

# 1. Fooling around with RANDOM

## Introduction

One of the most exciting ideas in programming, and even in real life, is the idea of randomness. We often say things like, "Oh, that person is so random!" Philosophers often muse about the randomness of events. The industry of gambling is entirely based on the idea of random outcomes. See the picture below:

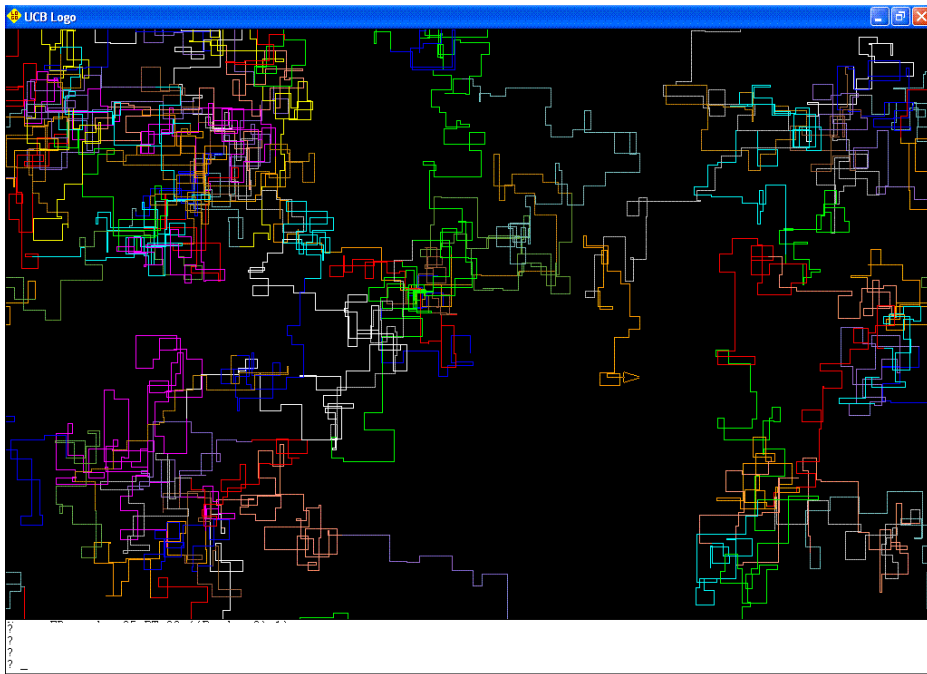


Figure 1-1

Doesn't it look like a city photographed from space? And would you believe that it is drawn using just 3 lines of Logo code?

```
REPEAT 5000 [  
  FD RANDOM 25 RT 90* ((RANDOM 3)-1)  
]
```

Read on to understand the magic `RANDOM` plays in the program above, and to learn how you can creatively use `RANDOM` for your own programs.

## The `RANDOM` Procedure

The Logo procedure `RANDOM` takes one input, say `M`, which must be a positive whole number, and then `RANDOM` outputs another positive whole number `N` such that, `N` is less than `M` and greater than or equal to 0. So, we say that `RANDOM` outputs an integer in the range (0 to `M-1`).

### Summary of `RANDOM`:

The instruction `RANDOM M` outputs an integer in the range (0 to `M-1`), i.e. an integer greater than or equal to 0 and less than or equal to `M-1`.

Mathematically speaking,  $0 \leq (\text{RANDOM } M) \leq M-1$

`RANDOM` is not totally *random*; its randomness is within *limits*.

Let's try a couple of examples.

Note: You must run these instructions a number of times to appreciate their random behavior.

```
? PRINT RANDOM 100
```

51

This will print an integer in the range (0 to 99) i.e. less than or equal to 99 and greater than or equal to 0. Every time you run this command you will get a different number in this range!

```
? PRINT RANDOM 100
```

95

What do you think the following instruction will print?

```
? PRINT (50 + RANDOM 100)
```

It will print a random integer in the range (50 to 149). Do you understand how? Think for a moment before reading on.


Here is the explanation:

PRINT will get some number – let's call it X – from the expression that follows. In other words,

$$X = 50 + (\text{RANDOM } 100)$$

Let's call the output of (RANDOM 100) by the name N. As we know, N can have any value in the range (0 to 99). If you add 50 to that, naturally we will get some number in the range (50 to 149).

For example, if RANDOM 100 gives 45, X will be 95; if RANDOM 100 gives 0, X will be 50; RANDOM 100 gives 99, X will be 149; and so on.

 Insight: Adding a number to (or subtracting from) the output of RANDOM allows us to *shift* the range of RANDOM's output.

