Leaving It To Chance

Continuing our look at Basic functions, we come to RND, which produces random (or nearly random) numbers for use in games or statistical programs

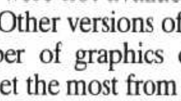
Now that we have seen how several of BASIC's functions work we shall look at one of the most commonly used — the RND function. RND is used to generate random numbers. It is also used in games whenever there is an element of chance.

Unfortunately, RND is one of the most variable 'words' in BASIC. Our description of it may differ from the way it is implemented in your home micro. Let's, therefore, clarify the differences between BASIC used in the Basic Programming course and your BASIC.

Most of our programs are based on Microsoft BASIC (or MBASIC). Microsoft is an American company and their BASIC was one of the first made widely available. BASIC is a language with no official standard, but Microsoft's version is as near to a standard as there is. Many other versions are modelled on Microsoft's, and the company has been commissioned to produce versions for several popular computers.

The chief difference between MBASIC and most of the more recent versions is that home computers now have powerful graphics capabilities that were not available when MBASIC was developed. Other versions of BASIC generally include a number of graphics commands and statements. To get the most from your computer, you will want to use its graphics capabilities to the full, and this will require a careful study of the owner's manual.

Of the various BASICS supplied with popular home computers, Sinclair BASIC (used in the ZX81 and Spectrum) and BBC BASIC probably differ most from MBASIC. Texas Instruments' version (used in the TI99/4A) also has a number of significant differences. As far as possible, we point out how to modify our programs in the 'Basic Flavours' boxes and you should refer to these if you have any problems running the programs.


CAROLINE HOLDEN

As mentioned previously, the RND function differs from version to version. Check in your BASIC manual to see how it has been implemented in your version. We are illustrating its use in a very simple dice game. As with previous programs we have done most of the work in subroutines. This technique has the advantage of making the programs more readable, easier to write and easier to debug.

The main program starts with the statement RANDOMIZE in line 20. Most, but not all, versions of BASIC need this statement to 'reseed' the RND function. It is actually quite hard to get computers to produce genuinely random numbers. Without this reseeding operation, the same sequence of supposedly random numbers would be produced each time by the RND function. Line 50 then calls a subroutine that uses RND to assign a random number to the variable D. The form we have used is:

```
320 LET D = INT(10 * RND)
```

This is the line most likely to need changing when you enter the program. Details of how different versions of RND work are given in 'Basic Flavours', so let's see what's happening in this Microsoft BASIC. The RND uses an expression (in brackets, as is usual with functions) as an option to alter slightly the sequence of numbers generated. With no expression — for example LET A = RND — the value of A will be a number between 0 and 1. We do not want a number smaller than 1 so we multiply the number by 10. This can be done like this: LET A = 10 * RND. If, for the sake of argument, RND had returned the value 0.125455, the value of A would now be 1.25455.

To eliminate the fractional part of the number and retain only the integer portion, we use the INT (integer) function like this: LET A = INT(10 * RND). Some versions of BASIC allow the upper limit of the random numbers generated to be specified in the expression used in the brackets after RND. For example, Dragon BASIC will print a whole number in the range 1 to 6 in response to: PRINT RND(6).

Since our Microsoft BASIC cannot do this, we check to see if the numbers returned are greater than 6 or less than 1 as such numbers are of no use in a dice game. This is done in lines 330 and 340:

```
330 IF D > 6 THEN GOTO 320
340 IF D < 1 THEN GOTO 320
```

If D is outside the limits 1 to 6, the GOTOs make the program jump back and try again.

Having chosen a random value for D between 1 and 6, the dice throw subroutine RETURNs to the main program. This prints the message YOUR SCORE IS A, followed by a picture of a dice. Notice how the appropriate picture of a side is selected. It is done in the SELECT subroutine. For example, if the dice (and therefore D) is a 1, line 410 calls the subroutine starting at line 530 thus:

```
410 IF D = 1 THEN GOSUB 530
```