the computer appears to be "thinking" between moves, rather than responding as soon as you hit NEWLINE. So, we can introduce a PAUSE subroutine which will slow things down a bit.

Add the following:

```
240 PAUSE 150
250 CLS
260 RETURN
```

Now, go through the program, and change all the lines except 250 which have the instruction CLS to GOSUB 240. Now, run the program again. Notice how much more effective this seems, with just the right delay for the computer to "think and reach a decision". You could have other numbers in line 240. Try it as PAUSE 400. You'll find this delay is far too long. So long that you'll get irritated. Using 200 also produces too long a delay. To me, 150 seems about right. You can use the subroutine delay in any games you like. As an exercise, try adding it to an earlier program, like RUSSIAN ROULETTE, and seeing if this adds to the playing of the game.

The general form of a "game delay":

Line Number PAUSE n

Line Number CLS

Line Number RETURN (if the delay is a subroutine).

As a general rule, you can insert a delay subroutine whenever you would normally use the CLS instruction. When you're feeding in a program that is likely to come pretty close to using all the ZX81's RAM, do not input the delay subroutine until you've got the rest of the program in. Use a CLS line instead, and if and when you find there is enough memory left for a delay subroutine, put that in.

Writing a Program — Some General Thoughts

Most books on programming suggest you start by drawing up a "flowchart" — a pretty combination of circles, diamonds and slanting rectangles — which sets out the path and operations the computer will follow to execute a program. In theory, this is fine. But in practice, especially when working with a relatively small memory such as in the ZX81, the time and trouble involved is probably not worth it.

It is, however, essential to know exactly what you want the computer to do before you start creating a new program, even if you're not quite sure how you are going to get the ZX81 to carry out the task.

Sometimes a rough sort of flowchart — just writing down the main steps the ZX81 will take and then linking these by lines and loops — will help to clarify your thinking. This is also a good way of spotting potential problems, like the danger of setting up infinite loops, or of not specifying the nature of the computer's decisions exactly.

For what it's worth, I work as follows:

Having worked out the general idea for a game (like "player thinks of an animal, computer tries to guess which animal") I then just think about the idea for a while. I might not write anything down for even a day or more. Usually, this "thinking

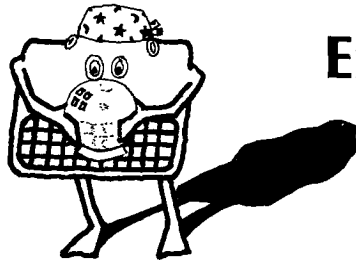
time" allows me to work out roughly how the core of the program is going to look. Next, on paper, I write down this core, starting with line number 100. This ensures there is plenty of room at the start of the program to add a title and instructions, define constants, set up arrays and the like.

The next stage is to feed the rough program into the ZX81 and see if it does what it is meant to do. Often a far more elegant method than the first rough version is found at the keyboard, but this would probably not have been discovered if the written version was not tried first.

Occasionally, if I get a clear idea for a program and I cannot get to the ZX81, I will write out the entire program by hand in advance. Sometimes (like the race game LONDON TO GLASGOW), a program works almost perfectly the first time. Other programs turned out to be near disasters, so the final version bears little more resemblance than title and intention to the original, hand-written version. If you're working on a fairly simple game, or are adapting from a magazine or book, it is just as well to work directly on the ZX81, but keep a notebook handy to record things like the fact that, for example, string A$ is for the player's name.

If you're writing a program direct on the keyboard, and you want to add a subroutine which will eventually be at the end of the rest of the program, give it a very big number (like 5000). It can easily be renumbered afterwards (and the GOSUB command changed). This is simpler, and more elegant, as well as using less memory, than having to use a GOTO to jump over a subroutine which has been given too low a line number.

If there is a phrase that you'll need the computer to print at several parts of the program, such as "THE SCORE AT THIS POINT IS", assign a string to this phrase as soon as you realise you're going to have to place the line in several different places in the program. A lot less memory is involved in the line PRINT B$ appearing six times, than in six of PRINT "THE SCORE AT THIS POINT IS". If the information to be included several times is longer than one line, a subroutine is probably the most memory-efficient device to use. Later on this book looks at a program which makes use of a large number of strings for phrases which are used over and over again (LONDON TO GLASGOW) and when you come to the game, you'll see how efficient a process this can be.



Extra-Sensory Perception

The next program we will look at uses the ZX81's "greater than" and "less than" facilities to compare two numbers, and then make a decision on the basis of whether one number is less than, or greater than, the other. This program also makes use of many of the things we've covered so far, including the IF...THEN, the counter and the random number generator.

Input the following program:

```
1Ø  PRINT "GUESS MY
    NUMBER"
45  PRINT "WHAT
    NUMBER, BETWEEN Ø
    AND 99, AM I
    THINKING OF?"
5Ø  LET J = INT (1ØØ*RND)
55  LET S = Ø
6Ø  LET S = S + 1
65  INPUT A
7Ø  PRINT, A
75  IF A = J THEN GOTO
    1Ø5
8Ø  IF A < J THEN PRINT
    "HIGHER"
85  IF A > J THEN PRINT
    "LOWER"
9Ø  IF S < 5 THEN GOTO
    6Ø
95  IF S = 5 THEN GOTO
    13Ø
1Ø5 PRINT "YOU ARE
    RIGHT. I WAS
    THINKING OF ";J
11Ø GOTO 14Ø
13Ø PRINT "TIME IS UP. I
    WAS THINKING OF ";J
14Ø PRINT "DO YOU
    WANT ANOTHER
    GO?"
145 INPUT H$
15Ø CLS
155 IF H$ = "YES" THEN
    GOTO 45
16Ø STOP
```

Notice how the core of the program (lines 5Ø to 95) make the decision on what the ZX81 is to do next.

This program, in its present form, has only used up a part of the ZX81's memory. It is possible to dress the program up a bit, by giving the computer your name, and — as you'll see — giving it the ability to award the player a random payoff if the number

is guessed correctly. Try this new form. Some of the original program remains, but certain lines have to be replaced, and others added.

```
1Ø  PRINT "GUESS MY
    NUMBER"
15  PRINT "(C) HARTNELL
    1980"
2Ø  PRINT
25  PRINT
3Ø  PRINT "HI, WHAT IS
    YOUR NAME?"
35  INPUT A$
4Ø  CLS
45  PRINT "WHAT
    NUMBER, BETWEEN Ø
    AND 99, AM I
    THINKING OF, ";A$;"?"
5Ø  LET J = INT(100*RND)
55  LET S = Ø
6Ø  LET S = S + 1
65  INPUT A
7Ø  PRINT ,A
75  IF A = J THEN GOTO
    1ØØ (note change from
    1Ø5)
8Ø  IF A < J THEN PRINT
    "HIGHER"
85  IF A > J THEN PRINT
    "LOWER"
9Ø  IF S < 5 THEN GOTO
    6Ø
95  IF S = 5 THEN GOTO
    13Ø
1ØØ LET K = INT (3*RND)
1Ø5 PRINT "YES, I WAS
    THINKING OF ";J;" I
    HEREBY AWARD YOU
    THE TITLE OF";
11Ø IF K = Ø THEN PRINT
    " SMART ALEC OF THE
    YEAR, ";A$
115 IF K = 1 THEN PRINT
    " MASTER OF ESP,
    ";A$
12Ø IF K = 2 THEN PRINT
    " JERK OF THE
    KEYBOARD, ";A$
125 GOTO 14Ø
13Ø PRINT "YOU ARE
    STUPID, ";A$;". I WAS"
135 PRINT ,"THINKING OF
    ";J
14Ø PRINT
145 PRINT "DO YOU
    WANT ANOTHER
    GO?"
15Ø INPUT N$
155 CLS
16Ø IF N$ = "YES" THEN
    GOTO 45
165 PRINT "THANKS FOR
    PLAYING, ";A$
17Ø STOP
```

When you run this program, you'll find the lines 1Ø5 to 12Ø wrap around to the following line when printing. This is not a good feature. As an exercise, try and rewrite this section so the lines print separately, so you do not run the risk of having a word cut in half, with one half on one line, and the next on the other. (Hint: you'll need to use commas in your PRINT line).

Kill the Chopper

It is possible to use a very similar core to produce what appears to be a completely different program. Input the following:

```
1Ø  PRINT "CHOPPER —
    (C) HARTNELL 1980"
2Ø  PRINT
3Ø  PRINT
4Ø  PRINT "A DEADLY
    CHOPPER IS HIDING"
45  PRINT "BEHIND 1 OF
    16 TREES"
5Ø  PRINT
55  PRINT
6Ø  PRINT "YOU HAVE
    ONLY 6 SHOTS. OK?"
65  INPUT A$
7Ø  CLS
75  LET G = Ø
8Ø  LET J = INT
    (16*RND)+1
85  PRINT, " Ø Ø"
9Ø  PRINT, "two inverse
    space, graphic 7, two
    inverse space"
95  PRINT,"graphic 5,
    space, graphic 7, space,
    graphic 5"
1ØØ PRINT, "graphic 5,
    three spaces, graphic 5"
1Ø5 PRINT, "graphic 5 five
    times)"
11Ø PRINT, "(space, inverse
    space, space, inverse
    space"
115 PRINT
12Ø PRINT
125 PRINT "GUESS THE
    NUMBER OF THE
    TREE"
13Ø PRINT
135 PRINT ,"YOU WANT
    TO SHOOT AT"
14Ø LET G = G + 1
145 INPUT M
15Ø CLS
155 IF M = J THEN GOTO
    22Ø
16Ø IF G = 6 THEN GOTO
    195
165 PRINT "YOU MISSED,
    ";6 — G;" TO GO"
17Ø PRINT
175 PRINT
18Ø IF M < J THEN PRINT
    "TRY TO THE RIGHT"
```

```
185 IF M > J THEN PRINT
    "TRY TO THE LEFT"
190 IF G < 6 THEN GOTO
    85
195 PRINT
    "AAAAAAAARGHH....."
200 PRINT
205 PRINT
210 PRINT "HE WAS
    BEHIND TREE
    NUMBER ";J
215 STOP
220 PRINT "GOT HIM"
225 PRINT " Ø Ø(after the
    two zeros, space, two
    inverse space, space,
    graphic E, space,
    graphic 5)"
230 PRINT"(three spaces,
    two inverse space,
    seven spaces, graphic 4,
    space, graphic 4)"
235 PRINT
240 PRINT
245 PRINT
250 PRINT "PRESS G FOR
    ANOTHER GO"
260 PAUSE 40000
265 IF INKEY$ = "G"
    THEN GOTO 70
270 CLS
275 PRINT "CHICKEN"
```

The idiotic appearance of the "chopper" was chosen at random. (It was named after my dog.) Make up your own name and appearance for your beastie (making sure its "dead" version looks vaguely like a deflated version of the live creature). You can also alter the value of J (the number of trees) and/or G (the number of shots you get before being killed).

This framework can also be used for, say, vultures hiding in nests, poisonous snakes in holes, or even unfriendly aliens in space, lurking behind asteroids. The "conversation" parts of the program can also be altered to what ever form you choose.

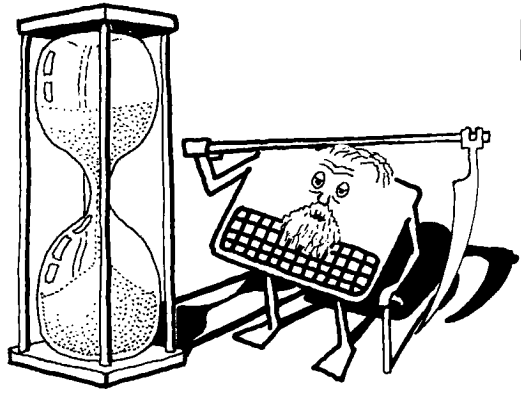
We can in fact shorten this program considerably by using the AT function. Delete all of the PRINT statements from the program and instead input the following lines.

```
10  PRINT "CHOPPER — (C) HARTNELL 1980";
    AT 3,0;"A DEADLY CHOPPER IS HIDING",
    "BEHIND 1 OF 16 TREES";
    AT 7,0; "YOU HAVE ONLY 6 SHOTS. OK?"
85  PRINT ,"00",,"two inverse space graphic S two
    inverse space",,"graphic 5 space graphic 7
    space graphic 5",,"five graphic 5",,"space
    inverse space space inverse space"
165 PRINT "YOU MISSED.";6-G;"TO GO";AT 2,0
180 IF M<J THEN PRINT "TRY TO THE RIGHT"
185 IF M>J THEN PRINT "TRY TO THE LEFT"
195 PRINT "AAAAAAAAARGHH.....";AT 5,0;"HE
    WAS HIDING BEHIND TREE NUMBER";J
```

```
220 PRINT "GOT HIM";AT 12,17;"ØØ space,two
    inverse space, space, graphic 5 , space,
    graphic 5",,"three space, two inverse space,
    seven space, graphic 4, space, graphic 4";AT
    17,Ø;"PRESS G FOR ANOTHER GO"
275 PRINT ,"CHICKEN"
```

You'll find the program runs exactly as before, except that it's considerably shorter. AT X,Y in the middle of a PRINT statement will move the PRINT position to line X, column Y. If the AT function is followed by a semi-colon it will stay there and wait for the next item to PRINT. See how in line 165 the AT function is at the end of the statement so that the next PRINT will start there. Note also the double commas in lines 85 and 22Ø. This lines the PRINT position up to the middle of the screen.

A little gimmick you can add to this (and any other program you like, assuming you have enough memory for it) is to get the program to ask the player's name. You work out a subroutine which ensures the computer will only play with you (or with people fortunate enough the share your name). If any other name is fed in when the ZX81 asks who is playing, it STOPS. As an exercise, work out such a subroutine and add it to your version of CHOPPER.



Days of Our Lives

Let's go back now to number-crunching, and have a look at a program which gives interesting (and sometimes scary or absurd) information on how long the player has to live. A little later on

in this book is a program designed to give a proper look at life expectancy, but that program is a little more complicated. For now, this program — DAYS OF OUR LIVES — produces a pretty good demonstration of your ZX81, and the less mathematical of your friends will be convinced, yet again, that you are some kind of genius.

What the program does, in essence, is work out approximately how old the player is in days, and compares this with average life expectancy (67½ years for males, 74½ for females). It also assumes the player sleeps eight hours a night. Input the following program and run through it a few times. Then read the discussion that follows the program listing.

```
10  PRINT "DAYS OF OUR
    LIVES"
15  PRINT "(C) HARTNELL
    1980"
20  PRINT
25  PRINT "HI, WHAT IS
    YOUR NAME?"
30  INPUT A$
35  CLS
40  PRINT "OK, ";A$;",
    LETS WORK OUT
    SOME"
45  PRINT "INTERESTING
    THINGS ABOUT YOU"
50  PAUSE 150
55  CLS
60  PRINT "HOW OLD
    ARE YOU IN YEARS?"
65  INPUT A
70  PRINT "AND
    MONTHS?"
75  INPUT B
80  LET X = 30.002*
    (A*12+B)
85  CLS
90  PRINT "YOU ARE
    ";INT X;" DAYS OLD,
    "A$
95  INPUT C$
100 PRINT "ARE YOU
    FEMALE?"
105 INPUT D$
110 CLS
115 IF D$ = "YES" THEN
    GOTO 130
120 LET Y = 24637 - X
125 GOTO 135
130 LET Y = 27192 - X
135 PRINT A$;", YOU
    HAVE ";INT Y;" DAYS
    TO LIVE - BASED ON
    STATISTICS"
140 PAUSE 150
145 CLS
150 PRINT "YOU HAVE
    SLEPT FOR
    ABOUT";INT (X/3)
155 PRINT "DAYS SO
    FAR"
160 PAUSE 150
165 CLS
170 PRINT "AND WILL
    SLEEP FOR
    ABOUT ";INT (Y/21)
175 PRINT "WEEKS OF
    YOUR REMAINING
    LIFE"
180 PAUSE 150
185 CLS
190 PRINT "IS THERE
    ANYONE ELSE THERE
    WHO"
```

```
195 PRINT "WOULD LIKE
    A GO, ";A$;"?"
200 INPUT H$
210 IF H$ = "YES" THEN
    GOTO 25
215 PRINT "OK BYE, ";A$
```

If you were thinking about what you were feeding into the ZX81, you probably have a pretty good idea of what each line and command was for, but, if not, look at it on your screen and follow through this next section, line by line with your program.

First of all, of course, the program got your name (A$), then PAUSED at line 50 for a few seconds (and clear the screen at line 55). This stopping of the program, as you learned earlier, does not do anything, but makes the program much more interesting to run than supplying all the information at once.

The program obtained your age in years (A) and months (B), then worked out what this was in days (line 80, which assumed there were 30 days in a month). Next the ZX81, in line 90, told you what this figure was (X). The question in line 100 — ARE YOU FEMALE? — is needed because males and females have different life expectancies. If D$ (the answer to the line 100 question) is YES, the computer goes to line 130, where it subtracts your age in days from the life expectancy (74½ years, converted to days) for females born after 1961 (it is only half a year less for females born 1951 — 1960). If the answer is not YES, the computer moves to the next line (120) and subtracts your age in days from 24,637 days, the life expectancy for males born after 1960 (again, it is half a year less for males born 1951 — 1960).

Line 135 prints how many days after are left. Lines 150 and 155 give the number of days slept so far (your age in days, divided by three), assuming you sleep eight hours a night. Then the ZX81 takes your remaining expected days (Y), divides this figure by three (to get the days spent in sleep) and then divides this by seven to convert this figure to weeks. Actually, as you can see, the program simply divides by 21 (i.e. 3 x 7).

Lines 190 to 210 ask if someone else wants a go, and if so, goes right back to line 25, where the string A$ is ready to be assigned to the new name. If there is no-one else to play the ZX81 proceeds to the end of the program.

If you feel like experimenting, there is no reason to stick with the program as I've written it. Let it ask, for example, if the player is male (changes lines 100 to 130). Instead of the somewhat unimaginative farewell (line 215) you might wish to put in some mournful line like: WELL, A$, I GUESS YOU'D

BETTER RUN OFF AND MAKE THE MOST OF THE Y/30 MONTHS YOU HAVE LEFT TO LIVE.