### ***Play Time:***

1. Write a Logo instruction using `RANDOM` to get a number in the range:
  - a. (500 to 999)
  - b. (-200 to 0)
  - c. (-30 to 29)
2. The following program draws a wheel of 15 triangles. Insert a `SETPC` command and use `RANDOM`, such that every triangle is of a different color (as shown). Since the background is black, you must exclude black.

```
REPEAT 15 [  
  triangle 100  
  RT 360/15  
]
```

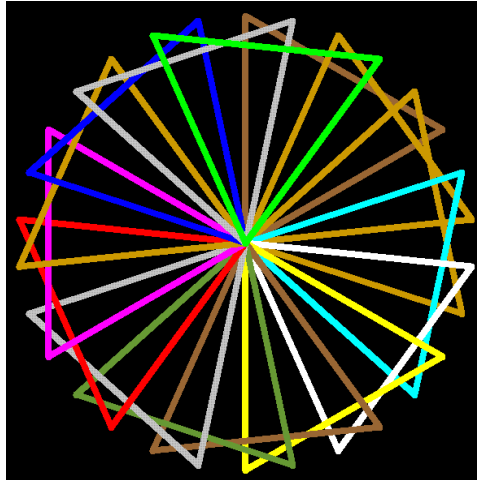


Figure 1-2

3. Write a Logo instruction using `RANDOM` to always get an *even* number in the range (0 to 198).

### **RANDOM Designs**

Ok, so `RANDOM` gives us a random number. How do we use that idea?

Let us explore and find out. See the design below:

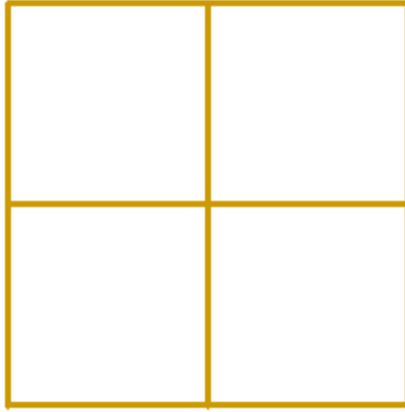


Figure 1-3

We already know how to draw this figure in Logo:

```
REPEAT 4 [square 100 RT 90]
```

What if we replace the input of `square` with a *random* number?

```
REPEAT 4 [square (RANDOM 100) RT 90]
```

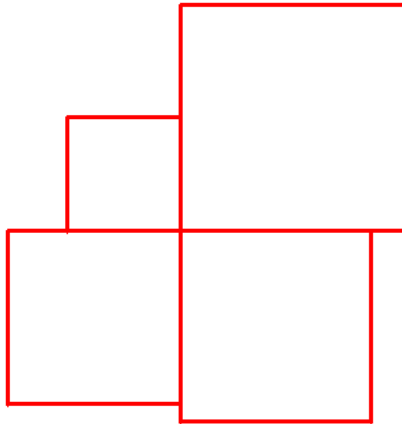


Figure 1-4

As you can see, we get this weird-looking window. In fact, your window will most likely look quite different! Get it? It's the `RANDOM` thing!

Now, if we simply put this in a loop and run it a number of times, we get something that looks like a shattered window!

```
REPEAT 400 [  
  REPEAT 4 [square (RANDOM 100) RT 90]  
]
```

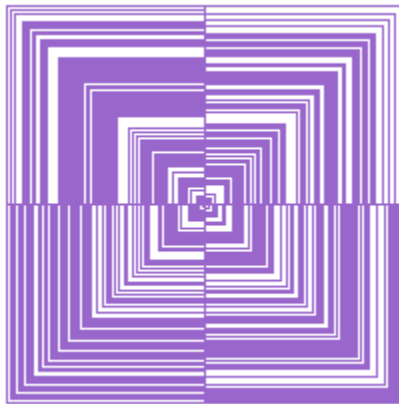


Figure 1-5

If you increase the outer loop count to a really large number (like 10000) you will just get a large painted square. That's because, when run 10000 times, `RANDOM` ends up giving practically all possible numbers in the available range (0 to 100), and so, we get squares of all possible sizes drawn very close to each other giving the effect of `FILL`.

🌀 **Insight:** If called a large number of times, `RANDOM` usually ends up giving every value in the given range – although there is no guarantee, nor is it predictable how often it will give a particular value.

So, as you see, the application of `RANDOM` really depends on our creativity.

### ***Play Time:***

1. You must have noticed that every chapter in this book begins with a peculiar rectangular design that looks vaguely like a city skyline. Here it is again:

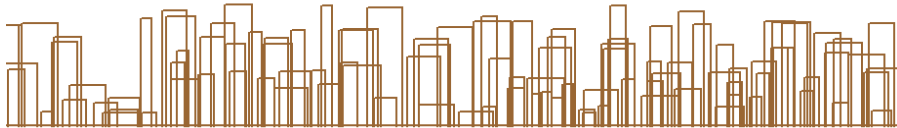


Figure 1-6

It is actually drawn using `RANDOM`. Can you figure out how it is drawn?

