From the SelectedWorks of Abhay B Joshi

Spring March 1, 2012

Logo Programming (Part 2) - a creative and fun way to learn mathematics and problem-solving

Abhay B Joshi
Sandesh R Gaikwad

Available at: https://works.bepress.com/abhay_joshi/2/

LOGO PROGRAMMING

(PART 2)

a creative and fun way to learn
mathematics and problem-solving

Abhay B. Joshi & Sandesh R. Gaikwad

All illustrations and figures in this book, including those on the book-cover, have been created using Logo programs.

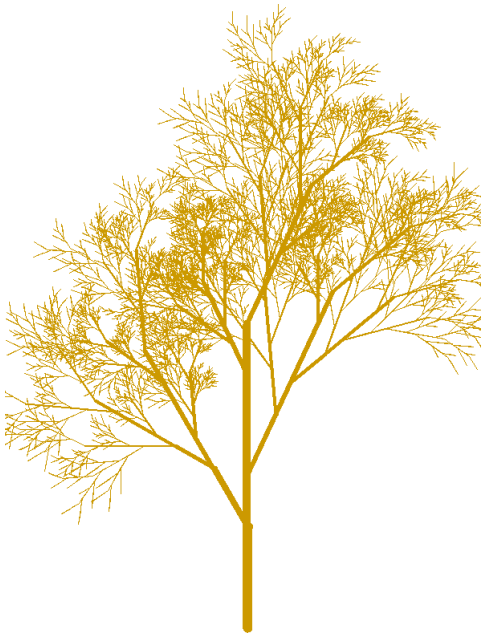
Logo Programming (Part 2)

a creative and fun way to learn mathematics and problem-solving

Copyright © 2011-12 by Abhay B. Joshi and Sandesh R. Gaikwad. All rights reserved.

Published by:
SPARK Institute, Pune (a project of Time Foundation)
Printed in INDIA
First Edition: November 2011
First Reprint: December 2011
Second Edition: March 2012

Price of the book set (Part 1 and 2): Rs. 250
How to order: http://www.spark-institute.com/logo.html



We dedicate this book to our parents.
They encouraged us to experiment and explore.

Contents

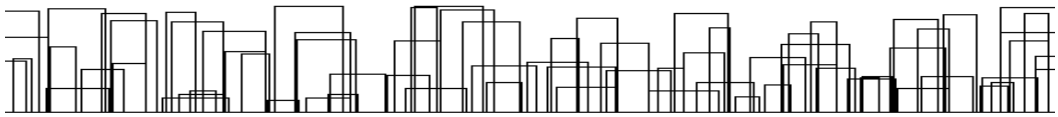
Preface

In Part 1, we developed a good understanding of the basics of computer programming, and explored interesting ideas such as, repetition, mirror images, and rotation. We became comfortable with the Logo Turtle and started teaching it lots of new words. Towards the end, we learnt how to solve complicated problems by breaking them down into smaller problems.

In Part 1, we basically went around admiring this red sports car called Logo and got to know some its capabilities.

In Part 2, we are going to climb in, take the driver's seat, and actually take this sports car for a spin. That means we are going to get a sense of its real power. In the chapter on curves, we will learn how to deal with sharp curves on the road. In "Power of Polygons" we will become expert designers of geometric patterns. And, through the ideas of randomness, recursion, and animation, we are going to get a glimpse of the wild west of programming. So, jump in, fasten your seat belts, and get ready for a rollercoaster ride in the world of programming.

Just like part 1, part 2 also contains lots of programming examples and assignments. The “Solutions” section at the end of each chapter provides solutions of some (not all) problems and assignments. Go to the website (http://www.spark-institute.com/logo.html) to see solutions of all programming assignments. But, of course, give the problems a serious try yourself first.