You could also, of course, add a delay subroutine where CLS appears in the program (lines 35, 55, 85, 11Ø, 145, 165, 185 and 2Ø5). By all means experiment with the program, and put your own personal stamp on it. By doing this, you'll gain far more knowledge about programming than you will just by feeding in the programs in this book, line for line, and then leaving them this way.

If you'd like to work out a whole new program based on the core of this program, feed the following program into your ZX81, and then work around it as you choose.

```
1Ø  PRINT "HOW OLD
    ARE YOU IN YEARS?"
2Ø  INPUT A
3Ø  PRINT , "AND
    MONTHS?"
4Ø  INPUT B
5Ø  LET X = 365.25*(A +
    12*B)
6Ø  PRINT "ARE YOU
    FEMALE?"
7Ø  INPUT A$
8Ø  CLS
9Ø  IF A$ = "YES" THEN
    GOTO 12Ø
1ØØ LET Y = 24637 - X
11Ø GOTO 13Ø
12Ø LET Y = 27192 - X
13Ø PRINT "YOU ARE";INT
    X;"DAYS OLD AND"
14Ø PRINT "HAVE ABOUT
    ";INT Y;" DAYS TO
    LIVE,"
15Ø PRINT "BASED ON
    STATISTICS"
16Ø PRINT "YOU HAVE
    SLEPT FOR ABOUT
    ";INT (X/3)
17Ø PRINT "DAYS SO FAR,
    AND WILL SLEEP"
18Ø PRINT "FOR ABOUT
    ";INT (Y/2);" WEEKS
    OF"
19Ø PRINT "YOUR
    REMAINING LIFE"
2ØØ STOP
```

Toying with the Muse

To make a change from working with numbers, here is a program that — after a fashion — writes poetry. Most of the poems it comes up with are pretty awful, but from time to time you'll get a real gem. And even the worst efforts of the ZX81 are not as bad as some of the worst modern poetry published today.

The heart of this program is our old friend the random number generator. It also makes use of the memory-saving device of linking the number generated to a subroutine. Input the program as it appears below, and run it *several* times. Then read the notes that follow the listing. (Note that there is a space just inside the quote marks in the "seed-phrase" lines following the command PRINT.)

```
1    PRINT , "SEASIDE
     POETRY"
2    PRINT , "(C) HARTNELL
     1980"
3    PRINT
4    PRINT
5    PRINT "PRESS
     NEWLINE TO WRITE A
     POEM"
6    INPUT A$
7    CLS
10   LET J = INT (21*RND)
15   IF J = 0 THEN PRINT
     "...";
20   IF J >= 1 AND J <=
     3 THEN GOSUB 1000
30   IF J >= 4 THEN
     GOSUB 10 + J*10
40   GOTO 10
50   PRINT " SUN";
55   RETURN
60   PRINT " SAND";
65   RETURN
70   PRINT " SEAGULLS";
75   RETURN
80   PRINT " WAVES";
85   RETURN
90   PRINT " ROCKS";
95   RETURN
100  PRINT " SEAWEED";
105  RETURN
110  PRINT " HOT";
115  RETURN
120  PRINT " GRITTY";
125  RETURN
130  PRINT "BEATING,
     BEATING";
135  RETURN
140  PRINT " HARSH";
145  RETURN
```

```
150 PRINT " HOURS
    PASSING";
155 RETURN
160 PRINT " ECHOED
    OVER";
165 RETURN
170 PRINT " BRIGHTLY
    DREW";
175 RETURN
180 PRINT " DREAMILY
    EASED";
185 RETURN
190 PRINT " SHADOWED
    OVER";
195 RETURN
200 PRINT " FAINTLY
    WHISPERED";
205 RETURN
210 PRINT " YET AGAIN";
215 RETURN
1000 PRINT
1001 PRINT
1002 PRINT
1003 RETURN
```

When you run this, you'll be quite pleased (or at least I was when I first wrote it) to find out how often, just by chance, quite attractive poems (!) are written by the program. Note the high number of RETURN commands. You'll remember that after going to a subroutine (GOSUB), the program executes the subroutine, then at the command RETURN, goes back to the line **after** the instruction to go to the subroutine. In this program, the line after the GOSUB instruction is 40, which instructs the computer to go further back, to line 10, to generate another random number. The decisions on where to place the semicolons (;), which tell the computer to print the next words on the same line, were made partly by running the program, and just seeing how the words fell, and also by the general decision that half the nouns (such as SUN, SAND and SEAGULLS) would end a sentence, that is, would not be followed by a semicolon. I also decided that adjectives (as in lines 120 and 140) would always allow for a word to follow them, and that about half the other "seeds" (lines 150 to 210) would do the same.

By looking at the program listing you'll see the lines 15, 20 and 30 make certain decisions, based on the random number generated. Line 15 ensures that, just under 5% of the time, "..." will be printed. Line 20 prints three blank lines (GOSUB 1000) just under 15% of the time, and line 30 directs the hardworking ZX81 to print one of the "seeds" just over 80% of the time. These proportions were worked out by running the program over and over again, adjusting the lines 15, 20 and 30 until — the majority of the time — a pleasant poem layout resulted. Probably you've realised that the program runs until the screen is full (which gives a fairly good poem length). If you wanted to limit the number of lines in the poem to less than full screen length, you could place a counter (or a FOR...NEXT) at the return target for

the GOSUB at line 3Ø (i.e. at line 4Ø). The kind of poem the program writes depends, as you can see, almost entirely on the words placed in the PRINT lines. The best way to decide on the words to go in the PRINT statements is to pick one topic, and then make every word and phrase relate to this topic.

The original POETRY program was written when I was on holiday, and after producing about 20 SEASIDE POEMS, I decide to input some words and phrases about the city where the program was written. The changes in the program produced some remarkably effective poems (plus a generous crop of duds). If you want to see how it works, change the following lines using the EDIT facility, making sure you put a space just inside the quote marks.

```
1    PRINT , " SALZBURG
     POETRY"
5Ø   PRINT " CHURCHES";
6Ø   PRINT " BAROQUE
     TOWERS";
7Ø   PRINT " MOUNTAIN
     VISTAS";
8Ø   PRINT " MUSIC BY
     MOZART";
9Ø   PRINT " FORTRESS";
1ØØ  PRINT " COBBLED
     STREETS";
11Ø  PRINT " TIMELESS";
12Ø  PRINT " BELOVED";
13Ø  PRINT " MEMORIES IN
     STONE";
14Ø  PRINT " STEADFAST";
15Ø  PRINT " TRADITIONS
     RELIVED";
16Ø  PRINT " ECHOED";
17Ø  PRINT " COPPER
     DOMES";
18Ø  PRINT " DREAMING"
19Ø  PRINT " SHADOWED
     OVER";
2ØØ  PRINT " FAINTLY
     WHISPERED"
21Ø  PRINT " YET AGAIN"
```

Try writing a program, using words and phrases based on the last place you spent a holiday in, or pick a topic like "clouds", "kittens" or "vintage cars"...and see what you, the ZX81 and the Muse can create. Here is one poem written from a set of words and phrases about London.

...TOURISTS CROWD ALWAYS MOVING

RUSHING, PUSHING I HAVE SEEN IT
BIG BEN CHIMES, AND RIVER THAMES
  CATHEDRAL SPIRES
ALL BUT FORGOTTEN RUSHING, PUSHING
PIGEONS IN TRAFALGAR SQUARE

I HAVE SEEN IT
AT LAST, SUN. RIVER THAMES

TIMELESS
I HAVE SEEN IT

Not very brilliant, I guess, but acceptable. The "seeds" for this poem are:

```
1    PRINT ,"POEMS OF
     LONDON TOWN"
5Ø   PRINT " BOBBIES,";
6Ø   PRINT " TOURISTS
     CROWD";
7Ø   PRINT " PIGEONS IN
     TRAFALGAR SQUARE"
8Ø   PRINT " CHAOS"
9Ø   PRINT " AT LAST,
     SUN.";
1ØØ  PRINT " STREETS OF
     ...";
11Ø  PRINT " TIMELESS"
12Ø  PRINT " ALL BUT
     FORGOTTEN";
13Ø  PRINT " MEMORIES
     OF MAJESTY,"
14Ø  PRINT " RIVER
     THAMES";
15Ø  PRINT " BIG BEN
     CHIMES, AND";
16Ø  PRINT " RUSHING,
     PUSHING";
17Ø  PRINT " ALWAYS
     MOVING,"
18Ø  PRINT " CATHEDRAL
     SPIRES"
19Ø  PRINT " GREY RAIN
     FALLS ON...";
2ØØ  PRINT " I HAVE SEEN
     IT"
21Ø  PRINT " MANY
     TIMES";
```

Notice how, in this program, link words like AND and ON finish some lines.

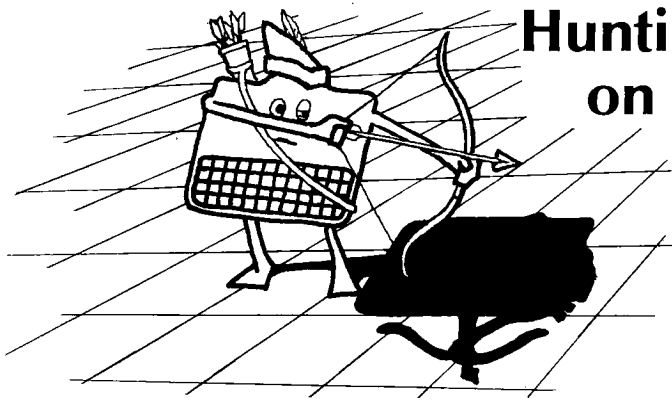
Here is another poem, from yet another set of words:

```
GRAVEYARDS ABOMINATIONS RELIVED
DARK, DARK SHRINKING WITCHES CACKLE
ECHOED CHILL OF ODDNESS CHANCES LOST

SHRINKING CALLING, CALLING, GRAVEYARDS
ECHOED CALLING, CALLING
DEATH IS NEAR, SAY WITCHES CACKLE
TURN AND REACH FOR

DARK, DARK WITCHES CACKLE
SKELETONS RATTLE
GRAVEYARDS
```

The "seeds" are as follows:

```
1    PRINT ,"BLACK
     SABBATH POETRY"
5Ø   PRINT "
     GRAVEYARDS";
6Ø   PRINT " SKELETONS
     RATTLE"
7Ø   PRINT " CHILL OF
     ODDNESS";
8Ø   PRINT " WITCHES
     CACKLE";
9Ø   PRINT " HOPE NOW
     DEAD";
1ØØ  PRINT " CHANCES
     LOST";
11Ø  PRINT " TIMELESS"
12Ø  PRINT " DARK,
     DARK";
```

```
130 PRINT " MEMORIES
    OF PAIN,";
140 PRINT " SHRINKING";
150 PRINT "
    ABOMINATIONS
    RELIVED";
160 PRINT " ECHOED";
170 PRINT " CALLING,
    CALLING,";
180 PRINT " DEATH IS
    NEAR, SAY";
190 PRINT " SCREAMS
    ARE";
200 PRINT " TURN AND
    REACH FOR";
210 PRINT "AGAIN,
    AGAIN"
```



Hunting on a Grid

There are many games, with names like HUNT THE HURKLE or BURIED TREASURE or DEPTH-CHARGE, which are based on trying to guess the location of one or more points on a grid. Each location is specified by quoting the co-ordinates of the point. To start our investigation of these games, input the following program, which sets up a very simple game of this type (you can leave out lines 10, 60, 70, 80 and 90 if you want to save time).

```
10 PRINT "HUNT GAME"
20 LET X =
   INT(10*RND)+1
30 LET Y =
   INT(10*RND)+1
40 PRINT "A ZX81 IS
   HIDDEN ON A 10 x 10
   GRID"
50 PRINT " YOU HAVE 10
   GUESSES TO FIND IT"
60 PRINT "INPUT YOUR
   GUESS, PRESSING
   NEWLINE"
70 PRINT "AFTER EACH
   DIGIT. TWO DIGITS
   ARE"
```

```
8Ø  PRINT "NEEDED, THE
    FIRST ONE YOU
    INPUT"
9Ø  PRINT "MUST BE FOR
    THE HORIZONTAL CO-
    ORDINATE"
1ØØ FOR J = 1 TO 1Ø
11Ø PRINT "FIRST
    NUMBER?"
12Ø INPUT A
13Ø PRINT A;
14Ø PRINT " SECOND
    NUMBER?"
145 INPUT B
15Ø PRINT B
16Ø IF A = X AND B = Y
    THEN GOTO 24Ø
17Ø IF A <>X OR B <>Y
    THEN PRINT "TRY
    AGAIN"
18Ø PAUSE 4ØØØØ
19Ø CLS
2ØØ NEXT J
21Ø PRINT "END OF
    GAME"
22Ø PRINT "ZX81 WAS
    HIDDEN AT ";X,Y
23Ø STOP
24Ø PRINT "YOU HAVE
    FOUND IT"
```

Once you've played this a few times, you'll realise that you're really "shooting in the dark" when trying to guess the ZX81's location, and there is no feedback as to how close you are.

You can add the following lines to give you an idea of how you're going:

```
162 IF A = X AND B <>Y
    THEN PRINT A;" IS
    RIGHT,";B;" IS NOT"
164 IF A <>X AND B = Y
    THEN PRINT A;" IS
    WRONG,";B; " IS
    RIGHT"
```

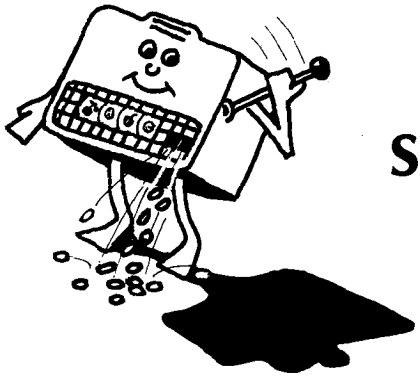
Add these lines, and run the program a few times. You can then add another line to give the player more information:

```
166 IF J = 6 THEN PRINT
    "CLUE: NUMBERS
    ADDED = ";X+Y
```

Of course, many other features can be added. Here's a version of the hunt game which allows the player to select the size of the grid, and gives clues in terms of direction.

```
1Ø  PRINT "HUNT THE
    SPIDER",,"(C)
    HARTNELL 1980",,,,,
2Ø  PRINT "IN THIS GAME
    A SPIDER WILL
    WEAVEA WEB AND
    HIDE ON
    IT",,,, "HEIGHT OF
    WEB—";
3Ø  INPUT X
4Ø  PRINT X,," AND
    WIDTH-";
5Ø  INPUT Y
6Ø  CLS
7Ø  LET A = INT (X*RND)
    + 1
8Ø  LET B = INT (Y*RND)
    + 1
```

```
90  FOR J = 1 TO 5
100 PRINT "THE SPIDER IS
    HIDING ON A",X;" BY
    ";Y;" WEB.",,"WHERE
    IS IT?",,,,,,,,
    "NORTH/SOUTH?
    space";
110 INPUT C
120 PRINT C;" AND
    EAST/WEST?"
130 INPUT D
140 CLS
150 IF C = A AND D = B
    THEN GOTO 300
160 PRINT "GUESS
    NUMBER ";J;" WAS
    ";C;" ";D;" AND WAS",
    "WRONG. TRY TO
    THE";
170 IF C < A THEN PRINT
    "NORTH space";
180 IF C > A THEN PRINT
    "SOUTH space";
190 IF D < B THEN PRINT
    "EAST"
200 IF D > B THEN PRINT
    "WEST"
210 PRINT
220 PRINT
230 NEXT J
240 PRINT "GAME
    OVER",,"SPIDER WAS
    AT ";A;" ";B,
    "ANOTHER GAME?"
250 INPUT A$
260 CLS
270 IF A$ = "YES" THEN
    GOTO 20
280 PRINT "THANKS FOR
    PLAYING"
290 STOP
300 PRINT "YOU FOUND
    THE SPIDER IN ";J;"
    TRIES"
```



Slot/Fruit Machines

Once you've paid for your ZX81, bought some software (and this book) and possibly invested in some additional memory, your wallet may be getting a little light. Here's a program which you can use to get a little money out of your richer friends.

The random number generator (of course) can easily be used to simulate a slot machine, or fruit machine, game. A very simple program of this type, with two "fruits" could be as follows:

```
10  PRINT "SLOT
    MACHINE"
20  LET C = 5
30  LET A = 0
40  FOR J = 1 TO 2
50  LET D = RND
60  IF D >.5 THEN GOTO
    100
70  LET A = A + 1
80  PRINT "ORANGE",
90  GOTO 110
100 PRINT "CHERRY",
110 NEXT J
120 IF A <>1 THEN LET
    C = C + 2
130 IF A = 1 THEN LET
    C = C - 1
140 IF C = 0 THEN GOTO
    210
150 IF C >10 THEN GOTO
    230
160 PRINT "YOU NOW
    HAVE £";C
170 PRINT "PRESS ANY
    KEY FOR NEXT GO"
180 PAUSE 40000
190 CLS
200 GOTO 30
210 PRINT "YOU HAVE
    BUSTED"
220 STOP
230 PRINT "YOU HAVE
    BROKEN THE BANK"
```

Input this program and run it a few times. It works as follows: Line 20 sets the counter for ORANGEs (A) at 0, and line 30 sets the CHERRY counter (B) at 0. Your money is C, and line 20 sets it at £5 to begin the game.

Each play of the game requires two random numbers to be generated. These are counted by line 5Ø (and 13Ø) and generated by line 6Ø. If the random number is 2, program control moves to line 11Ø where CHERRY is printed. Control then goes back to line 5Ø. If the random number is 1, A becomes A + 1, and the computer prints ORANGE before going back to line 5Ø for the next number.