(Hint: Think about the rectangular wave that we drew some time back).

2. Once you get a working program above, play with it to get very tall buildings, wide buildings, colored buildings, and other interesting variations.

### **Drawing the Space Photo**

Ok, now that you have a good idea of how `RANDOM` works, can you figure out how the cityscape (pictured from space) in Figure 1-1 is created? Think about it before reading on.

If you inspect the cityscape closely, you will notice a simple procedure repeated a number times. This procedure consists of the following steps:

- The Turtle moves some distance forward
- Then, it either turns left or right, or remains straight

Making step 1 random is easy. But, how about step 2? How can we make the Turtle randomly turn left/right or remain straight? It is as if the Turtle needs to randomly choose from the commands: `LT -90`, `LT 0`, and `LT 90`.

### ***Using RANDOM on a set instead of a range:***

The above problem can be restated as: Can we use `RANDOM` to get a number (randomly of course) from the set  $\{-90, 0, 90\}$ ? If we can, we will simply provide that number to the `LT` command.

This may sound a bit tricky, but it is not all that difficult to work out. Here is how we can do it: There are 3 members in the set, and `RANDOM 3` will give us 3 possible outputs in the range (0 to 2) i.e. 0, 1, or 2. We can *shift* this range to  $-1, 0, -1$  by adding  $-1$ . Next, if we multiply by 90, we will get  $-90, 0$ , or  $-90$ . Get it?

So, the following instruction will give us the required random left/right/straight turn for the Turtle:

```
LT (90 * ((RANDOM 3) - 1))
```

(Note: Work out, in your mind, all possible values `RANDOM` will return above and what the final effect will be.)

With this, can you write the instructions to draw the cityscape? (The solution is given at the end of the chapter).

### **Using Screen Space Randomly**

Here is another interesting use of the `RANDOM` command. We could draw a number of stars (or any other object really) on the screen as if it were a night sky. We know how to use `RANDOM` to draw stars of different sizes and even colors. But, how do we make them appear at *random* locations on the screen?

Think about this a moment before reading on the solution.



Well, it is really quite simple. If you remember, we have defined and used a procedure called `jump` to move our Turtle around the screen. So, all we have to do is *randomize* the inputs of this `jump` procedure.

`Jump 100 200` moves the Turtle 100 steps horizontally and 200 steps vertically. Instead, if we call `Jump (random 100) (random 200)` it will shift at some random location within the given space constraints.

So, just play with these input numbers to get the random placement of your Turtle. Remember that the full size of your visible screen is about 1000 steps horizontally and 600 vertically.

The following program draws a starry night sky!

```
;Point star procedure. Input: 'size' is length
ERASE "pstar
TO pstar :size
REPEAT 20 [FD :size BK :size RT 360/20]
END

;Jump procedure.
ERASE "jump
TO jump :X :Y
RT 90 PU FD :X LT 90
FD :Y PD
END

;Draw at random location, with random color and size.
CS SETPENSIZE 1 REPEAT 40 [
  jump RANDOM 1000 RANDOM 600
  SETPC 1 + (RANDOM 15)
  pstar 10 + (RANDOM 25)
]
```