11. Drawing Curvy Objects

Introduction

With the exception of circles, we have so far only seen how to draw designs that contain straight lines. We don't know yet how to draw nice curvy objects like a lotus flower, a flying bird, or a fish. Or a beautiful garden as shown below:

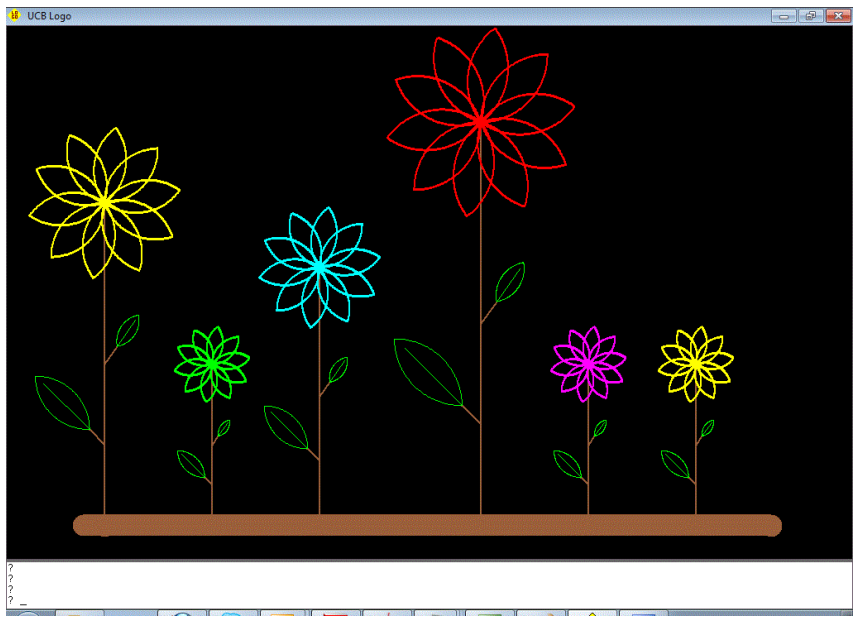


Figure 11-1

In this chapter, we will discover that we can draw curves without needing to learn anything new; not even new Logo commands. We will see that it's all just the magic of simple geometry.

The median of the petal would be at half the angle of the right turn (i.e. 45). So, in order to get a symmetric petal around the Turtle's present orientation, we should tilt the Turtle to the left by 45 before drawing the petal.

```
ERASE "petal2
TO petal2 :size ; size is diameter (of the full circle)
  LT 45
  REPEAT 2 [
    q.dcircle :size
    RT 90
  ]
  RT 45
END
```

Petal2 150

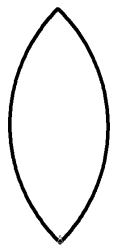


Figure 11-11

Play Time:

1. Draw a candle as shown below.

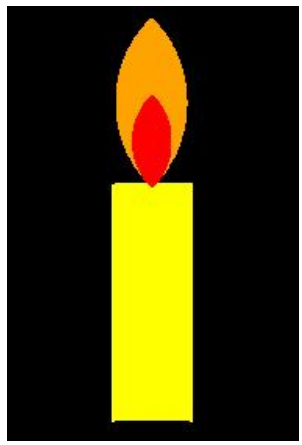


Figure 11-12

```
;Quarter circle anti-clockwise. Size is diameter.
ERASE "Q.dcircleL
TO Q.dcircleL :size
  REPEAT 90 [FD :size*3.14159/360 LT 1]
END
```

Using a combination of *right* quarter circle and *left* quarter circle we can create a nice looking wave:

```
ERASE "waves
TO waves :n :size
  REPEAT :n [
    q.dcircle :size
    q.dcircleL :size
  ]
END

waves 3 40
```

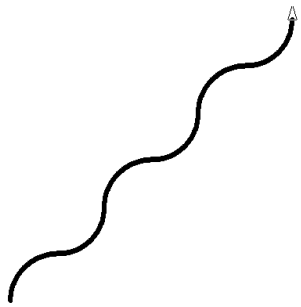


Figure 11-26

- Self-study: Make the wave above appear horizontal.