After two numbers have been generated, and the "fruits" printed out, the computer checks to see if both are the same (if they are, A will equal 2 or Ø) and if so, adds two to your money (that is, lets C = C + 2). If the fruits are not the same, the computer takes £1 off you (that is, lets C = C − 1). Can you see how the computer knows the two fruits were different?

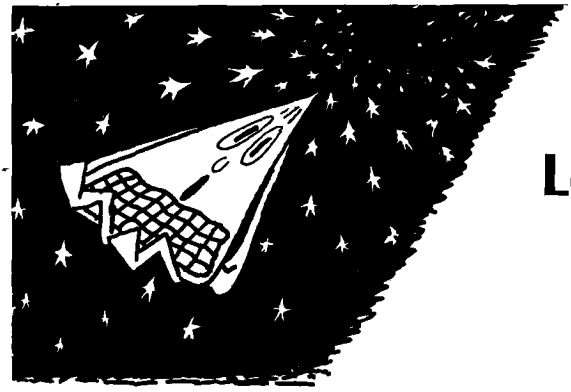
Next, it prints out your stake (line 16Ø). The computer checks this amount. If it is less than £1, control goes to line 23Ø to print YOU HAVE BUSTED. If it is greater than 1Ø control moves to line 25Ø where the computer prints YOU HAVE BROKEN THE BANK. If neither of these conditions are true, the computer acts on the next line (19Ø) which instructs you to press NEWLINE for the next go.

That may seem a little complicated spelt out in detail, but you should be able to follow it through on the program. Once you're sure you understand the workings of this program, write your own version with three fruits, and thus make it more like a "real" slot machine. You'll have to add some lines between others in my program to do this, and change a few lines completely. I'm not going to give you a program for a "three fruits" machine as you will benefit far more from working out your own program to simulate the working of such a machine. And it is far more fun playing a game with a program you've written yourself, than just feeding in someone else's work.

Once you've written and played with the three fruits version, read on to see one way of writing a four fruits slot machine (but with pretty odd fruit). Note that in this program, different winning combinations are worth different amounts, and generate comments.

```
1Ø  PRINT ,"SLOT
    MACHINE"
2Ø  PRINT ,"(C) HARTNELL
    198Ø"
3Ø  PRINT AT
    6,Ø;"STARTING STAKE
    (£ < 2Ø)"
4Ø  INPUT M
5Ø  CLS
6Ø  LET C = Ø
7Ø  LET D = C
8Ø  LET E = C
9Ø  LET F = C
1ØØ FOR J = 1 TO 4
```

```
11Ø LET B = INT (4*RND)
12Ø IF B = Ø THEN GOTO
    18Ø
13Ø IF B = 1 THEN GOTO
    21Ø
14Ø IF B = 2 THEN GOTO
    24Ø
15Ø LET C = C + 1
16Ø PRINT "UNCLE"
17Ø GOTO 26Ø
18Ø LET D = D + 1
19Ø PRINT,"CLIVES"
2ØØ GOTO 26Ø
21Ø LET E = E + 1
22Ø PRINT,"MAGIC"
23Ø GOTO 26Ø
24Ø LET F = F + 1
25Ø PRINT,"ZX81"
26Ø NEXT J
27Ø IF C=1 AND D=1
    AND E=1 AND F=1
    THEN GOTO 32Ø
28Ø IF C=4 OR D=4 OR
    E=4 OR F=4 THEN
    GOTO 32Ø
29Ø IF C=3 OR D=3 OR
    E=3 OR F=3 THEN
    GOTO 35Ø
3ØØ LET M = M - 2
31Ø GOTO 37Ø
32Ø LET M = M + 1Ø
33Ø PRINT,"graphic A
    inverse space JACKPOT
    inverse space graphic
    A"
34Ø GOTO 37Ø
35Ø LET M = M + 2
36Ø PRINT ,"graphic Y
    THREE OF A KIND
    graphic T"
37Ø IF M < 1 THEN GOTO
    43Ø
38Ø IF M > 49 THEN GOTO
    5ØØ
39Ø PRINT ,,,,,,"YOU NOW
    HAVE £";M,,,,"PRESS
    ANY KEY FOR NEXT
    SPIN"
4ØØ PAUSE 4ØØØØ
41Ø CLS
42Ø GOTO 6Ø
43Ø PRINT "END OF
    GAME. YOU ARE
    BROKE"
44Ø PRINT "ANOTHER
    GAME?"
45Ø INPUT B$
46Ø IF B$ = "YES" THEN
    GOTO 3Ø
47Ø CLS
48Ø PRINT "OK. BYE BYE"
49Ø STOP
5ØØ PRINT "YOU HAVE
    £";M;
    "AND
    HAVE","BROKEN THE
    BANK"
51Ø GOTO 44Ø
```



Lost in Space

We'll return now to the HUNTING ON A GRID idea, and work on finding objects which are located on more than two axes, that is, are hidden in three- (or even four-) dimensional space. Imagine our spider is hiding inside a cube, each side of which is X units long. Each point in the cube can be specified by giving three co-ordinates: height, width and depth. Before reading on to see how I've done it, try to work out a program in which a space capsule is lost inside a cube of space, with each side of the cube four kilometres long.

```
10  PRINT TAB 7;"LOST IN
    SPACE"
20  PRINT
30  PRINT "YOU HAVE 15
    HOURS TO
    FIND","CAPSULE LOST
    IN","A 7KM CUBE OF
    SPACE"
40  LET A = INT
    (7*RND+1)
50  LET B = INT
    (7*RND+1)
60  LET C = INT
    (7*RND+1)
70  FOR J = 1 TO 15
80  PRINT,,"INPUT 3
    SEARCH CO-
    ORDINATES"
90  INPUT D
100 INPUT E
110 INPUT F
120 CLS
130 PRINT ,D,E,F
140 IF A = D AND B = E
    AND C = F THEN
    GOTO 270
150 PRINT ,"(graphic A)
    WRONG (graphic
    A)",,,,"HOURS OF AIR
    LEFT - ";15 -
    J,,,"MOVE ";
160 IF A > D THEN PRINT
    "UP ";
170 IF A < D THEN PRINT
    "DOWN ";
180 IF NOT B = E THEN
    PRINT "ACROSS ";
190 IF C > F THEN PRINT
    "FORWARDS"
```

```
200 IF C < F THEN PRINT
    "BACKWARDS"
210 PRINT
220 IF J = 14 THEN PRINT
    "DANGER — DEATH
    IMMINENT"
230 NEXT J
240 CLS
250 PRINT,, "FAIL —
    ASTRONAUTS DEAD"
    ,,,"CAPSULE WAS AT "
    ;A;"  ";B;"  ";C
260 GOTO 280
270 PRINT,,"YOU FOUND
    THEM WITH",,16-J;
    "HOURS OF AIR LEFT"
280 PRINT,,"ANOTHER
    MISSION?"
290 INPUT H$
300 CLS
310 IF H$ = "YES" THEN
    GOTO 30
320 PRINT ,"FAREWELL,
    BRAVE CAPTAIN"
```

Arrays

Arrays, and the use of the DIM (dimension) statement, puzzle many newcomers to BASIC. The DIM statement is used if you want to set up a list with a 'name' (the name is a letter like A, B or whatever). For example, you might want the numbers 1 to 15 in the list named A, in the form LET A(1) = 1, LET A (2) = 2 and so on to LET A(15) = 15. To tell the ZX81 you need an array to hold these *elements* (1, 2 and so on to 15) input:

```
10  DIM A(15)
```

The figure in brackets is number of items or elements you want in the list. The first element is A(1), not A(0). You can have a much bigger array if you like, provided you don't exceed the memory. In ZX81 BASIC you can also have what are called "multidimentional" arrays, which means you can have more than one number inside the brackets. For example you might want a table of size, say, five by six, called A. To set this up type DIM A(5,6). You can now have elements called A(1,1), A(1,2), and so on. The first number cant be bigger than five, and the second number can't be bigger than six. A(3,4) is not the same thing as A(4,3), and if you have a two dimensional array then you can't refer to A(3) for instance - you must always keep to two numbers. Similarly you could have a three dimensional array by typing DIM A(4,4,4). The numbers inside the brackets are known as SUBSCRIPTS, and an element of the form K(9) or F(2,3) is called a subscripted variable.

Here's a program to show the DIM statement:

```
10  DIM A(10)
20  FOR J = 1 TO 10
30  LET A(J) = 2*J
40  PRINT "A(";J;")=";A(J)
50  NEXT J
```

When you run this, you will get:

A(1) = 2...and so on to...
A(10) = 20

Change line 10 to DIM A(5) and run the program again. This time you'll get just A(1) = 2...to...A(5) = 10. The program stopped at this point (giving the error code 3/30, meaning that the subscripted variable required, i.e. J = 6 to J = 10 was out of range). Change line 10 now to DIM A(20) and run the program. You'll find you get exactly the same result as having DIM (A10). As I said before, you change nothing, in practice, by having a larger array than you actually want. Now add the following lines:

```
10  DIM A(10)
60  PAUSE 40000
65  CLS
70  FOR A = 1 TO 15
80  LET A(A) = 3*A
90  PRINT "A(";A;") =
    ";A(A)
100 NEXT A
```

When you run this, you find the same A(0) = 0 through to A(10) = 20 after which the ZX81 will wait for you to press a key. The ZX81 will then display:

A(1) = 3 ...down to...A(10) = 30

The error code 3/80 will be displayed, because the demand made on the array by line 80 was greater than the array defined by line 10 (that is, line 70 made the next A equal 11, and line 80 therefore wanted a subscripted variable called A(11) which, of course, it could not find because the array only had room for 11 elements).

You can also have STRING arrays, but these are slightly different. For a start, if you already have a string called A$ then you can't dimension a string array to be called A$. Another thing to watch for is that all of the elements of a string array have to be the same length, so to set up a string array A$ with five elements you don't just say DIM A$(5) — You have to say DIM A$(5,the number of characters in each element). Run this program and see what happens.

```
10  DIM A$(6,6)
20  FOR A=1 TO 6
30  PRINT "A$(";A;")=";
40  INPUT A$(A)
```

```
5Ø  PRINT "shift Q";
    A$(A);"shift Q"
6Ø  NEXT A
```

Try inputting strings that are not six characters long. You'll notice that those longer than six characters are chopped off after the sixth place, and those shorter are filled with spaces where necessary. Here is a program to show the DIM statement in use.

Now follows a very simple HUNT THE HURKLE program using the DIM statement. Input the program, run it a few times, then read through to find out how it works.

```
1Ø  PRINT "DIM SPIDER"
2Ø  DIM A(4)
3Ø  DIM B(4)
5Ø  FOR G = 1 TO 4
6Ø  LET A(G) = 2
7Ø  LET B(G) = 2
8Ø  NEXT G
9Ø  LET K = INT
    (4*RND)+1
1ØØ LET L = INT
    (4*RND)+1
11Ø LET A(K) = 1
12Ø LET B(L) = 1
13Ø FOR D = 1 TO 5
14Ø PRINT "WHERE IS
    SPIDER, TRY
    NUMBER ";D
15Ø INPUT T
16Ø INPUT Y
17Ø IF A(T) = 1 AND B(Y)
    = 1 THEN PRINT
    "YOU FOUND IT"
18Ø IF A(T) = 1 AND B(Y)
    = 1 THEN STOP
19Ø NEXT D
2ØØ PRINT "SORRY, TIME
    IS UP"
21Ø PRINT "SPIDER WAS
    AT";K,L
22Ø STOP
```

Having run this program a few times, and examined the listing, you are probably asking yourself what was achieved by using the DIM statements which could not have been achieved without them. The answer is: Nothing. However, the DIM comes into its own when you want to hide more than one object on a grid, without creating a whole set of co-ordinates of the type LET A = INT (4*RND)+1, LET B = INT (4*RND)+1, LET C = INT (4*RND)+1 and so on, to put the first hidden object at A,B; the second at C,D; and so on.

Before we look at this, here are two more programs which hide a single object.

```
5   PRINT "DIMMER
    SPIDER"
1Ø  DIM A(2)
2Ø  DIM B(2)
3Ø  FOR C = 1 TO 2
```

```
40  LET A(C) = INT
    (4*RND)+1
50  NEXT C
60  FOR J = 1 TO 7
70  PRINT "WHERE IS THE
    SPIDER —
    ATTEMPT ";J
80  FOR C = 1 TO 2
90  INPUT B(C)
100 NEXT C
110 PRINT B(1)," ";B(2);
120 IF A(1) = B(1) AND
    A(2) = B(2) THEN
    GOTO 200
130 IF A(1) = B(1) AND
    A(2) <> B(2) THEN
    GOTO 220
140 IF A(1) <> B(1) AND
    A(2) <> B(2) THEN
    GOTO 240
150 PRINT " NO"
160 IF J = 5 THEN GOSUB
    260
170 NEXT J
180 PRINT "TIME IS UP.
    SPIDER WAS AT ";A(1);
    " ";A(2)
190 GOTO 205
200 PRINT " YOU HAVE
    FOUND IT"
210 STOP
220 PRINT A(1); " IS
    RIGHT"
230 GOTO 170
240 PRINT A(2); " IS
    RIGHT"
250 GOTO 170
260 PRINT "HINT:
    LOCATIONS ADD UP
    TO ";A(1)+A(2)
265 PAUSE 200
270 RETURN
```

If you like, you can change line 160 to read:

```
160 IF J = 3 OR J = 7 THEN GOSUB 260
```

This just reinforces the same hint when J = 7 as when J = 3.

If you want the 'hint' to come at random, you could change line 160 to:

```
160 IF J = 2*A(1) THEN GOSUB 260
```

or to:

```
160 IF J = 2*A(2) THEN GOSUB 260
```

Another version of this line, suggested by Ian Rodgers, is:

```
160 IF 7*RND < J THEN GOSUB 260
```

This final version is probably the best, because it means that the closer to J = 7 you get, the more likely you are to be given a hint.

The core of the program DIMMER SPIDER can be used to produce a far more interesting game, and in this program we will introduce a new ZX81 function.

```
5   PRINT "PESKY PIKSY"
10  RAND
15  CLEAR
20  DIM A(2)
25  DIM B(2)
30  FOR C = 1 TO 2
```

```
40  LET A(C) = INT
    (7*RND)+1
50  NEXT C
60  FOR J = 1 TO 11
70  PRINT "WHERE IS
    PIKSY — ATTEMPT ";J
80  FOR C = 1 TO 2
90  INPUT B(C)
100 NEXT C
110 PRINT B(1);" ";B(2);
120 IF A(1) = B(1) AND A(2)
    = B(2) THEN GOTO
    200
130 IF A(1) = B(1) AND
    NOT A(2) = B(2) THEN
    GOTO 235
140 IF NOT A(1) = B(1)
    AND A(2) = B(2) THEN
    GOTO 250
150 PRINT " NO"
160 IF 10*RND < J THEN
    GOSUB 260
170 IF J = 4 OR J = 7
    THEN CLS
175 NEXT J
180 PRINT "TIME IS UP.
    PIKSY WAS AT
    ";A(1);" ";A(2)
190 GOTO 305
200 CLS
225 PRINT AT 10,0;"YAY
    (graphic A)
    CAPTURED"
230 GOTO 305
235 PRINT A(1); " IS
    RIGHT"
240 GOTO 170
250 PRINT A(2); " IS
    RIGHT"
255 GOTO 170
260 LET G = INT (5*RND)
270 IF G = 0 THEN PRINT
    "HINT: LOCATIONS
    ADD UP TO"; A(1) +
    A(2)
280 IF G = 1 THEN PRINT,
    "HINT: DIFERENCE
    BETWEEN LOCATIONS
    IS ";ABS(A(1) — A(2))
290 IF G >= 2 THEN
    GOTO 320
300 RETURN
310 STOP
320 PRINT "I WAS AT:"
    A(1);" ";A(2);" NOW"
330 PRINT "I AM
    MOVING"
340 PRINT "PRESS ANY
    KEY"
350 PAUSE 40000
360 CLS
370 GOTO 10
```

Have a look at line 280. The function ABS stands for "absolute". If a number is positive, the absolute value of that number is just the number. If a number is negative, the absolute value is the number multiplied by −1 (i.e., the number without its negative sign). If you leave out ABS in line 280, the clue will be so specific it will almost certainly give the location away, so ABS makes for a better game.

The next program locates a number of objects (10) on a grid, and gives a score based, of course, on how many you hit.

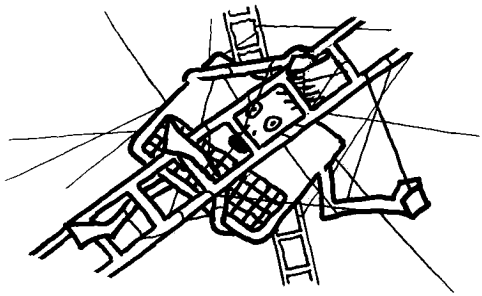
However, if more than one object is at the same location you get more than one score. The program also awards you a "rating" at the end.

```
2    PRINT "ALIENS + ASTEROIDS"
5    LET K = Ø
2Ø   DIM A(1Ø)
3Ø   DIM B(1Ø)
4Ø   DIM C(1Ø)
5Ø   DIM D(1Ø)
1ØØ  FOR J = 1 TO 1Ø
11Ø  LET A(J) = INT (4*RND)+1
12Ø  LET B(J) = INT (4*RND)+1
13Ø  NEXT J
16Ø  FOR Z = 1 TO 8
17Ø  PRINT CHR$(RND*1Ø+128); "SHOT";Z;"graphic 5";
18Ø  INPUT C(Z)
19Ø  INPUT D(Z)
2ØØ  PRINT C(Z);" ";D(Z);" ";
23Ø  FOR J = 1 TO 8
24Ø  IF A(J) = C(Z) AND B(J) = D(Z) THEN LET K = K + 1
25Ø  IF A(J) = C(Z) AND B(J) = D(Z) THEN PRINT "HIT inverse space SCORE: ";K
26Ø  IF K = 1Ø THEN GOTO 43Ø
27Ø  NEXT J
28Ø  NEXT Z
29Ø  CLS
31Ø  PRINT AT 2,8; "two inverse space one space TIME IS UP"; AT 4,8;"two graphic T YOU HIT ";K;" ALIENS two graphic Y"; AT 6,8; "MARKSMAN RATING:";INT (1ØØ*K/(3*RND+1.5)); AT 8,5; "THEY ARE HIDDEN AT:-"
39Ø  FOR J = 1 TO 1Ø
4ØØ  PRINT, A(J); " ";B(J)
41Ø  NEXT J
42Ø  STOP
43Ø  CLS
45Ø  PRINT TAB 5
47Ø  PRINT "eight inverse space YOU GOT THEM ALL eight inverse space eight space YOU GOT THEM ALL seven space";
51Ø  GOTO 47Ø
```

The screen display for this game when run is, unfortunately, a little cramped. You'll need more than 1K for this program.

