TUTORIAL: Random Numbers ~~Distribution~~

by Rich Tietjens

We Bally users are fortunate in having a built-in random number function (RND) for our computer. Oddly, this is not common in small computers, and many hobbyists go to amazing lengths to generate "psuedo-random" numbers for their systems. However, many people do not realize what a gem the RND command really is. To understand it first one must know ~~re~~ the definition of a truly random number; i.e., any number (within the specified range) has an equal chance of appearing, and no pattern will ever predictably repeat. To see what this means, load Program 1 and run it.

When Program 1 stops (which will take quite a while), most of the screen will be black, with a jagged upper edge. This shows that all numbers within the range 1-160 have an approximately equal chance of appearing in a series of over 12,800 repetitions of Line 30 of Program 1.

Program 1 is equivalent to rolling a single 160-sided die. To see the effect of multiple "dice", load Program 2 (on "top" of Program 1) and run it.

Program 2 is equivalent to rolling 32 six-sided dice - the same type used in most board games. On the average, each die rolled produces a 3½ (obviously not something found anywhere except the edges of real dice); therefor, the graph of this program produces a "hump", or what is usually called a bell curve. This should run only about ½ as long as program 1, since only a little over 2,000 iterations of line 30 are required, on the average, to reach the top of the screen.

These two programs demonstrate to the extreme the effect of substituting a single RND statement when one programs a game or simulation, if the original used two or more dice. For example, in Monopoly, two dice produce an average roll of seven - however, the odds of rolling seven are

twice the odds of rolling a 2 or a 12. If the programmer uses N = RND(11)+1, the numbers produced will in fact be in the range of 2 to 12, but only half as many sevens as should be expected will appear. The impact of this really comes home in playing Craps!

Obviously, not all games use dice. Let us suppose you are writing a space-war game on the order of StarFire. If you want the "target" ship to move randomly, you would have a like routine to that in Program 3. With this program, the box will gyrate about until it eventually leaves the screen area altogether. If, however, you would like the target to tend in a specific direction, you could use multiple RND statements, as in Program 4. This change to Program 3 will cause the box to gradually go to the upper right, because the average number produced is +0.5.

The disadvantage to this approach is that more characters must be read and more commands executed, thus slowing the program's running speed. However, it does produce a more uniform "drift" in the desired direction, with more apparent intelligent control of the "target" ship.

Thus it should be plain that the Bally Random Number command, "RND", is truly random, and that, to simulate dice requires multiple statements. Hopefully, some of this knowledge will make itself felt when new programs are produced, with better conformance to the programmers' desires. And, if you find that you don't like the results, remember that programs are like pancakes: you usually throw away the first one.

```
1.  PROGRAM 1
2.  RND DISTRIBUTION
3.
4.
5.
10  CLEAR; NT = 1
20  FOR X=1 TO 160; @(X)=0; NEXT X
30  X = RND(160); . 1 "RND" STATEMENT
35  @(X) = @(X) + 1
40  LINE -43, X-80, 0; . MOVE START POINT
50  IF @(X)<=88 LINE @(X)-44, X-80, 1;
    MU = @(X); GOTO 30
60  GOTO 60; . DON'T STOP UNTIL "HALT"
    IS PRESSED

1.  PROGRAM 2
2.  MULTIPLE RND STATEMENTS
30  X=0; FOR I = 1 TO 32; X=X+RND(6);
    NEXT I; X=X-32; . EQUALS 32 "DICE"
```

```
1.  PROGRAM 3
2.  RND MOVEMENT
3.
4.
5.
10  CLEAR; X=0; Y=0
20  BOX X, Y, 3, 3, 3
30  H=RND(7)-4; V=RND(7)-4
40  BOX X, Y, 3, 3, 3
50  X=X+H; Y=Y+V
60  IF ABS(X)<78 IF ABS(Y)<41 GOTO 20
```

```
1.  PROGRAM 4
2.  DRIFTING RND MOVE
30  H=RND(3)+RND(4)-4; V=RND(3)+
    RND(4)-4
```