Logo Challenges

Chestnut leaves:

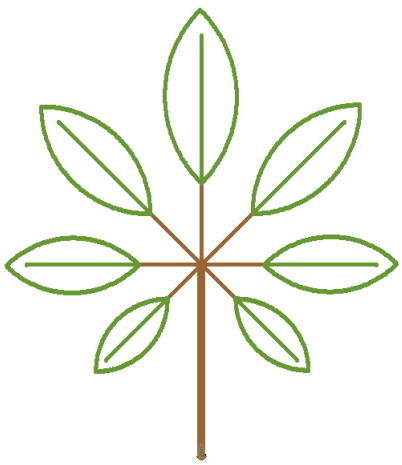
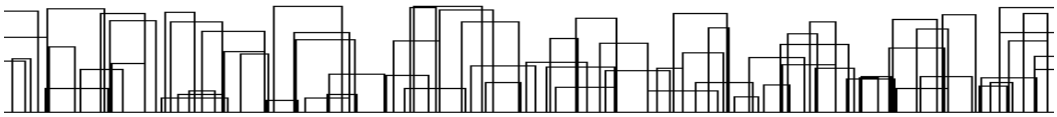


Figure 11-27

The figure shows the leaf pattern of a Chestnut tree. The amazing thing is that all Chestnut leaves appear in the exact same pattern as shown in Figure 11-27: always in groups of 7 with one large leaf at the top, and remaining 6 appearing as mirror images.

Hint: First write a procedure to draw an individual leaf (use a 'size' input to control the size of the leaf). Then, design the pattern above.



12. Power of Polygons

Introduction

Earlier, we used our knowledge of geometry to draw interesting but simple designs using regular polygons. We can explore this idea further and develop some expertise in this very exciting field of "Geometric Designs".

You must have seen patterns - like the one shown here - used for tiles, cutlery, architectures, and even paintings. All that we really need to know to create such designs is polygons, Turtle Round Trip principle, and the Logo commands that we already know. Are you interested? Then, read on.

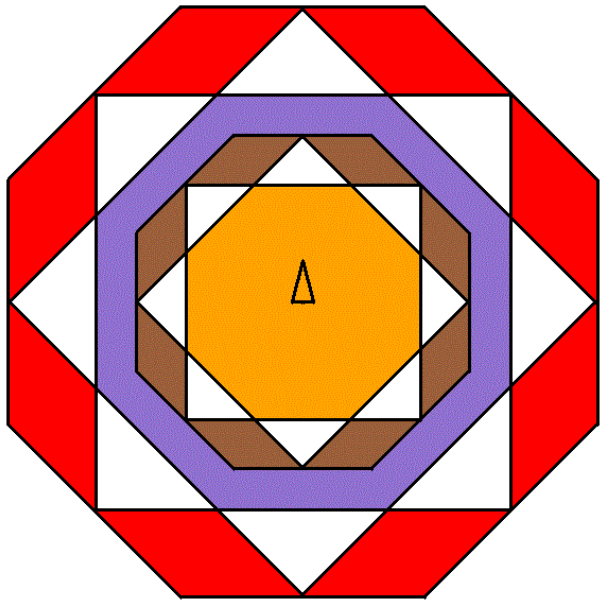


Figure 12-1

Reversing Sides

What will we need to do to draw a figure shown below?

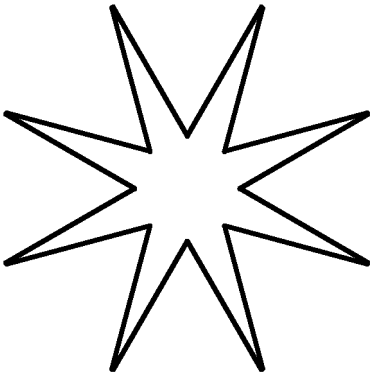


Figure 12-9

What if you were told that this figure is related to the next figure in some way? Clearly, the outside octagon has been made to hide.

But, how do we get the following figure in the first place?

Here is the code that will produce the design.

```
REPEAT 8 [
  ;Draw the inside triangle
  Polygon 100 3
  ;Draw side of octagon
  FD 100 RT 360/8
]
```

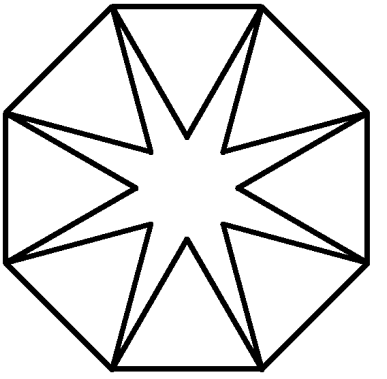
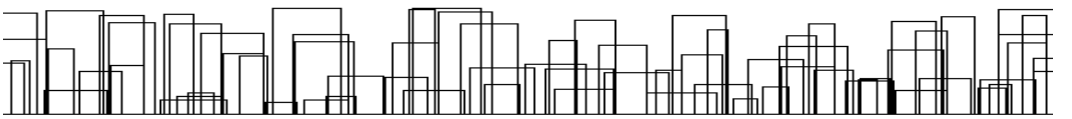


Figure 12-10

It's really the same idea as before, except this time, the triangle is drawn on the inside of the main octagon.