You'll notice a randomly generated graphical character before the word SHOT each time. These graphics are the inverse of the graphics available direct from the keyboard. The number of every character is given in the manual, and to print the character corresponding to a particular number, you just input PRINT CHR$ number of character. The black blob you get on a hit is the inverse of the space (character number 128). The same character is used at the end if you manage to destroy all the

aliens. The "marksman rating" at the end adds a little interest, and is actually related to your skill in destroying the aliens. The more you killed (K) the higher your score. It is divided by (3*RND+1) just to make it a little more interesting.



Strings and Ladders

The discipline of programming in just 1K will stand you in good stead when later you add extra memory. It is very easy to get into the habit of programming sloppily, inefficiently or inelegantly when you have memory to spare. One way of making the most of your 1K is to use variables, assigned at the start of the program, for constants which you intend to use at various parts of the program. It turns out that if any constant is used MORE THAN THREE TIMES in a program then a considerable amount of memory is saved by assigning it to a variable instead. The following game SNAKES AND LADDERS shows these techniques in use.

```
1    LET P = 1
2    LET Q = P + P
10   DIM C$(Q,P+P+Q)
20   PRINT "SNAKES AND
     LADDERS";TAB P
30   FOR Z = P TO Q
40   PRINT
     "PLAYER";Z;TAB P
50   INPUT C$(Z)
60   NEXT Z
70   CLS
80   PRINT "TO START
     GAME";
90   DIM A(Q)
100  FOR Z = P TO Q
110  PRINT "PRESS
     NEWLINE
120  INPUT M$
130  CLS
135  PAUSE 200*RND
140  LET M$ = "SNAKE"
```

```
15Ø LET E = SGN
   (RND-.4)*INT
   (6*RND+P)
16Ø IF E >= P THEN LET
   M$ "LADDER "
17Ø LET A(Z) = A(Z) + E
18Ø PRINT C$(Z);"-";
   M$;"WORTH ";E;TAB
   7;"SCORE IS
   ";A(Z);TAB P;TAB P-P
19Ø IF A(Z) >= 2Ø THEN
   GOTO 23Ø
2ØØ PRINT "FOR NEXT
   MOVE space";
21Ø NEXT Z
22Ø GOTO 1ØØ
23Ø PRINT "inverse
   space";C$(Z);" WINS
   BY ";
   A(Z)-A(P+Q-Z);
   "POINTS", "ANOTHER
   GAME?"
24Ø INPUT M$
25Ø IF M$ = "YES" THEN
   GOTO 7Ø
26Ø PRINT "OK. BYE"
```

Notice the delay in line 135. This is randomly anything between nothing and about four seconds.

Look at line 14Ø to 16Ø. These determine whether a player will get a SNAKE (and a negative score) or a LADDER (and a positive score). Ti uses the function SGN which we haven't met before. SGN gives an answer of one if the number after it is positive, or minus one if the number after it is negative. Since RND−.4 will be positive about 6Ø% of the time, each player has slightly more chance of getting a ladder than a snake. If line 15Ø had read LET E = SGN (RND−.5)*INT(6*RND+P) there would be a pretty good chance the game would never end, since each player's gains would approximately equal their losses, and the players would lose interest very quickly.

The next game is based on exactly the same ideas as SNAKES AND LADDERS but takes longer to play, and - because it has more variables - is considerably more interesting.

We are going to introduce a new and very useful trick here, called SLICING. Suppose we had a string called A$ containing "ABCDEFGHIJ", and in our program we wrote A$(4 TO 8). (TO is the keyword on key 4). This would mean the segment of A$ from the 4th character to the 8th - in this case "DEFGH". We can also SLICE single characters - A$(9) is just a short way of writing A$(9 TO 9). Remember you can't have an array called A$ and a string called A$ both at the same time, so the ZX81 doesn't get confused about what we mean by A$(9). You don't have to use variables either - you could just as well write "ABCDEFGHIJ"(9) which would mean the 9th character of "ABCDEFGHIJ", or "I". The number in brackets can be a variable -"ABCDEFGHIJ"(N)

will be different depending on the value of N. If N is one it will mean "A", if N is five it will mean "E", and so on. We shall use SLICING, in conjunction with a new function CODE in this next program, ROADRACE - see if you can work out what lines 18Ø and 2ØØ are doing.

```
1    LET P = 1
2    LET Q = P + P
1Ø   PRINT "ROADRACE"
2Ø   DIM A(Q)
3Ø   DIM A$(Q,P+P+Q)
4Ø   RAND
5Ø   FOR Z = P TO Q
6Ø   PRINT "DRIVER ";Z
7Ø   INPUT A$(Z)
8Ø   NEXT Z
9Ø   LET A(P)=39Ø
1ØØ  LET A(Q)=A(P)
11Ø  PRINT "PRESS
     NEWLINE"
12Ø  INPUT M$
13Ø  CLS
14Ø  FOR Z=P TO Q
15Ø  PRINT A$(Z)
155  PAUSE 1ØØ
16Ø  LET C=INT
     (11*RND)-Q-Q
17Ø  IF C < P THEN LET
     C=P
18Ø  LET A(Z)=A(Z)+24
     -CODE "N graphic 5 )
     £< graphic 1"(C)
19Ø  IF A(Z)< P THEN
     GOTO 23Ø
2ØØ  PRINT "ALL OK
     SMASH COPS-
     PUNCTUREPETROL-
     OIL"(CODE "graphic 1
     graphic E $);3"(C) TO
     CODE "graphic T £(/25"
     (C));" ";A(Z);"space
     MILES TO GO";TAB P
21Ø  NEXT Z
22Ø  GOTO 11Ø
23Ø  PRINT " WINS BY
     ";4Ø*(A(P+Q-Z)
     -A(Z));"POINTS"
```

SLICING is used in line 18Ø to find the number of points to be subtracted. If C is one then the SLICE will be "N", and so the CODE is CODE "N", or 51. The score you get is then 24 − 51 or −27. It is negative because the score is the number of miles LEFT TO GO. If C were two the SLICE would be "graphic 5" and the score would be 24 − CODE "graphic 5" which happens to be 19, ie you go BACK 19 miles. SLICING is used very efficiently in line 2ØØ. Suppose C were equal to 4. The first SLICE would be "graphic 1 graphic E $);3"(4), which means ")". The CODE of ")" is 17. The second SLICE is "graphic T £(/25"(4) which is "/", and CODE of "/" is 24. You can find all of characters' CODEs PRINT "ALL OK SMASH COPSPUNCTUREPETROLOIL"(17 TO 24), which means the segment between the 17th and 24th characters, or "PUNCTURE". See if you can work out what would happen if C were 5. (Hint: the CODE of ";" is 25 and the CODE of "2" is 3Ø).

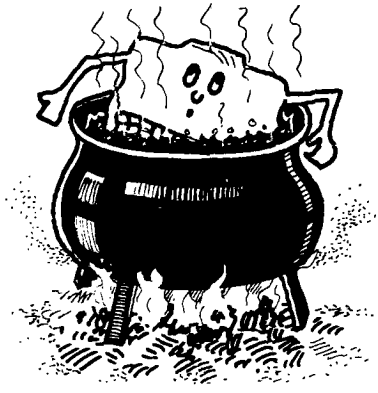
In this game, as in SNAKES AND LADDERS, there is a slightly better than even chance of getting an ALL OK (and a positive mileage). See if you can find the line in the listing that ensures this.

Comparing line 135 in SNAKES AND LADDERS with line 155 in ROADRACE is instructive. In the first program, the delay is random (and varies from a delay of practically zero when RND is small to a much longer period when RND is large). In ROADRACE, the delay is set (purely arbitrarily at 100). There is no reason why you can't set the delay in either program to zero (delete the PAUSE statement, but leave in the CLS) or any number you like, or — if you prefer the unexpected — at a random number. Do not set the random limit too high (like, say, PAUSE 50000*RND because you run the risk of (a) losing interest in the game if the delay is close to 5000 time and again; and (b) you may think you've just got a long delay when, in fact, your ZX81 has gone into an infinite loop for some reason (this is likely if you've either made a mistake when inputting the program, or with some ZX81's, the computer has become very hot).

The next game — 52 BLUFF — uses a different string idea. Here we have assigned all the most commonly used phrases to strings at the start of the program, in the same way that we did with variables in the previous programs.In this game the ZX81 deals two cards. If you think the next card to be dealt will lie between the first two, you place a bet of your choice. This game is more interesting than some computer betting games like FRUIT MACHINE because you can decide on the likelihood of a win and adjust your bet accordingly. You can even decide not to bet at all.

```
10  LET S = 30
20  LET A$ = "  "
30  LET F$ = "CARD 1: "
40  LET G$ = "CARD 2: "
50  LET H$ = "CARD 3: "
60  LET A =
    INT(13*RND)+1
70  LET B =
    INT(13*RND)+1
80  IF B = A THEN GOTO
    70
90  PRINT "two graphic 5
    STAKE: £";S;"two
    graphic 8"
100 PRINT F$;A$;
110 LET Z = A
120 GOSUB 460
130 PRINT
140 PRINT G$;A$;
150 LET Z = B
160 GOSUB 460
170 PRINT "WAGER?"
180 INPUT C
190 CLS
200 IF C = 0 THEN PRINT
    "COWARD"
210 PRINT A,B
```

```
22Ø LET D =
    INT(13*RND)+1
23Ø IF D = A OR D = B
    THEN GOTO 22Ø
24Ø PRINT
25Ø PRINT H$;A$;
26Ø LET Z = D
27Ø GOSUB 46Ø
28Ø IF A < D AND D < B
    THEN GOSUB 36Ø
29Ø IF A > D AND D > B
    THEN GOSUB 36Ø
3ØØ IF D > B AND D > A
    THEN GOSUB 41Ø
31Ø IF D < B AND D < A
    THEN GOSUB 41Ø
32Ø IF S <1 THEN GOTO
    44Ø
33Ø INPUT K$
34Ø CLS
35Ø GOTO 6Ø
36Ø LET S = S + 2*C
37Ø IF S >199 THEN PRINT
    "YOU HAVE BROKEN
    THE BANK"
38Ø IF S >199 THEN STOP
39Ø IF C < >Ø THEN PRINT
    "YOU WIN £";2*C
4ØØ RETURN
41Ø LET S = S - C
42Ø IF C < > Ø THEN
    PRINT "YOU LOSE
    £";C
43Ø RETURN
44Ø PRINT ,"inverse space
    YOU ARE BROKE"
45Ø STOP
46Ø IF Z >1 AND Z <
    = 1Ø THEN PRINT "A"
47Ø IF Z = 1 THEN PRINT
    "A"
48Ø IF Z = 11 THEN PRINT
    "JK"
49Ø IF Z = 12 THEN PRINT
    "QN"
5ØØ IF Z = 13 THEN PRINT
    "KG"
51Ø RETURN
```



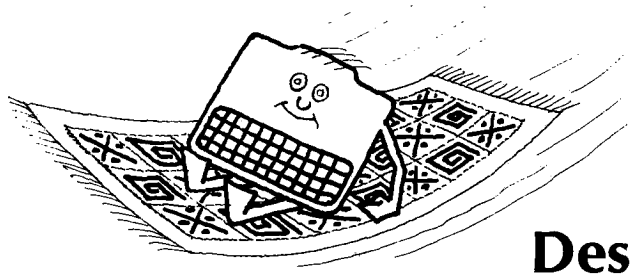
Hot Sauce

You will recall that earlier in this book there was program in which the ZX81 thought of a number, then gave you hints to help you guess it. You probably realised that if you started with 5Ø as your first guess, it was pretty easy to narrow down the number by going to either 25 or 75 on the second guess. This method allowed you to get the correct number fairly easily. Here is another program which appears somewhat similar. But it is far harder to work out a system to beat it.

```
1Ø   PRINT "HOT SAUCE"
2Ø   PRINT  "(C) HARTNELL
     1980"
3Ø   PRINT
4Ø   PRINT "WHATS YOUR
     NAME, PARDNER?"
5Ø   INPUT A$
6Ø   CLS
7Ø   PRINT
8Ø   PRINT
9Ø   LET S = Ø
1ØØ  PRINT "OK, ";A$;", I
     AM THINKING OF"
11Ø  PRINT "A NUMBER
     BETWEEN 1 AND 1ØØ"
12Ø  PRINT "YOU HAVE 12
     GUESSES"
13Ø  LET J =
     INT(99*RND)+1
14Ø  PRINT
15Ø  LET S = S + 1
16Ø  IF S = 13 THEN GOTO
     42Ø
17Ø  PRINT "WHAT
     NUMBER AM I
     THINKING OF?"
18Ø  INPUT A
19Ø  CLS
2ØØ  IF A = J THEN GOTO
     36Ø
21Ø  IF A < J THEN GOTO
     3ØØ
22Ø  IF A - J < 5 THEN
     PRINT "BOILING, ";A$
23Ø  IF A - J < 12 AND A
     - J > 4 THEN PRINT
     "HOT"
24Ø  IF A - J < 25 AND A
     - J > 11 THEN PRINT
     "WARM"
25Ø  IF A - J < 45 AND A
     - J > 24 THEN PRINT
     "COLD"
```

```
26Ø IF A − J > 44 THEN
     PRINT "FREEZING,
     BABY"
27Ø PRINT
28Ø PRINT ,"NEXT
     GUESS?"
29Ø GOTO 15Ø
3ØØ IF J − A < 5 THEN
     PRINT "VERY, VERY
     CLOSE"
31Ø IF J − A < 12 AND J
     − A > 4 THEN PRINT
     "PRETTY CLOSE"
32Ø IF J − A < 25 AND J
     − A > 11 THEN PRINT
     "JUST SO−SO"
33Ø IF J − A < 45 AND J
     − A > 24 THEN PRINT
     "PRETTY HOPELESS"
34Ø IF J − A > 44 THEN
     PRINT
     "PATHETIC, ";A$
35Ø GOTO 28Ø
36Ø PRINT "YOU WERE
     RIGHT, ";A$
37Ø PRINT "I WAS
     THINKING OF ";J
38Ø PRINT "SO YOU GET
     ANOTHER GO"
39Ø LET S = Ø
4ØØ GOTO 13Ø
41Ø CLS
42Ø PRINT "SORRY, ";A$;
     ", YOU DIDNT GUESS
     IT"
43Ø PRINT
44Ø PRINT "I WAS THINK
     OF ";J
```

The term "absolute" (ABS on the keyboard) was introduced just after the listing for PESKY PIKSY. Refer back to this if you're not sure what ABS does. The HOT SAUCE program could be written quite differently from the above, by using the ABS. In the listing here, the ZX81's response to your guess is determined by whether the number you guess is higher or lower than the one it is thinking of, *and* by the difference between the numbers. If you rewrite HOT SAUCE using ABS, you'll find that only the difference between the numbers will determine the comment ("PATHETIC" or "BOILING" or whatever) the ZX81 will make. And just as HOT SAUCE is more difficult to play than GUESS MY NUMBER, HOT SAUCE with ABS is harder than without it. As an exercise, write a HOT SAUCE program using ABS. You should find it uses less memory than the above listing. You could also program the ZX81 to make much longer comments on your guesses.



Designs on your VDU

We have used the RND function time and time again in programs. It is one of the most useful features for games. But, as I pointed out earlier, the random number generated is not really a random number at all, but is one of a very, very long list of numbers, so long that the number appears to be random. The use of the RAND function (on the T key) sets the starting point of the long, long list of numbers to a number related to the number of times your TV screen has been scanned since the ZX81 was turned on. This next program reads the number of frames time and time again, each time a little later than before, and uses this number to generate a random number which, in turn, is used to instruct the ZX81 to print a specific "character".

You'll find in your manual a list of characters and their corresponding codes.

If you told your computer to print CHR$ 12 you would find it would print the pound sign (£). CHR$ means the character whose code follows. A simple program to fill the screen with randomly generated characters, and their corresponding codes, is as follows:

```
10  LET K = 63*RND
20  PRINT "CODE NO. ";K;
    " IS ";CHR$ K
30  GOTO 10
```

You will notice that K is not INTed before the character is printed. This is because it doesn't need to be - CHR$ will automatically round a number up or down to the nearest integer, so CHR$ 42.4 is the same thing as CHR$ 42, and CHR$ 6.9 is the same thing as CHR$ 7. In effect it will add 0.5 and then INT it so that CHR$ 55.5 is the same as CHR$ 56, not CHR$ 55. By letting K = 63*RND we are allowing for anything from character zero (space) to character 63 (Z). The inverse characters

have codes Ø to 1Ø and 128 to 138. You can fill the screen with these by running the following little program:

```
1Ø  LET K = 1Ø*RND
2Ø  IF RND < .5 THEN LET
    K = K + 128
3Ø  PRINT CHR$ K;
4Ø  GOTO 1Ø
```

By combining, more or less, the preceding program with the RAND function, we can develop an interesting program. Try the following:

```
1Ø  FOR J = 1 TO 2Ø STEP
    2
2Ø  PRINT AT J,J;
3Ø  NEXT J
4Ø  PRINT "IF YOU PRESS
    NEWLINE I WILL"
5Ø  PRINT "CREATE A
    CARPET DESIGN FOR
    YOU"
6Ø  INPUT A$
7Ø  CLS
8Ø  LET G = 3*RND
9Ø  PRINT "CODE NAME
    FOR DESIGN:—
    inverse quote";
1ØØ FOR H = 1 TO 5 + G
11Ø LET D = RND*22
12Ø PRINT CHR$(D + 166);
13Ø NEXT H
14Ø PRINT "inverse quote"
15Ø PRINT
16Ø FOR J = 1 TO 48
17Ø RAND
18Ø LET Z = RND*1Ø
19Ø PRINT CHR$(Z + 128);
2ØØ NEXT J
21Ø FOR J = 1 TO 45 + G
22Ø LET Z = RND*1Ø
23Ø PRINT CHR$ Z;
24Ø NEXT J
25Ø GOTO 16Ø
```

There are several things we can learn from this program. Lines 1Ø to 3Ø simply print the title ten times, which makes a more interesting start than just having the title printed once near the top of the screen. Line 8Ø sets G equal to a random number in the range Ø to 3. The value of G is used in lines 1ØØ and 21Ø. Lines 9Ø to 14Ø create a name for the design, by printing the characters (all inverse letters) whose codes are generated by line 12Ø. The naming part of the program is followed by two FOR/NEXT loops, linked by a final GOTO statement. The RAND is within the first FOR/NEXT loop.