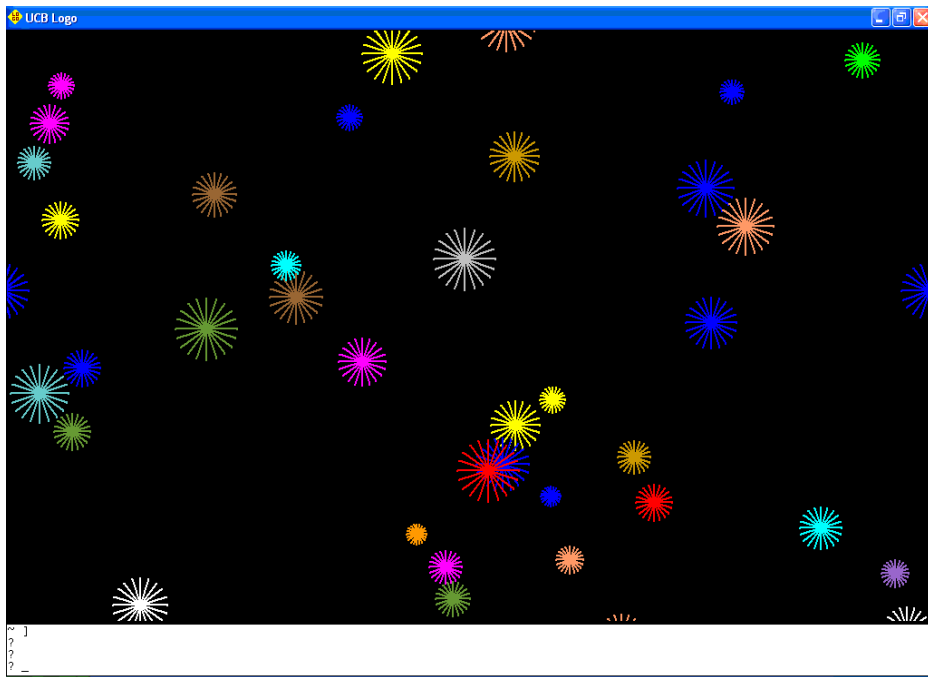


Figure 1-7

### ***Play Time:***

1. Write a program that shows a lot of bubbles floating in air. Each bubble should have a different size (and may be color too) and should appear at a random location on the screen.
2. Show your other favorite objects on the screen using `RANDOM`.

### **Key Observations**

- The `RANDOM` procedure returns a different output every time you call it even though its input is the same.
- Application of `RANDOM` for a specific property of your design requires you to convert that property into a range or set of values, which can then be controlled by `RANDOM`.

## What We Learnt in this Chapter

1. The `RANDOM` procedure.
2. The `RANDOM` procedure inserts unpredictability in our programs. We can use it in creative ways.
3. `RANDOM` only gives a number. We learnt how to connect that number to a specific property of our design that we want to randomize.

## Suggestions to Teachers/Parents

`RANDOM` is an extremely powerful idea. It will require a lot of exploration to really appreciate its power and to understand how best to utilize it. Think of patterns and designs which can be *randomized*, and give them to the children for practice.

## Solutions to Selected Problems

### **Random numbers:**

Logo instructions using `RANDOM` to get numbers from the given range:

Range: (500 to 999)

Instruction: `500 + RANDOM 500`

Range: (-200 to 0)

Instruction: `(RANDOM 201) - 200`

Range: (-30 to 29)

Instruction: `(RANDOM 50) - 30`

Even numbers in the range: (0 to 198)

Instruction: `(RANDOM 100) * 2`

Explanation: RANDOM 100 will give numbers in the range (0 to 99). Multiplying by 2 will give us only the even numbers in the range (0 to 198).

### **Wheel of Triangles:**

```
REPEAT 15 [  
  SETPC 1+(RANDOM 15)  
  triangle 100  
  RT 360/15  
]
```

Since we want all colors except black, we want to use the color range (1 to 15). This range consists of 15 numbers. Adding 1 shifts the range from (0 to 14) to (1 to 15).

### **Skyline:**

This design is a variation of a rectangular wave. Here are the procedures to draw a single wavelet and a rectangular waveform respectively:

```
;A single wavelet. Input: height, width  
Erase "wavelet  
To wavelet :h :w  
FD :h RT 90 FD :w RT 90 FD :h LT 90 FD :w LT 90  
End
```

```
;Regular wave. Input: number of waves  
Erase "wave  
To wave :n  
Repeat :n [  
  wavelet 60 20  
]  
End  
? wave 10
```



Now, we will randomize the width and height of each wavelet. We will use the range (10 to 120) for height and (10 to 30) for width.

```

;Irregular wave. Input: number of wavelets
Erase "rwave
To rwave :n
Repeat :n [
  wavelet 10+random 120 10+random 30
]
End

```

**? rwave 10**



Finally, to create the appearance of buildings overlapping each other, we will draw a large number of such wavelets such that the Turtle will wrap around the screen while continuing to draw these tall and short buildings. The resulting picture will give us a semblance of a skyline as shown in Figure 1-5.

**? rwave 120**

### **City from space:**

The procedure to draw the cityscape consists of the following steps:

- The Turtle moves some *random* distance forward
- Then, it either *randomly* turns left or right, or remains straight

This procedure is then repeated a number of times.

```

Repeat 5000 [
  FD random 25
  RT 90*((Random 3)-1)

```

```
]
```

To get a colorful cityscape, we will randomly change color after a certain number of steps. We want colors 1 to 15 (we want to avoid black since the background is black). `RANDOM 15` will give us number in the range (0 to 14) which we can shift to (1 to 15) by simply adding 1.

```
REPEAT 100 [  
  SETPC 1+(RANDOM 15)  
  Repeat 50 [  
    FD RANDOM 25 RT (90 * ((RANDOM 3)-1))  
  ]  
]
```

### **Bubbles:**

```
; Circle procedure. Input: diameter  
ERASE "dcircle  
TO dcircle :d  
REPEAT 360 [FD 3.14159*:d/360 RT 1]  
END
```

```
; Bubble. Input: number of bubbles  
ERASE "bubbles  
TO bubbles :n  
REPEAT :n [  
  PU jump ((RANDOM 800)-400) (RANDOM 250) PD  
  dcircle (10 + RANDOM 100)  
]  
END
```

```
? bubbles 50
```