You could get rid of the octagon in two ways: (1) avoid drawing it in the first place – this will require you to modify the triangle procedure, or (2) Use PENERASE to erase the Octagon. The following code shows the second approach to get the star shown above.

```
PENERASE
REPEAT 8 [FD 100 RT 360/8]
PENPAINT
```



13. Perspectives and Concentric Shapes

Introduction

Do you like the design shown below?

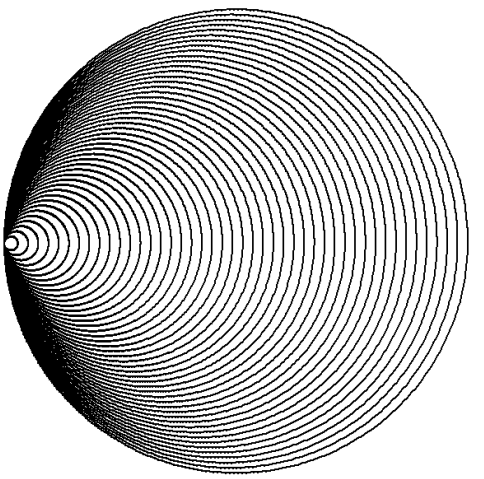


Figure 13-1

Can you guess how it can be drawn in Logo?

Yes! You can give a series of `circle` commands and increase the size slightly every time. But, that would be tedious, won't it?

How about drawing this beautiful design shown next?

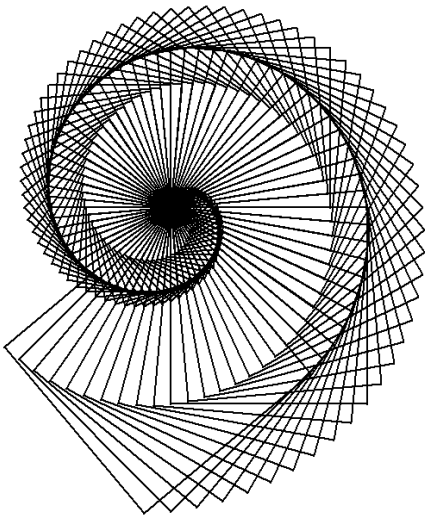


Figure 13-2

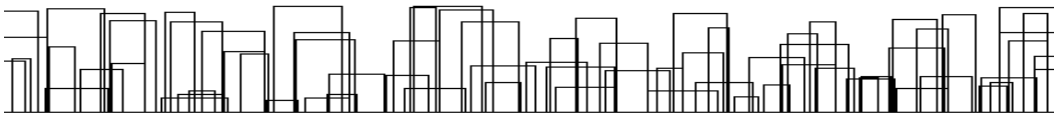
Can you guess how this design can be drawn in Logo?

If you look carefully, you will see a series of squares drawn around a point, with each square slightly bigger than the previous one. Once again, you could give a series of `square` commands and increase the size slightly every time. But, that would be horribly tedious, wouldn't it!

Continue reading this chapter, if you would like to learn a much more elegant and simpler way of drawing this sort of designs in which there is repetition for sure, but there is a twist after each repetition.

Counting Repetitions

Imagine that you want to run around a tree, and your friend is watching you standing aside. You could ask him to count 1, 2, 3… as you go around the tree. So, you could ask him any time, “Hey, what round am I doing now?” And he would tell you, "It's the 9th round", or something like that. In a sense, your friend is your *counter*.