Once you've run this program a few times, rewrite it by adding a further FOR/NEXT loop (with the FOR in line 155, and the NEXT in place of 25Ø) to stop the program before it crashes because the screen is full, and to allow you to have another go without having to go back into command mode first. Set the limit of this last loop so as to get the maximum amount of "carpet" on the screen. Experiment with the size of the other

two loops, and with the maximum value of G and see what this does to the final display.

If your machine can work in SLOW mode try deleting the FOR/NEXT loop you've just put in and insert these lines instead.

```
155 LET X = 31
165 LET X = X + 1
166 IF X < 32 THEN GOTO
    17Ø
167 LET X = Ø
168 SCROLL
215 LET X = X + 1
216 IF X < 32 THEN GOTO
    22Ø
217 LET X = Ø
218 SCROLL
25Ø GOTO 16Ø
```

First Steps Towards the Stars

Most computer systems in the world, micro to mainframe, have at least one space war-type game in their library. The limited memory on the basic ZX81 precludes all but the simplest versions of this old favourite. However, study of the following program will teach you some of the fundamentals of star games, and will give you a core to build on when you buy extra K for your machine.

```
1    LET P = 1
2    LET Q = P + P
1Ø   LET R = 31Ø
2Ø   PRINT "TIMEWARP";
     TAB P
3Ø   PRINT "space
     TREASURE?";TAB P;"2
     ENEMIES?"
4Ø   INPUT T$
5Ø   INPUT E$
6Ø   INPUT F$
7Ø   LET H = Q*Q*Q*Q
8Ø   LET G=H/Q*Q
9Ø   GOSUB R
1ØØ  LET L$ = E$
11Ø  IF RND < P/Q THEN
     LET L$ = F$
12Ø  IF G > = P THEN
     GOTO 15Ø
13Ø  PRINT "YOU ARE
     DEAD"
14Ø  STOP
15Ø  PRINT "SHIELD
     ";G,"TIME
     ";H,"DANGER ";L$
16Ø  LET H = H - P
17Ø  GOSUB R
18Ø  IF H > = P THEN
     GOTO 21Ø
19Ø  PRINT "TIMEWARP
     HAS IMPLODED",
     "YOU HAVE FAILED"
2ØØ  STOP
21Ø  IF RND < P/Q THEN
     GOTO 25Ø
22Ø  LET G = G - P
23Ø  PRINT "THE ";L$;"GOT
     YOU"
24Ø  GOTO 9Ø
25Ø  LET G = G + P
26Ø  PRINT "YOU KILLED
     THE ";L$, "YOU ARE
     CLOSER TO THE ";T$
27Ø  GOSUB R
28Ø  IF RND < .8 THEN
     GOTO 9Ø
29Ø  PRINT
     "CONGRATULATIONS
     ",,"YOU GOT THE
     ";T$, "OUT OF THE
     TIMEWARP","WITH
     ";H;"TIME UNITS TO
     SPARE"
3ØØ  STOP
31Ø  PAUSE R/Q
32Ø  CLS
33Ø  RETURN
```

If this program followed a shorter version of the next program we will look at, and if the value of H could be used to print the "sector of the galaxy" we were in (i.e. IF H >3 AND H <7 THEN PRINT "YOU ARE IN SIRIUS SECTOR") we would be well on the way to creating a much better "space game". If you decide to add extra K, you could start by modifying this program. To give you room to move, multiply each line number by 1Ø.

In this program, for the first time, we access the ZX81's frame counter. This allows us to add a time dimension to programs.

Input the following program, and run it, keeping in mind that your reactions to commands have a time limit to them. If you exceed the time, you'll get an ABORT MISSION display.

```
1    PRINT AT 7,11;"BLAST
     OFF "
2    PRINT AT 9,Ø;"TO
     START TAKE OFF
     ROUTINE"
3    PRINT "PRESS ANY
     KEY"
4    PAUSE 4ØØØØ
5    RAND
6    FOR J = 1 TO 2Ø
7    LET H = 1Ø*RND
8    CLS
9    PAUSE 2ØØ*RND
1Ø   PRINT AT 22*RND,Ø;
11   LET A = Ø
12   LET B = A
13   POKE 16436,A
14   POKE 16437,A
15   GOSUB 1ØØ + 1Ø*INT
     (4*RND)
16   LET A = PEEK 16436
17   LET B = PEEK 16437
18   IF J = 2Ø AND
     65536-A-256*B < 15Ø
     THEN GOTO 2ØØ
19   IF 65536-A-256*B<
     15Ø THEN NEXT J
2Ø   PRINT AT 11,5;"ABORT
     MISSION AT"
21   PRINT AT
     11,22;65536-A-256*B
22   LET B = B + 1
23   GOTO 21
1ØØ  PRINT "inverse space
     SET FUEL
     TO";INT(2ØØ*H*H/-
     3)-H
1Ø1  INPUT K
1Ø2  IF K = INT
     (2ØØ*H*H/3)-H THEN
     RETURN
1Ø3  GOTO 2Ø
11Ø  LET A$ = CHR$ (H +
     38)
111  PRINT "four graphic 4
     INPUT AIR PRESSURE
     VALVE ";A$
112  INPUT B$
113  IF B$ = A$ THEN
     RETURN
114  GOTO 2Ø
12Ø  PRINT "SIGNAL
     keyword TO ALIEN
     ";CHR$
     (168+J);CHR$(6*H);-
     CHR$
     (166+2*H);"WITH shift
     Q";CHR$(H+48);"shift
     Q"
121  INPUT B$
122  IF B$ = CHR$(H+48)
     THEN RETURN
123  GOTO 2Ø
13Ø  PRINT "two graphic T
     GIVE IDENTITY
     NUMBER ";1ØØ*H-INT
     (H/3)+J
131  INPUT K
132  IF K = 1ØØ*H-INT
     (H/3)+J THEN RETURN
133  GOTO 2Ø
2ØØ  SCROLL
21Ø  PRINT "two graphic T
     WE HAVE LIFT OFF
     two graphic T"
22Ø  SCROLL
23Ø  PRINT "two graphic T
     WE HAVE LIFT OFF
     two graphic T"
24Ø  SCROLL
25Ø  PRINT "two graphic T
     WE HAVE LIFT OFF
     two graphic T"
26Ø  GOTO 2ØØ
```

You'll need SLOW to RUN this program as it stands, but you can make it run in FAST by ignoring the AT's in lines 20 and 21, changing 200 to CLS, and deleting 220 and 240. The program uses a number of ideas we've been introduced to recently. Lines 8 and 9 introduce a random delay after clearing the screen. As you run this program you'll see the commands appear at different points on the screen. They are placed there by the random PRINT AT in line 10. The timer starts with the POKE statements (lines 13 and 14) where the count is reset, and the ZX81 then keeps counting backwards, frame by frame, from 65536, until you get to the PEEK statements (lines 16 and 17) via the randomly selected (line 15) subroutine.

The next program, can be made as heart-stopping as you like.

```
1    LET P = 1
2    LET Q = P + P
3    LET R = 16436
4    LET S = 256
10   PRINT "FRENZY";AT
     Q,P-P;"TO RISK
     SANITY PRESS N/L"
20   INPUT A$
30   LET G =
     P+Q+Q+RND*SQR S
40   CLS
50   FOR J = P TO G
60   IF J <>1 THEN PRINT
     "TIME LAST GO ";A
70   PRINT "TEST NO. ";J;"
     OUT OF ";INT G,
     "YOU HAVE space
     inverse space";
     S/Q-J*SQR S;"inverse
     space space NUT
     SECONDS";AT
     11*RND+Q+Q,P-P;
80   LET F =
     INT(G*Q+J*(P+Q-
     )/Q-SQR S*RND)
90   POKE R, P - P
100  POKE R + P, P - P
110  PRINT "OK DUM DUM
     keyword COPY THIS
     NO",F
120  INPUT K
130  LET A = S*S-PEEK
     R-S*PEEK (R + P)
135  CLS
140  IF K <> F THEN
     PRINT "YOU CANT
     EVEN keyword COPY
     NOS"
150  IF A > 340-J*Q*Q*Q
     THEN PRINT "YOURE
     FAR TOO keyword
     SLOW"
160  IF K = F AND A <
     340-J*Q*Q*Q THEN
     NEXT J
170  PRINT "YOUR SANITY
     RATING IS space ";
     A*J/Q/Q/Q+Q*Q*-
     Q*RND
     ,"AGAIN?"
180  INPUT A$
190  IF A$ <> "NO" THEN
     GOTO 40
```

Doing it in Your Head

The next two programs do not have timers, but could well be adapted to have a real time limitation if you like.

```
10  PRINT "UNICORNS AND
    GRIFFINS"
20  LET G = 6 +
    INT(5*RND)
30  PRINT
40  PRINT "DEGREE OF
    DIFFICULTY (1 TO 5)?"
50  INPUT Q
60  IF Q < 1 OR Q > 5 THEN
    GOTO 40
70  PRINT "PRESS NEWLINE
    TO START"
80  INPUT A$
90  FOR J = 1 TO G
110 CLS
120 PAUSE 200*RND
160 LET F = 2*G + 3*J/2
170 LET Z =
    Q*(INT(15*RND)+10*J)
180 PRINT AT
    6*RND,0;"UNICORN shift
    Q S NUMBER IS ";F
190 PRINT "WHAT DOES
    GRIFFIN ADD TO"
200 PRINT "MAKE UNICORN
    shift Q S NUMBER =
    ";Z;"?"
210 INPUT K
220 IF K + F = Z THEN
    NEXT J
230 PRINT
240 LET T =
    F*INT(100*RND+2)
250 IF J = 1 THEN LET
    T = 0
260 PRINT "SCORE FOR
    THAT ROUND WAS ";T
270 IF J <> G AND K + F <
    > Z THEN GOSUB 330
280 IF J = G THEN GOSUB
    360
290 PRINT "DO YOU WANT
    TO TACKLE THE"
300 PRINT "UNICORN
    AGAIN?"
310 INPUT H$
320 IF H$ <> "NO" THEN
    GOTO 20
325 STOP
330 IF K + F <> Z THEN
    PRINT "THE UNICORN
    BEAT YOU"
340 RETURN
350 PRINT
360 PRINT "YOU HAVE
    BEATEN THE UNICORN"
370 PRINT "YOUR IQ IS ";
    T*J + J
380 RETURN
```

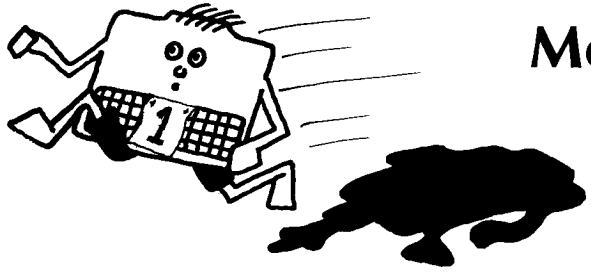
This program introduces an idea which you can use in many games — the "degree of difficulty". Generally, the "degree" can be used directly, to multiply or divide something, or to be added to or taken away from the limits on a FOR/NEXT loop. In other games, you might have to add lines like (if, say A was the degree): IF A = 1 THEN LET G = 200; or IF A > 7 THEN GOSUB 90. Look back at some of the earlier programs in this book, and work out ways of modifying them in the light of later things you have learned.

The "degree of difficulty" can easily be worked into the following program. However, as it becomes more and more difficult already as it proceeds, it might be better to make it a little easier before adding the option of increasing the difficulty. Line 280 is the one to modify to make the game simpler, and it is here that the "degree' factor can be added.

```
10  PRINT ,"ECHO
    CHAMBER"
20  LET Z =
    INT(8*RND)+1
30  FOR G = 1 TO Z + 5
35  CLS
40  LET D =
    INT(4*RND)+1
50  IF D = 1 THEN LET
    A$ = "KIDDO"
60  IF D = 2 THEN LET
    A$ = "SMART ONE"
70  IF D = 3 THEN LET
    A$ = "GENIUS"
80  IF D = 4 THEN LET
    A$ = "COMPUTER
    FREAK"
90  LET K = RND
100 PRINT
110 PRINT
120 PRINT "TRY NO. ";G;
    " OUT OF ";Z + 5
130 PRINT
140 PRINT "THE NUMBER
    YOU HAVE TO"
150 PRINT "REMEMBER, ";
    A$
160 PRINT ,,"IS ";K
170 PRINT
180 PRINT "WHEN YOU
    ARE SURE YOU"
190 PRINT "CAN DO THIS,
    PRESS N/L"
200 INPUT B$
210 IF B$ < > ""THEN
    STOP
220 GOSUB 290
230 PRINT "OK, ";A$;
    ", WHAT WAS"
240 PRINT ,"THE
    NUMBER?"
250 INPUT H
260 IF H = K THEN GOTO
    320
270 PRINT "YOU BLEW
    IT ";A$
280 STOP
290 CLS
300 PAUSE 100*G
310 RETURN
320 CLS
330 PRINT "YAY,";A$;",
    YOU GOT IT","RIGHT"
340 PRINT "WHEN READY
    FOR"
350 PRINT "NEXT ONE,
    PRESS N/L"
360 INPUT C$
```

```
37Ø  IF C$ < > "" THEN
     STOP
39Ø  IF G = Z + 5 THEN
     GOTO 44Ø
4ØØ  NEXT G
41Ø  CLS
44Ø  PRINT 'YOU SURE
     HAVE A
     PRODIGIOUS"
45Ø  PRINT ,"MEMORY,";A$
46Ø  STOP
```

In this program, which puts a decimal number in the range from nought to one on the screen, and then asks you to remember it for a time period which gets longer with each new number, note that lines 4Ø to 8Ø control what A$ will be in each run through the master FOR/NEXT loop. Line 3ØØ, as you can easily see, determines how long the delay loop will be. By the last few shots in a round, the delay seems interminable. Have a look at the listing for line 33Ø, noting the comma between the words IT and RIGHT. If the comma was not here, the word RIGHT would scroll around, half on one line and half on the other, when D equals 4. Can you see why?



Moving Graphics

Because the TV screen is not memory mapped, genuine moving graphics are unfortunately out of the question for the ZX81. But you can produce a sort of animated picture as this next program shows. Input it as listed, then try and work out a version that has two people running in a race, with "moving legs".

```
1Ø  LET N = 9
2Ø  DIM A(N)
3Ø  FOR Z = 1 TO N
4Ø  LET A(Z) =
    A(Z)+2*RND
5Ø  PRINT AT 2*Z,A(Z);"...
    graphic R";CHR$
    (Z+156);"graphic 6"
6Ø  IF A(Z) > =24 THEN
    GOTO 9Ø
7Ø  NEXT Z
8Ø  GOTO 3Ø
9Ø  PRINT AT Ø,11;"CAR
    ";CHR$
    (Z+156);"WINS"
```

In this program the "movement" comes from the line 5Ø which prints three dots before the "car" is printed. If you wanted a person with waving arms, a random number could determine which "arm subroutine" was printed. The two people running a race program could be written by combining the idea used in GRAND PRIX for moving the cars across, with a sub-subroutine to determine the position of the figures' legs.

If you have more than 1K memory, you could write a GRAND PRIX which includes a number of hazards (such as SMASH INTO SIDE WALL, OUT OF PETROL, or GEAR BOX BLOWS UP) which could either remove a car completely from the race, or simply take it out for a while, and then making sure it had to start again. Another variation on the above ideas is to let the cars, people or whatever move from left to right on the screen, and then move back. It should not be too hard for you to work out how this can be done.

In this next game, a fly (heavily disguised as an 'X') has to land on a sugar cube (graphic A). You press the keys 5,6,7 or 8 (ie the keys with the cursor arrows on them) to move the fly. Notice the interesting way in which the ZX81 knows how to move the fly (lines 1ØØ to 13Ø). Incidently this program won't run in FAST unless you add an extra line at 65 saying PAUSE 2Ø. This program gives a pretty good imitation of moving graphics.

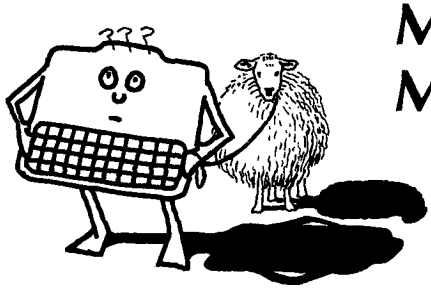
```
1Ø  LET Y=INT
    (28*RND)+1
2Ø  LET X=INT
    (18*RND)+1
3Ø  LET T=INT
    (28*RND)+1
4Ø  LET S=INT
    (18*RND)+1
5Ø  PRINT AT T,S;"X"
6Ø  PRINT AT Y,X;"graphic
    A"
7Ø  IF S=X AND T=Y-1
    THEN STOP
8Ø  LET U=S
9Ø  LET V=T
1ØØ IF INKEY$="5" AND
    S>=2 THEN LET
    S=S-1
11Ø IF INKEY$="6" AND
    T<=19 THEN LET
    T=T+1
```

```
12Ø IF INKEY$="7" AND
     T>=2 THEN LET
     T=T-1
13Ø IF INKEY$="8" AND
     S<=29 THEN LET
     S=S+1
14Ø PRINT AT V,U;"space"
15Ø GOTO 5Ø
```

Let the Games Begin

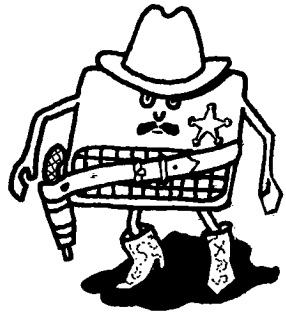
Now follows a series of games, all of which will fit within the 1K supplied with your machine. The games use ideas introduced in earlier sections of the book, and are here mainly for playing. You will probably learn something by studying the listings, but the main "teaching" is now behind you.

Did He Who Made the Lamb Make the UFO?



This game, which produces a remarkably attractive display, uses a subroutine to "draw" a flying saucer before each shot. Each saucer is a little different from the one that precedes it. The subroutine to generate the saucer is also used in the program LASER ROULETTE.

```
10  LET S = 0
20  LET T = S
30  CLS
40  PRINT "seven space
    graphic 3 graphic Q
    graphic W graphic 4"
50  GOSUB 500
60  PRINT "nine graphic T
    nine graphic Y","nine
    graphic Y nine graphic
    T"
70  GOSUB 500
80  PRINT "four space
    SLAUGHTER*"
90  PRINT
100 PRINT T;" OUT OF ";S
110 PRINT
120 LET S = S + 1
130 LET J = INT
    (1E5*RND)
140 PRINT "FOR SHIP
    ";S,"FIRE TO VECTOR
    ";J
150 POKE 16436,0
160 POKE 16437,0
170 INPUT M
180 IF M = J AND PEEK
    16436 +256*PEEK
    16437 > 65000 10*S
    THEN LET T = T + 1
190 IF RND < .95 THEN
    GOTO 30
200 PRINT "YOU
    SLAUGHTERED ";T;"
    SHIPS"
210 STOP
500 LET A = INT(11*RND)
    +128*INT(2*RND)
510 PRINT "three space";
520 FOR M = 1 TO 12
530 PRINT CHR$ A;
540 NEXT M
550 PRINT
560 RETURN
```



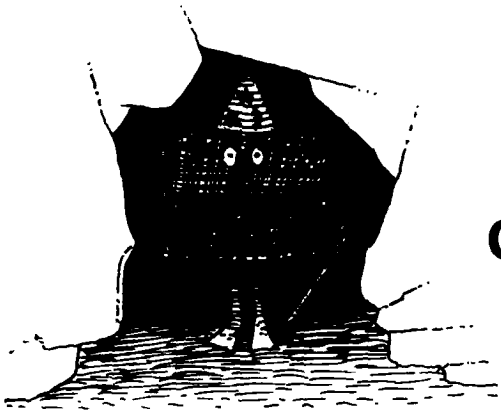
High Noon at the Old Intergalactic

In this game, you have to outdraw a nimble-fingered alien. The screen goes blank for a random length of time, then comes back, showing the alien (he moves across during the game). You must hit NEWLINE as fast as you can. If you do so quickly enough the alien dies, falling apart as splendidly as did the CHOPPER in an earlier game. Once you've mastered this game, change the 65500 in line 130 to 65520, and you'll find you have to learn to beat the alien all over again.

```
10  PRINT "YOU HAVE 6
    SHOTS keyword TO
    OUTDRAW THE ALIEN
    BEFORE IT DESTROYS
    YOUR three spaces
    BASE"
20  LET T = 1
30  LET M = 3*T+6
40  PRINT AT 17,M;"/";TAB
    M;"shift H";TAB
    M;"inverse space";TAB
    M;"graphic T graphic
    D"
50  PAUSE 300*RND
60  IF INKEY$ <> " "
    THEN GOTO 40
70  PRINT "SHOT ";T
80  POKE 16436,0
90  POKE 16437,0
100 INPUT H$
110 LET T = T + 1
120 CLS
130 IF PEEK 16436 +
    256*PEEK 16437 >
    65500 THEN GOTO 190
140 IF T <> 6 THEN
    GOTO 170
150 SAVE "X"
160 GOTO 200
170 PRINT "MISSED"
180 GOTO 30
190 PRINT "YOU HIT THE
    ALIEN keyword AND
    SAVED THE BASE"
200 PRINT AT 17,M;"two
    space *";TAB M;"/
    three space *";TAB
    M;"graphic 7 graphic
    6";TAB M;"two space
    graphic 4 graphic 2 two
    space graphic 1"
210 IF T < 6 THEN GOTO
    30
```

Line 6Ø is there to stop the player from cheating by pressing newline too early. A PAUSE can be terminated by pressing any key, so if line 6Ø were not there a player could just quickly press newline twice — once to end the pause and once to INPUT in line 7Ø, but with line 6Ø in place any attempt to do this will just restart the PAUSE all over again.

A novel feature of this game is line 15Ø, which puts the computer into the SAVE mode if you lose. A far more frustrating punishment for a loss is to have to computer erase the program. Can you work out what line 39 would have to be to make the program disappear completely?



Cave Master

You will recall that, earlier in this book, a program under the imaginative name of TIME WARP was listed. The next program uses the same basic program to produce something a little more down to earth. All of the programs in this book can (and should) be developed by you in whatever direction you prefer. Only by doing this will you develop your own programming skills. Anyway, here is one way TIME WARP can be warped.

```
1    LET P = 1
2    LET Q = P + P
1Ø   LET R = 31Ø
2Ø   PRINT "CAVE
     MASTER"
7Ø   LET G = Q*Q*Q
8Ø   LET H = G*Q
9Ø   GOSUB R
1ØØ  LET L$ = "CRAZED
     WIZARD"
11Ø  IF RND < P/Q THEN
     LET L$ = "WICKED
     WITCH"
```

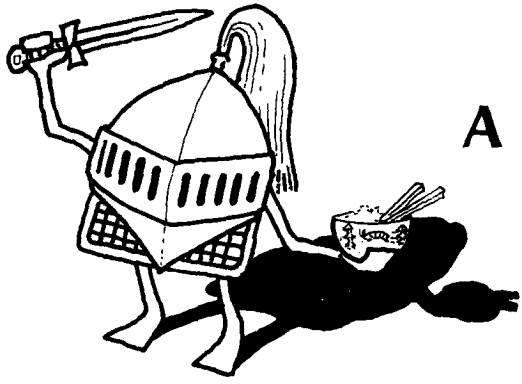
```
12Ø IF G > = P THEN
     GOTO 15Ø
13Ø PRINT "DEATH
     COMES TO US ALL"
14Ø STOP
15Ø PRINT CHR$ (G + 128);
     "AURA TONE";G,,
     "LEVEL OF MAGIC—"
     H,"HORRORS ";L$;
     "AHEAD"
16Ø LET H = H − P
17Ø GOSUB R
18Ø IF H > = P THEN
     GOTO 21Ø
19Ø PRINT "YOU TOOK
     TOO LONG"
2ØØ STOP
21Ø IF RND < P/Q THEN
     GOTO 25Ø
22Ø LET G = G − P
23Ø PRINT "THE
     ";L$;"ZONKED YOU"
24Ø GOTO 9Ø
25Ø LET G = G − P
26Ø PRINT "YOU ZAPPED
     THE ";L$,"YOU ARE
     CLOSER TO THE
     FOOLS GOLD"
27Ø GOSUB R
28Ø IF RND < .8 THEN
     GOTO 9Ø
29Ø PRINT "YOU DID IT
     WITH ";H;" MAGIC
     SPELLS two space
     LEFT"
3ØØ STOP
31Ø PAUSE R/Q
32Ø CLS
33Ø RETURN
```

Turning the Tables

There are many, many programs in which the computer thinks of a number, and the human player has to guess it. There are two of them in this book. This next program, written by Trevor Sharples, turns the tables.

```
2   PRINT "MYSTIC"
4   PRINT
6   PRINT "THINK OF A
    NO BETWEEN 1 AND
    1ØØ AND I WILL GUESS
    IT"
8   PRINT
1Ø  PRINT "PRESS N/L TO
    PLAY"
12  INPUT A$
14  GOSUB 86
16  LET T = Ø
18  LET Z = 1ØØ
2Ø  LET Y = 1
22  LET X =
    INT(5Ø*RND+RND)+1
24  PRINT "I GUESS ";X
26  PRINT
3Ø  PRINT"RIGHT (R) OR
    WRONG (W)?"
32  LET T = T + 1
34  INPUT A$
36  IF A$ = "R" THEN
    GOTO 66
38  GOSUB 86
4Ø  PRINT
44  PRINT "HIGHER (H)
    OR LOWER (L)?"
46  INPUT A$
48  GOSUB 86
5Ø  IF A$ = "L" THEN
    GOTO 62
52  LET Y = X
54  LET X = Y + INT
    ((Z-Y-2)*(RND+
    RND)/2)+1
6Ø  GOTO 24
62  LET Z = X
64  GOTO 54
66  GOSUB 86
68  PRINT "I GOT IT IN
    JUST ";T;" TRIES"
72  PRINT
74  PRINT "ANOTHER
    GAME, BUD?"
76  INPUT A$
78  GOSUB 86
8Ø  IF A$ = "YES" THEN
    GOTO 4
```

```
82  PRINT "BYE BYE THEN
    "
84  GOTO 82
86  PAUSE 200*RND
88  CLS
90  RETURN
```



A Ching in his Armour

This is a pretty trivial program, but if you have a copy of the "I CHING" it can be most entertaining. As you can see, a lot of the work the ZX81 does in this program is non-essential. A random number generator could do the job just as well. But it is better that the operator do more than just press NEWLINE, if only to feel that he or she has influence on what is going on.

```
1   PRINT AT 4,11;"I
    CHING"; AT 7,4;"WHAT
    IS YOUR FIRST
    NAME?"
9   INPUT A$
10  PRINT AT 10,8;"AND
    LAST?"
13  INPUT B$
16  CLS
17  PRINT "YOUR
    HEXAGRAM FOR
    TODAY IS:"
18  PRINT
20  FOR D = 1 TO 6
21  LET C = (CODE
    A$*CODE B$)*RND >
    CODE A$*CODE B$/2
22  IF C THEN PRINT
    "(graphic 6 inverse
    space graphic 6 inverse
    space space"
23  IF NOT C THEN PRINT
    "inverse space graphic 6
    inverse space graphic 6
    inverse space graphic 6"
```

```
24  PRINT
25  NEXT D
26  PRINT
27  PRINT
28  PRINT "SEE THE I
    CHING FOR
    AN","INTERPRET-
    ATION"
3Ø  PRINT
33  PRINT "YOUR LUCKY
    NUMBER IS ";INT
    (CODE A$*CODE
    B$*RND)+1
```

Look at line 21. LET C equal something greater than something else. It doesn't really seem to make sense does it? In actual fact this is a very very useful trick to know when you want to save space. Conditions like greater than, equals, or less than, will always be either TRUE or FALSE. The number one means TRUE, and the number zero means FALSE. Line 21 says LET C equal either 1 (TRUE) or Ø (FALSE) depending on the condition. Lines 21 to 23 could have been written as:

```
21
22  IF (CODE A$*CODE
    B$)*RND > CODE
    A$*CODE B$/2 THEN
    PRINT "etc"
23  IF NOT (CODE
    A$*CODE B$)*RND >
    CODE A$*CODE B$/2
    THEN PRINT "etc"
```

as you can see, by making use of the variable C we have saved a fair amount of space, and in fact without it the program won't fit in 1K.

Nine Lives

In HANGCAT one player inputs a word. Then the second player tries to guess the word. If the second player is wrong he loses a life (hence the title). If the guess is right, the program prints out the correct letter in its correct position in the word.

```
10  PRINT "HANGGAT"
20  PRINT "PLAYER 1 TYPE
    IN A WORD"
50  INPUT A$
60  LET G$ = " "
70  FOR Z = 1 TO LEN A$
80  LET G$ = G$ + "-"
90  NEXT Z
95  LET U$ = " "
100 LET T = 9
110 CLS
120 PRINT "YOU HAVE
    ";T;" LIVES"
124 PRINT
125 IF U$ < > " " THEN
    PRINT "LETTERS USED
    =";U$
126 PRINT
130 PRINT "WHAT IS
    YOUR GUESS?"
140 INPUT X$
144 CLS
145 LET U$ = U$ + X$
150 LET F = 0
160 FOR Z = 1 TO LEN A$
170 IF X$ < > A$(Z) THEN
    GOTO 200
180 LET F = 1
190 LET G$(Z) = X$
200 NEXT Z
210 PRINT X$
215 PRINT
220 PRINT G$
230 IF F = 0 THEN PRINT
    "WRONG"
240 IF F = 1 THEN PRINT
    "RIGHT"
245 PRINT
250 IF F = 0 THEN LET T
    = T - 1
260 IF T > 0 AND G$ < >
    A$ THEN GOTO 120
270 IF G$ < > A$ THEN
    PRINT "YOURE DEAD"
280 IF G$ = A$ THEN
    PRINT "YAY CAT"
290 PRINT "THE CORRECT
    WORD WAS",A$
300 PRINT
310 PRINT "ANOTHER
    CAT?"
320 INPUT A$
330 CLS
340 IF A$(1) "Y" THEN RUN
```

Let the longer games begin

After a while, you'll discover the 1K supplied is a bit cramping, and you'll long to stretch your wings and add a byte or two. When you do this, you'll discover that many of the good habits you've learned when you were restricted to 1K can desert you. It is very easy to set up a long and sloppy set of IF/THENs which could easily be replaced by an IF/THEN instruction to GOSUB. When you have memory to spare, it often seems too much trouble to bother cleaning up your programs. Unused subroutines clutter up the bottom ends of your programs. GOTO statements cover a multitude of situations which arose because you did not give sufficient thought to the maximum line number you would need.

If you are going to take up flow-charting, now is the time to begin. If you can't be bothered with pretty triangles and things, at least discipline yourself to setting out — on paper — whate your program is supposed to do, with arrows linking FOR/NEXT loops, and lines leading to the first lines of subroutines. If you can be bothered, it is worth writing out a full listing for a program once you get it working. Examine it in detail, and you're sure to find more elegant ways of achieving the same ends. Be particularly critical of each and every GOTO command which is non-conditional.

All the programs given so far in this book can act as starter ideas for much bigger and better programs when you get extra memory. For example, the LUNAR LANDING program is a grown up version of the 1K LUNAR LANDER and LABYRINTH is a much-expanded version of TIMEWARP and CAVE-MASTER.

The best thing you can add to a program with added memory is the element of surprise. If you can include situations which do not occur every time a game is played, you'll ensure the game will remain interesting for a much longer time than would be the case if every situation is triggered every time a game is run.

As you know, many of the games in this book run far too fast to be good games without the use of a "delay subroutine". However, as you get into longer games (and I mean ones much longer than those listed in this section) you'll find that slow-running programs and response-times can be boring, especially if you have a graphical element in your program, which "moves" in some way from go to go.

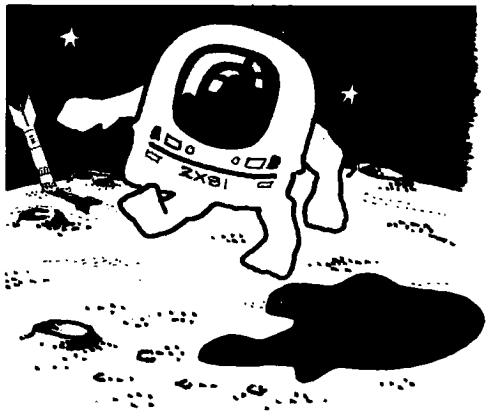
Variables which must be assigned at the start of a game can actually be listed right at the end of the program, ending with a

GOTO leading back to the start of the game program proper. You can have a line like "DO YOU WANT INSTRUCTIONS?" near the start of a program, and if the answer is "YES" the computer can GOTO the end of the program where the instructions are. LABYRINTH, if you wanted it to run more quickly, could have the instructions at the very end. However, when you add a 16K pack to your ZX81, you'll have very, very long programs, and you will not always want to wait while the computer searches a vast listing for the subroutine.

Another way of improving programs, and making them interesting to players for a longer time, is to use a feature you've seen in some programs, the "degree of difficulty". Make sure that this feature really does increase the difficulty of the game. Ensure that, even at the highest level of play, the final score (or successful landing, or obliterated aliens or whatever) is attainable.

You can add interest to games by awarding points, or scores, or ratings or whatever, that are genuinely related to the speed, skill or whatever the player demonstrated. A further twist is to award a "rank" (like "star fleet captain", "novice" or "incompetent fool") to the player, depending on how well he or she did. Points and ranks ensure that a player remains interested in a game for a longer time, as the player will try to beat his or her previous best score or ranking.

Keeping these features in mind, have a look at the next two games, input them and run them, and then try to improve on them.

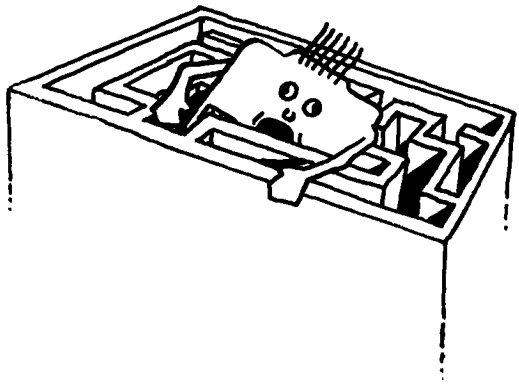


Lunar Landing

```
10  LET M = 0
20  LET T = 0
30  LET S = 0
40  LET H = 5000
50  PRINT "LUNAR
    LANDING"
60  LET F = 2000/RND
70  LET Q = -17
80  LET B = 1
90  RAND
100 PRINT
110 PRINT
120 GOTO 300
130 PRINT "(graphic 5) +
    IS TOWARDS LUNA
    (two graphic 8)"
140 INPUT Z
150 IF Z < -50 OR Z > 50
    THEN GOTO 410
160 PRINT "FOR HOW
    MANY SECONDS?"
170 INPUT E
180 CLS
190 LET T = T + E
200 LET S = S + 10 +
    3*E*(Z + 1)/B
210 LET F = F -
    3*E*ABS(3*RND*Z)
220 IF F < 500 THEN PRINT
    "three graphic 5 FUEL
    LOW three graphic 8"
230 LET H = H - E*S
240 IF H < 20 AND H > -10
    AND S < 12 THEN
    GOTO 460
250 IF H < -10 THEN
    GOTO 430
260 IF F < 0 THEN GOTO
    430
270 LET X =
    INT(10*RND)+1
280 IF X = 5 AND M <> 2
    THEN GOSUB 790
290 PRINT " "
300 PRINT CHR$(128 + M);
    " HEIGHT ABOVE
    SURFACE: ";H
310 IF Q <> -17 THEN LET
    Q = Q - 16*RND
320 IF Q < 0 AND Q > -17
    THEN GOTO 430
330 IF NOT Q = -17
    THEN PRINT "six
    graphic + OXYGEN
    LEFT";Q
340 PRINT CHR$(128 + M);
    " VELOCITY: ";S
```

```
350  IF NOT B = 1 THEN
     PRINT CHR$(128 +
     M);"three graphic 5
     WARNING — THRUST
     ERRATIC"
360  PRINT CHR$(128 + M);
     " FUEL LEFT: ";F
370  PRINT CHR$(128 + M);
     " FLIGHT TIME: ";T
380  GOSUB 510
390  PRINT
400  GOSUB 550
410  PRINT "THRUST (-50
     TO +50)?"
420  GOTO 130
430  CLS
440  PRINT "CRASH. HIT
     SURFACE AT ";ABS S
450  GOTO 440
460  CLS
470  PRINT "SUCCESSFUL
     LANDING"
480  PRINT
490  PRINT "FINAL
     VELOCITY: ";ABS S;
500  GOTO 490
510  FOR A = 1 TO 64
520  PRINT CHR$ (128 +
     M);
530  NEXT A
540  RETURN
550  PRINT
560  PRINT "(space graphic
     3 graphic 4)"
570  FOR Y = 1 TO 2
580  PRINT "(space)";CHR$
     (128 + M);CHR$(128 +
     M)
590  NEXT Y
600  PRINT "graphic A two
     graphic 5 graphicA"
610  PRINT "graphic A two
     graphic 5 graphic A"
620  PRINT
630  FOR G = 1 TO 2
640  LET K = INT(10*RND)
650  IF K = 0 THEN LET W$
     = ";;;;"
660  IF K = 1 THEN LET W$
     = ",,,,"
670  IF K = 2 THEN LET W$
     = ": :"
680  IF K = 3 THEN LET W$
     = " : :"
690  IF K = 4 THEN LET W$
     = " ::"
700  IF K = 5 THEN LET W$
     = "::"
710  IF K = 6 THEN LET W$
     = "...."
720  IF K = 7 THEN LET W$
     = "::::"
730  IF K = 8 THEN LET W$
     = "::"
740  IF K = 9 THEN LET W$
     = " .."
750  PRINT W$
760  NEXT G
770  PRINT
780  RETURN
790  CLS
800  LET M = M + 1
810  RAND
820  FOR V = 1 TO 7
830  PRINT
840  NEXT V
850  LET U =
     INT(50000*RND)
860  FOR V = 1 TO 6
870  PRINT "HOUSTON,
     WE HAVE A
     PROBLEM..."
880  PRINT CHR$ (127+
     RND*(11)+1);"
     DANGER ";CHR$
     (127+RND*(11)+1)
890  NEXT V
900  PRINT
```

```
91Ø  PRINT
     "MALFUNCTION. USE
     COMPUTER"
92Ø  PRINT "ACCESS
     CODE ";U;" FOR
     DETAILS"
93Ø  INPUT V
94Ø  CLS
95Ø  IF U < > V THEN GOTO
     43Ø
96Ø  LET V = INT(2*RND)
97Ø  IF V = Ø THEN GOSUB
     1Ø3Ø
98Ø  IF V = 1 THEN
     GOSUB 1Ø7Ø
99Ø  PRINT "N/L TO
     RETURN TO FLIGHT"
1ØØØ INPUT V$
1Ø1Ø CLS
1Ø2Ø RETURN
1Ø3Ø LET Q = 1ØØ +
     19*RND
1Ø4Ø PRINT "OXYGEN
     METER UNRELIABLE"
1Ø5Ø PRINT "(graphic A 23
     times)"
1Ø6Ø RETURN
1Ø7Ø LET B = B +
     INT(3*RND)+1
1Ø8Ø PRINT "THRUST
     CONTROL ERRATIC"
1Ø9Ø PRINT "twenty two
     graphic 6"
11ØØ RETURN
```



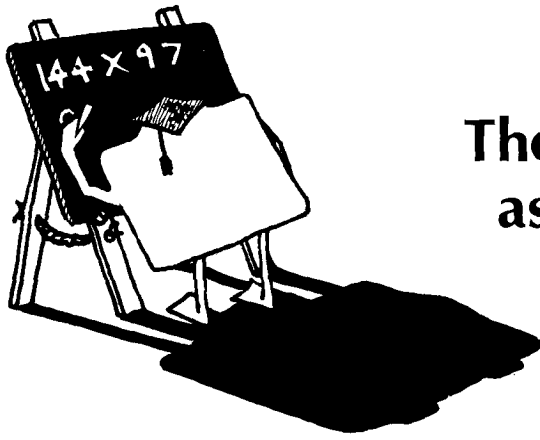
Labyrinth

```
1Ø   GOSUB 1ØØØ
4Ø   LET X = Ø
5Ø   LET S = 3Ø
6Ø   LET W = 1
7Ø   PRINT "inverse space
     YOU ARE AT THE
     START OF A"
8Ø   PRINT "inverse space
     LABYRINTH OF MANY
     TWISTING,"
9Ø   PRINT "inverse space
     TURNING TUNNELS.
     YOU HAVE"
1ØØ  PRINT " inverse space
     A SACK HOLDING 3Ø
     PIECES OF"
11Ø  PRINT "inverse space
     SILVER. YOU MUST
     GET TO"
```

```
12Ø  PRINT "inverse space
     THE END OF THE
     LABYRINTH WITH AT"
13Ø  PRINT "inverse space
     LEAST 2Ø TO PAY THE
     MINOTAUR"
15Ø  PRINT AT 2Ø,18;"PRESS
     NEWLINE"
16Ø  INPUT A$
17Ø  RAND
18Ø  IF A$ < > " " THEN
     STOP
19Ø  GOSUB 1ØØØ
2ØØ  IF W <1 THEN LET W
     = 1
21Ø  PRINT "inverse space
     THIS IS MAZE/
     TUNNEL"
23Ø  IF W = 1Ø THEN
     GOTO 1Ø7Ø
24Ø  PRINT "inverse space
     NUMBER ";W;
     " OF THE LABYRINTH"
26Ø  PRINT "inverse space
     (1Ø IS THE END)"
27Ø  PRINT
28Ø  LET X = X + 1
29Ø  PRINT "inverse space
     THIS IS CHALLENGE
     NUMBER ";X
3ØØ  IF S <1 THEN LET S =
     5
32Ø  PRINT "inverse space
     YOU HAVE ";S;
     " SILVER PIECES"
33Ø  LET K =
     INT(5*RND)+1
34Ø  PRINT
35Ø  PRINT "inverse space
     FACING YOU NOW
     ARE ";K;" DOORS"
36Ø  PRINT "inverse space
     WHICH ONE WILL
     YOU TRY?"
37Ø  INPUT A
38Ø  GOSUB 1ØØØ
39Ø  IF RND < .1 THEN
     GOSUB 69Ø
4ØØ  IF A < > K THEN GOSUB
     42Ø
41Ø  IF A = K THEN
     GOSUB 69Ø
42Ø  LET K = INT(4*RND)
43Ø  IF K = Ø THEN LET E$
     = "RODENTING RAT"
44Ø  IF K = 1 THEN LET E$
     = "WART-FACED
     WOGGLE"
45Ø  IF K = 2 THEN LET E$
     = "ELLIPSOID
     OCTOPUS"
46Ø  IF K = 3 THEN LET E$
     = "WACKED-OUT
     WIZARD"
47Ø  PRINT "inverse space
     FOOL, YOUVE
     WALKED IN ON"
48Ø  LET E = INT(4*RND)
49Ø  IF E = Ø THEN LET F$
     = "FLAMING BRAND"
5ØØ  IF E = 1 THEN LET F$
     = "SHINING SWORD"
51Ø  IF E = 2 THEN LET F$
     = "POISONED
     NEEDLE"
52Ø  IF E = 3 THEN LET F$
     = "GOSUB-MACHINE-
     GUN"
53Ø  PRINT "inverse space
     ";E$;" ARMED"
54Ø  PRINT "inverse space
     WITH A ";F$
55Ø  PRINT
56Ø  PRINT "inverse space
     WHICH WEAPON DO
     YOU CHOOSE?"
```

```
58Ø  PRINT "inverse space A
     FLOATING POINT
     ROM (1),"
6ØØ  PRINT "inverse space A
     FOR/NEXT LOOP (2),"
62Ø  PRINT "inverse space
     OR A POKED
     ADDRESS (3)?"
63Ø  INPUT B
64Ø  LET C =
     INT(3*RND)+1
65Ø  GOSUB 1ØØØ
66Ø  IF B = C THEN
     GOSUB 117Ø
67Ø  IF B <> C THEN GOSUB
     124Ø
68Ø  GOTO 14Ø
69Ø  LET K = INT(4*RND)
7ØØ  IF K = Ø THEN GOSUB
     76Ø
71Ø  IF K = 1 THEN
     GOSUB 81Ø
72Ø  IF K = 2 THEN
     GOSUB 85Ø
73Ø  IF K = 3 THEN
     GOSUB 9ØØ
74Ø  GOTO 15Ø
76Ø  PRINT "inverse space
     YOUVE FALLEN
     THROUGH"
77Ø  PRINT " inverse space
     A TRAPDOOR...."
78Ø  LET W = W - 1
79Ø  LET S = S -
     INT(2*RND)-1
8ØØ  RETURN
81Ø  PRINT "inverse space A
     WALL OF FLAME
     ENGULFS YOU"
82Ø  LET W = W - 1
83Ø  LET S = S -
     INT(2*RND)-1
84Ø  RETURN
85Ø  PRINT "inverse space
     THE LOVELY PRINCESS
     SEMOLINA"
86Ø  PRINT "inverse space
     SOOTHES YOUR
     FEVERED BROW"
87Ø  LET S = S +
     INT(5*RND)+1
88Ø  LET W = W +
     INT(3*RND)+1
89Ø  RETURN
9ØØ  PRINT "inverse space
     JOY OH JOY. A
     HOARD OF"
91Ø  PRINT "inverse space
     SILVER. CHOOSE UP
     TO 5 PIECES"
92Ø  PRINT "inverse space
     BUT BE WARNED. THE
     MORE"
93Ø  PRINT "inverse space
     YOU TAKE, THE MORE
     IT WILL"
94Ø  PRINT "inverse space
     COST YOU. HOW
     MANY?"
95Ø  INPUT D
96Ø  LET S = S + D
97Ø  LET W = W -
     INT(D/2)
98Ø  RETURN
1ØØØ CLS
1Ø1Ø PRINT "ten inverse
     space graphic E nine
     graphic 7 graphic R
     twenty-one inverse
     space graphic 5
     LABYRINTH graphic 8
     twenty-one inverse
     space graphic W nine
     graphic 6 graphic Q
     eleven inverse space"
1Ø2Ø FOR Z = 1 TO 19
```

```
1030 PRINT "thirty two
     inverse space"
1040 NEXT Z
1045 PRINT AT 3,0
1050 RETURN
1070 PRINT "inverse space
     YOU ARE AT THE
     END"
1080 PRINT "inverse space
     DO YOU HAVE
     ENOUGH SILVER?"
1090 PRINT "inverse space
     PRESS NEWLINE TO
     FIND OUT"
1100 INPUT C$
1110 IF S < 20 THEN PRINT
     "inverse space THE
     MINOTAUR HAS
     EATEN YOU"
1120 IF S < 20 THEN GOTO
     1110
1130 IF S > 19 THEN PRINT
     "inverse space YES,
     YOU HAVE ";S; "
     SILVER"
1140 IF S > 19 THEN PRINT
     "inverse space PIECES.
     YOU HAVE WON",
1150 IF S > 19 THEN GOTO
     1130
1160 STOP
1170 PRINT "inverse space
     YOU BEAT THE ";E$
1180 LET S = S +
     INT(3*RND)+1
1190 PRINT "inverse space
     AND HAVE ";S; "
     SILVER PIECES"
1200 LET W = W +
     INT(4*RND)+1
1210 PRINT
1220 PRINT "inverse space
     YOU ARE
     APPROACHING
     SECTOR ";W
1230 RETURN
1240 PRINT "inverse space
     THE ";E$; " BEAT
     YOU,"
1250 LET S = S -
     INT(4*RND)-1
1260 PRINT "inverse space
     LEFT YOU WITH ";S;"
     SILVER"
1270 LET W = W - 1
1280 IF W <1 THEN LET W
     = 1
1290 PRINT "inverse space
     AND SENT YOU BACK
     TO ";W
1300 RETURN
```



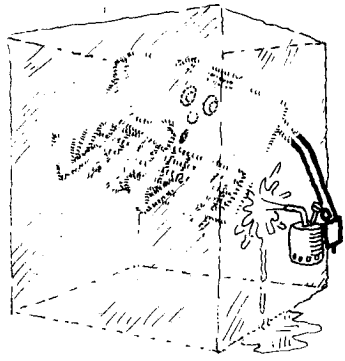
The ZX81 as Teacher

The ZX81 is an effective game-player. Writing and running programs on the computer enhances programming skills. However, the ZX81 can also be used in a direct role as a teaching aid. Its main use is in the field of quizzes. While the ZX81 can only select from a list of non-numerical questions, the computer can easily be programmed to create its own numerical questions.

Another use in teaching is for the ZX81 to create lists and tables. We will look at this use first.

```
1    PRINT AT
     2,8;"MULTIPLICATION
     TABLES"
2    PRINT AT 6,2;"WHICH
     TIMES TABLE WOULD
     YOU","LIKE ME TO
     PRINT?"
3    INPUT A
10   CLS
11   FOR J = 1 TO 12
12   PRINT ,,J;" X ";A;" = "
     ;A*J
13   NEXT J
14   PRINT AT 14,0;"DO
     YOU WANT ANOTHER
     GO?"
17   INPUT A$
18   CLS
19   IF NOT A$ = "NO"
     THEN GOTO 2
21   PRINT AT 21,8;, "OK.
     BYE FOR NOW"
```

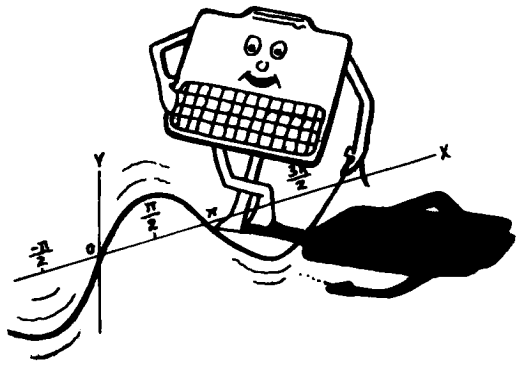
A far more useful table is produced by the following program (and the display is a little more imaginative). Again, this program is given to suggest ideas for your own programs.



A Degree of Conversion

```
1Ø  RANDOMISE
2Ø  PRINT "(5 spaces)A
    DEGREE OF
    CONVERSION"
3Ø  LET K = 1Ø*RND
4Ø  FOR S = 1 TO 32
5Ø  PRINT CHR$(K + 128);
6Ø  NEXT S
7Ø  PRINT AT 5,Ø;"WHAT
    IS THE LOWEST TEMP.
    (F) YOU"
12Ø PRINT "WANT TO
    CONVERT?"
13Ø INPUT A
14Ø PRINT ,"AND
    HIGHEST?"
15Ø INPUT B
16Ø IF B < A THEN LET E =
    A
17Ø IF A < B THEN LET F =
    A
18Ø IF B < A THEN LET F =
    B
19Ø IF A < B THEN LET E =
    B
2ØØ PRINT "IN WHAT
    DEGREE STEPS?"
21Ø INPUT Z
22Ø CLS
23Ø PRINT TAB
    8;"F","C";TAB 24;"K"
25Ø LET H = 1Ø*RND
26Ø FOR U = 1 TO 21
27Ø PRINT CHR$(H + 128);
28Ø NEXT U
29Ø PRINT
3ØØ FOR D = F TO E + Z
    STEP Z
31Ø LET C = 5*(D-32)/9
32Ø LET K = C + 273
33Ø PRINT TAB 8;D,C;TAB
    24, K
35Ø NEXT D
36Ø STOP
```

In all programs, and especially in educational ones, you have to try and anticipate any mistakes (deliberate or otherwise) users will make when running a program. The lines 16Ø to 19Ø cover the possibility that a user will input the higher temperature first, rather than the lower one which was requested.



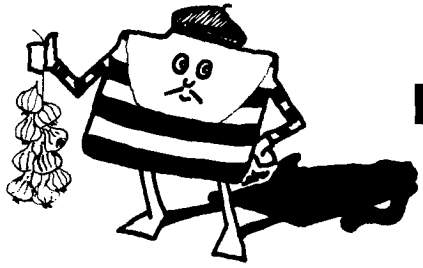
Sines

The following table, which prints out numbers and their sines, uses a counter to stop execution when the memory is full. You can use this idea if the program output is sequential and is likely to cause the program to crash by demanding more screen space than you have.

```
10  PRINT ,"SINES"
20  PRINT
30  PRINT ,"LOWEST
    NUMBER ";
40  INPUT A
50  PRINT A
60  PRINT ,"HIGHEST
    NUMBER ";
70  INPUT B
80  PRINT B
90  PRINT
100 PRINT "FOR TABLE
    space ";
110 LET L = 0
120 PRINT "PRESS
    NEWLINE"
130 INPUT A$
140 CLS
150 FOR D = A TO B
160 LET L = L + 1
170 IF L >11 THEN GOTO
    240
180 LET R = D*PI/180
190 PRINT "SIN";D;"
    IS",SIN R
200 NEXT D
210 PRINT
220 PRINT "END OF
    TABLE"
230 STOP
240 PRINT
250 PRINT "PRESS
    NEWLINE"
260 INPUT A$
270 LET L = 1
280 CLS
290 GOTO 180
```

Notice how line 18Ø changed from degress (D) which humans are more used to using, to radians (R) which computers (such as the ZX81) use. The ZX81 will always calculate in radians, so if you want to use degrees you must always convert first by multiplying by PI/18Ø.

I'm sure you will now be able to work out an endless stream of numerical quizzes for yourself, your children or your class to tackle. Non-numerical quizzes are very useful, but they require much more work in programming. Whereas the ZX81 can create its own numerical questions, each non-numerical question must be specified, and each answer included in full in the program.



French Vocabulary

Here is a sample quiz which can be adapted for any subject.

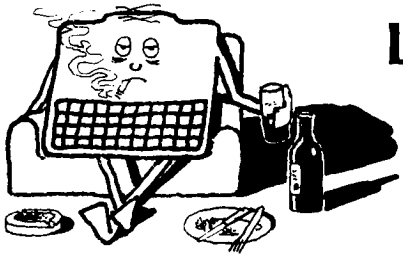
```
1    DIM E$ (1Ø,1Ø)
2    DIM F$ (1Ø,8)
3    DIM G$ (8)
1Ø   GOSUB 25Ø
2Ø   LET A$ = "GIVE THE
     FRENCH FOR"
3Ø   LET B$ = "CORRECT"
4Ø   LET C$ =
     "INCORRECT.
     ANSWER IS "
5Ø   PRINT "FRENCH
     VOCAB 1"
6Ø   LET A = Ø
7Ø   PRINT "HOW MANY
     QUESTIONS?"
8Ø   INPUT N
9Ø   FOR J = 1 TO N
1ØØ  CLS
11Ø  LET R =
     INT(1Ø*RND)+1
13Ø  PRINT A$;E$(R)
14Ø  INPUT G$
15Ø  IF G$ = F$(R) THEN
     LET A = A + 1
16Ø  IF G$ = F$(R) THEN
     PRINT B$
```

```
165 IF G$ < > F$(R) THEN
    PRINT C$;F$(R)
170 PRINT "PRESS
    NEWLINE"
175 INPUT H$
180 CLS
185 NEXT J
190 PRINT "OUT OF ";N;"
    QUESTIONS YOU"
200 PRINT "HAD ";A;"
    RIGHT,";A*100/N;"-
    PERCENT "
210 PRINT "ANOTHER
    GO?"
220 INPUT D$
230 IF D$ = "YES" THEN
    GOTO 60
240 STOP
250 LET E$(1) = "NEXT"
255 LET F$(1) =
    "PROCHAIN"
260 LET E$(2) = "YOUNG"
265 LET F$(2) = "JEUNE"
270 LET E$(3) = "HERE"
275 LET F$(3) = "ICI"
280 LET E$(4) = "BETTER"
285 LET F$(4) =
    "MEILLEUR"
290 LET E$(5) = "MORE"
295 LET F$(5) = "PLUS"
300 LET E$(6) = "INSIDE"
305 LET F$(6) = "DEDANS"
310 LET E$(7) = "HOT"
315 LET F$(7) = "CHAUD"
320 LET E$(8) = "FULL"
325 LET F$(8) = "PLEIN"
330 LET E$(9) = "SUGAR"
335 LET F$(9) = "LE
    SUCRE"
340 LET E$(10) =
    "EVERYWHERE"
345 LET F$(10) =
    "PARTOUT"
347 RETURN
```

Note that there is no mechanism in this program for preventing the same question being asked more than once in a run. The limit set on the random number in line 110 (that is, the number in brackets with RND) is equal to the number of different questions in the program. You can get six programs of this length onto one side of a C-12 cassette (just), so — with programs similar to the one listed above — a 120-word vocabulary can be tested with the programs stored on a single cassette. If your questions need more than a one word reply, you will find (of course) that you can fit far fewer questions into a single program before running out of RAM.

In this program, DIM E$(10,10) ensures that every element of E$ is ten characters long, so when, for example in line 250, the computer is told LET E$(1) "NEXT" it will actually be assigned "NEXT six spaces". Similarly F$ will always be eight characters long, so that F$(3) is "ICI five spaces", not "ICI". So we have to make sure that the string that the human player inputs is always eight characters long, since "ICI" is NOT equal to "ICI five spaces" and the correct answer would be counted as wrong! This is the purpose of line 3: DIM G$(8). Looks rather strange doesn't it — when dimensioning a string we've until now always

had at least two numbers in the brackets — one for the no of elements, and one for the length of each element (as in E$ and F$). The effect of DIM G$(8) is to set up a single string (ie not an array) called G$, but one which will ALWAYS be eight characters long. If in line 14Ø you input "ABC" G$ will be assigned "ABC five spaces". If you input "ABCDEFGHIJKLMN" G$ will be assigned "ABCDEFGH". Thus the comparison in lines 16Ø and 165 will be valid ones.



Life Expectancy

There was a light-hearted life expectancy program earlier in the book. The next program, based loosely on actuarial tables, is closer to a "serious" life expectancy program. However, it is simplified, and therefore to some degree rendered less accurate. It is not really an "educational" program (although it could be used within a classroom as a demonstration of the "real" use of computers) but this seemed the best place in the book to put it. This program will not fit within 1K.

```
1    LET C = Ø
2    PRINT "LIVES"
3    LET A = 71
4    PRINT ,"AGE (YRS)?"
5    INPUT B
6    GOSUB 54
7    PRINT ,"MARRIED?"
8    INPUT A$
9    GOSUB 54
1Ø   IF A$ = "YES" THEN
     LET A = 76
11   PRINT "HAVE YOU
     BEEN RICH"
12   PRINT "MOST OF
     YOUR LIFE?"
13   INPUT C$
14   GOSUB 54
15   IF C$ = "YES" THEN
     LET A = A - 3
```

```
16  PRINT "ARE YOU
    OVERWEIGHT?"
17  INPUT D$
18  GOSUB 54
19  IF D$ = "NO" OR B <
    4Ø THEN GOTO 24
2Ø  IF D$ = "YES" THEN
    PRINT, "BY HOW
    MANY POUNDS?"
21  INPUT P
22  LET C = C + P
23  GOSUB 54
24  PRINT "EXERCISE?
    NEVER. SOMETIMES.
    OFTEN"
25  INPUT E$
26  GOSUB 54
27  IF E$ = "SOMETIMES"
    THEN LET A = A + 3
28  IF E$ = "OFTEN"
    THEN LET A = A + 5
29  GOSUB 54
3Ø  PRINT "ARE YOU
    OFTEN TENSE?"
31  INPUT F$
32  GOSUB 54
33  IF F$ = "YES" THEN
    LET A = A - 3
34  IF F$ = "NO" THEN
    LET A = A + 3
35  PRINT "DRINK? LITTLE
    (Ø), MOD.(5)"
36  PRINT ,"HEAVY(1Ø)"
37  INPUT G
38  IF B > A THEN LET A
    = B + (B - A)/2
39  GOSUB 54
4Ø  PRINT "DO YOU
    SMOKE?"
41  INPUT H$
42  IF H$ = "YES" THEN
    LET A = A - 5
43  GOSUB 54
44  PRINT "OFTEN ILL?"
45  INPUT K$
46  IF K$ = "YES" THEN
    LET A = A - 3
47  IF K$ = "NO" THEN
    LET A = A + 3
48  GOSUB 54
49  PRINT "EST. AGE AT
    DEATH:"
5Ø  PRINT
51  PRINT ,"FEMALE— ";A
    + 7 - C/5 - G
52  PRINT, "MALE— ";A
    - C/5 - G
53  STOP
54  CLS
55  PRINT
56  PRINT
57  PRINT
58  RETURN
```

Useful Subroutines

Science of Cambridge claim the 1K RAM supplied with the standard ZX81 is equal to 4K of anybody else's RAM, because most commands and statements are stored in a single byte. Although it appears that this claim is a little ambitious, the RAM can be made to hold a pretty long program and the shortage of memory teaches ingenuity in programming. As you'll have discovered, Sinclair's dialect of BASIC has no facility for READ/DATA. In this section of the book we will look at some useful subroutines, some of which have been introduced in the programs. I thought it would be handy to have them all together in one place. Many of the subroutines were worked out by inventive members of the Users Club and have already been printed in the club magazine INTERFACE.

Clive Davies of Cheltenham devised the following subroutine to replace the missing READ/DATA function:

```
1Ø  LET N = VALZ$( TO 5)      2Ø  LET Z$ = Z$(6 TO )
```

This subroutine supplies five digits of data stored as Z$.

An efficient way of making the most of memory in some programs is to arrange for subroutine destinations to be specified by a multiple of, or a multiple plus an arithmetic manipulation of a randomly generated number. That sounds more complicated than it is in practice. Look at the following program:

```
1Ø  LET K =
    INT(4*RND)+1
2Ø  GOSUB 1Ø*K + 5Ø
3Ø  PRINT
4Ø  PRINT "THIS IS LINE
    4Ø"
5Ø  STOP
6Ø  PRINT "SUBROUTINE
    6Ø"
65  RETURN
7Ø  PRINT "SUBROUTINE
    7Ø"
75  RETURN
8Ø  PRINT "SUBROUTINE
    8Ø"
85  RETURN
9Ø  PRINT "SUBROUTINE
    9Ø"
95  RETURN
```

As you can see, line 2Ø simply manipulates the random number generated in line 1Ø to produce a destination for the subroutine jump. The manipulation can be a simple multiple (GOSUB 5*J), a multiple plus an addition (as in line 2Ø) or subtraction, or a conditional expression plus a manipulation of the random number (IF J > 1Ø THEN GOSUB 1Ø*J + N).

The following is a subroutine to renumber lines of a program in steps of 10. To alter the step size change line 9997, and alter the initial line number change line 9990. Lines 9993 and 9995, are necessary because the way the ZX81 stores its programs means there might be a character 118 (usually an end of line character) in the middle of a line of program, which has to be ignored. For example the line LET S = 123 contains 118 which doesn't show in the listing.

```
9989 KET X = 16508
9990 LET L = 10
9991 POKE X+1,INT (L/256)
9992 POKE X+2,L-256*INT
     (L/256)
9993 LET X = X + 4
9994 LET X = X + 1
9995 IF PEEK X = 126 THEN
     LET X = X + 6
9996 IF PEEK X <> 118
     THEN GOTO 9994
9997 LET L = L + 10
9998 IF 256*PEEK
     (X+1)+PEEK
     (X+2)=9989 THEN
     STOP
9999 GOTO 9991
```

Sometimes programs can be made to look neater by removing the numerical message at the bottom of the screen which indicates a STOP command, or end or program. This routine, devised by Toni Baker, will achieve this provided the routine is placed at very end of a program.

```
FAST
LET L = USR 681
```

Obviously you do not need the instruction FAST if you're already in FAST mode.

Joe Fitzpatrick, a users club member from Dublin suggests a subroutine which stops a program crashing when the screen is full. He included it in a program which displays each address and the character stored in it. His program essentially consists of printing the characters whose values are the variable A in the expression A = PEEK(B) where B is a number in the range 16424 to 17424. The subroutine to temporarily halt the program when the screen is full is:

```
30   IF PEEK (16442) < 3
     THEN GOSUB 100
.....
100  PRINT
110  PRINT "PRESS
     NEWLINE TO
     CONTINUE"
120  INPUT A$
130  CLS
140  IF A$ = "" THEN
     RETURN
150  STOP
```

The use of the quote marks in line 14Ø ensures that the ZX81 will RETURN only if just NEWLINE is pressed.

Pressing any other key before NEWLINE will cause the program to STOP.

Users' club member Jeremy Ruston contributed a single line which tells you how many bytes (that is, how much memory) the current program takes up. Just input: 9999 PRINT PEEK 16396 + 256*PEEK 16397 − 16562. If you then input RUN 9999 you'll find the screen will clear and a number will appear in the top left-hand corner of the screen, the number of bytes in the program which you currently have in the computer.

If the line is used as a direct command then the constant 16562 should be replaced by 165Ø9.

A Few Facts About the ZX81

The ZX81 display, which is *not* memory mapped (so moving graphics are out of the question, unfortunately), can be up to 24 lines of 32 characters each. The computer has its own modulator which produces an RF signal which can be connected via the lead provided to a TV set. The 1 volt composite video signal which feeds the modulator inside the ZX81 can be taken out, via a wire, and fed via a buffer circuit to a monitor.

1K byte, inside the ZX81, is supplied as standard. Extra RAM must be added externally via the edge connector at the rear. The ZX81 will take a maximum of 16K bytes. The edge connection at the rear of the ZX81 has the following: 8 data, 16 address, 13 control lines from the processor, clock, chip select for internal RAM and OV, 5V stabilised and 9 to 11V unstabilised supply lines.

The data and address lines are not buffered, and there are no I/O ports. Sinclair has plans to introduce an interface to support floppy discs, printer, teletype or I/O bus. There are sockets on the rear of the computer for connection to standard cassette recorders. Leads are provided with 3.5 mm jack plugs on them. The ZX81 loads the whole of the BASIC program into the cassette. You cannot load or fetch data on its own.

Like most BASICS, "Sinclair BASIC" is not compatible with other BASICS, although many programs can be adapted to run on the ZX81. The computer's BASIC interpreter, character set, operating system and monitor are contained in a 8K byte ROM, and works in floating point 5 byte arithmetic . Arithmetic operators provided are −, +, multiply, divide and raise to the power.

The BASIC instruction set for the ZX81 is as follows:

	Basic Commands and Statements
CLEAR	Wipes out all current variables
CLS	Clears the screen
CONT	Allows program execution to continue without losing variables.
COPY	Copy the screen onto the printer (if attached).
DIM...	Sets up an array, either numerical or string. Any number of dimensions.
FAST	Change to FAST mode.
FOR...	Sets up a loop to be terminated by NEXT.

GOSUB	Shifts control to line number specified, but return control when RETURN statement reached.
GOTO	Shifts control to line number specified.
IF THEN...	IF a conditions is true then carry out specified BASIC instruction.
INPUT	Allows user to enter data via keyboard.
LET...	Assigns a value to a variable, as in LET X = 1Ø.
LIST	Lists the current program.
LLIST	As LIST, but uses the printer instead of the TV screen
LOAD	Loads a program stored on tape.
LPRINT	As PRINT, but uses the printer instead of the TV screen
NEW	Clears the ZX81 for a new program.
NEXT	Terminates a FOR loop.
PAUSE	Stops computing and displays the screen
PLOT	Blacks in a quarter-square on the screen.
POKE	Assigns a new value to a memory location.
PRINT	Outputs data to the TV screen.
RAND	Sets random number seed to value of frame counter.
REM...	No effect
RETURN	Return control to line after the last GOSUB used.
RUN	Clears all variables, then shifts control to first line of program.
SAVE	Stores the current program on tape.
SCROLL	Moves the display file up by one line.
SLOW	Go into SLOW mode.
STOP	Returns control to the user.
UNPLOT	Blanks out a pqarter-square of the screen.
	Basic Functions
ABS	Absolute magnitude.
ACS	Arccosine in radians
ASN	Arcsine in radians
ATN	Arctangent in radians

CHR$	The character whose code is the number specified
CODE	The character code of the first character of a string.
COS	Cosine (radians)
EXP	raises e (2.718281829) to the power given
INKEY$	the key currently being pressed.
INT	the largest integer not greater than given number.
LEN	the number of characters in a string
LN	Logarithm to base e (2.718281829)
PEEK	The contents of given memory location
PI	The value 3.13159265...
RND	A random number between Ø and Ø.99999....
SGN	−1 if number negative, 1 if number positive, Ø if number zero.
SIN	Sine (radians)
SQR	The positive square root of a number
STR$	The string of characters that would appear on screen if the number was printed
TAN	Tangent (radians)
USR	calls the machine code routine at address specified.
VAL	evaluates the contents of a string (must contain a numerical expression)

MAKING THE MOST OF YOUR ZX 81

TIM HARTNELL

The ZX-81 computer from Sinclair Research, Ltd., is an exciting new breakthrough in personal computing. About the size of this book, it uses your television set for display and any cassette recorder to save programs. Though it can be used for games, for home recordkeeping, and for business functions, it is not "for" any of these uses. Because it is the least expensive, most complete, and most powerful computer of its size on the market, it is an ideal "first computer," to introduce adults and children to the world of computing.

Making the Most of Your ZX-81 is a brand new book, following in the footsteps of Tim Hartnell's highly successful book **Making the Most of Your ZX-80**, and has been completely rewritten to focus on the new improved features of the ZX-81. It includes a number of games and useful routines, but also shows you how to write programs and how to use features of the machine in those programs. **Making the Most of Your ZX-81** gives you the listings of over 60 programs, including some useful programs for home and classroom applications. You'll find this to be an invaluable resource for your ZX-81 computer!

Other books from Reston on the ZX-81 computer:

The ZX-81 Pocket Book by Trevor Toms
49 Explosive Games for Your ZX-81 by Tim Hartnell
Mastering Machine Code on Your ZX-81 by Toni Baker

Information about ZX-81 can be obtained from the National ZX Users Group, 599 Adamsdale Rd., N. Attleboro, Mass. 02760.

RESTON PUBLISHING COMPANY, INC.
A Prentice-Hall Company
Reston, Virginia 0-8359-4188-4