14. Fooled by 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. Take a look at the picture below:

Figure 14-1

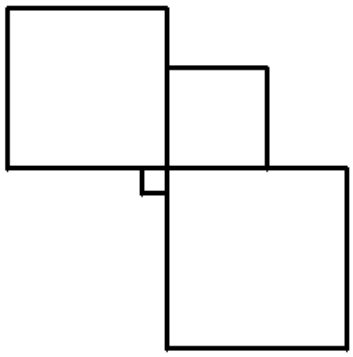


Figure 14-4

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 50 [
  REPEAT 4 [
    square (RANDOM 150)
    RT 90
  ]
]
```

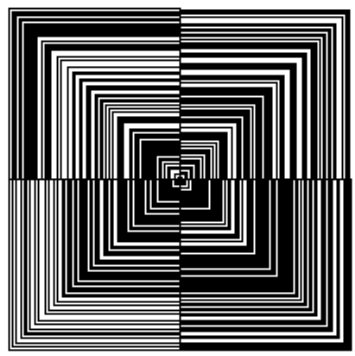
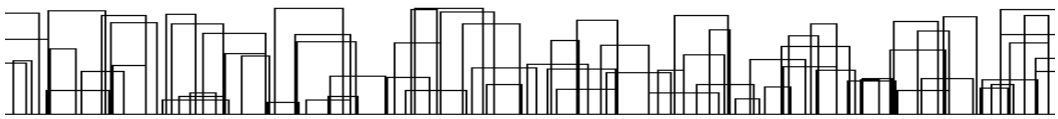


Figure 14-5

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 this is not guaranteed, 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.



15. Adding Life - Introduction to Animation

Introduction

Animation means life. Animation means motion and action. Animation means fun!

The good news is that we can do animation in Logo. So far, our Logo programs have produced nice-looking graphics and images. We are now ready to make our Logo drawings come alive and perform actions. As you might know already, animation is an illusion of motion (or action) created by persistence of vision and a successive display of slowly changing pictures.

For a quick demo of animation, just run the following program and watch the screen. Press Ctrl-Q (Alt-S in UCB Logo 6.0) when you have seen enough!

```
REPEAT FOREVER [
  FD 100 BK 100  ; Draw a line
  WAIT 5         ; Wait for some time
  PENERASE       ; Turn the Eraser mode on
  FD 100 BK 100  ; Erase the line drawn earlier
  PENPAINT       ; Restore the Pen
  RT 5           ; Turn right by 5 degrees
]
```

Isn't it interesting? Can you figure out how the program works by reading the comments in the code? (Note: `REPEAT FOREVER` is a special form of `REPEAT` in which things go on, well, forever!)

Read further to learn how to write animation programs in Logo.

```
ERASE "Bcwheel
TO Bcwheel
  SETPENSIZE 5
  REPEAT 100 [
    FD 50 BK 50 RT 360/100
  ]
  SETPENSIZE 20
  REPEAT 10 [
    PU FD 50 FD 20 PD
    FD 50 BK 50
    PU BK 20 BK 50 PD
    RT 360/10
  ]
END

CS SETBG 7 SETPC 0
Bcwheel
```

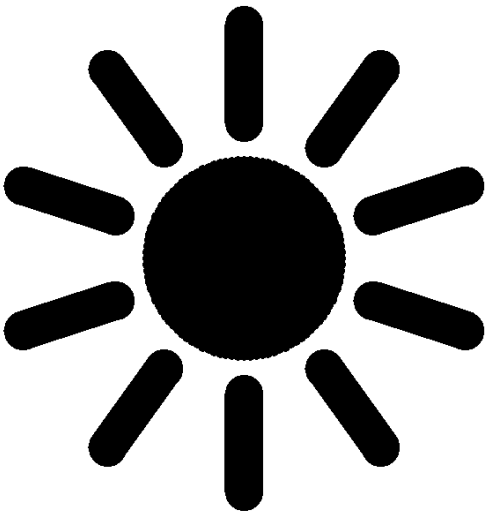


Figure 15-7

Shapes with Wings and Legs

Well, we don't mean literally, but in the sense that our stationary shapes will start moving around the screen as if they were living or motorized objects.

In a previous chapter, we wrote a procedure to draw a block arrow as shown below. Now, we will see how we can make it move around the screen.

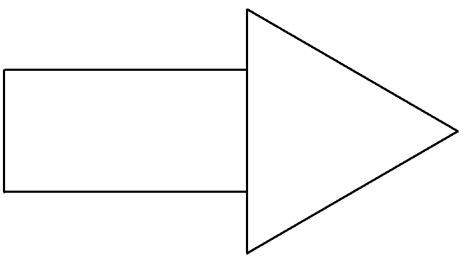
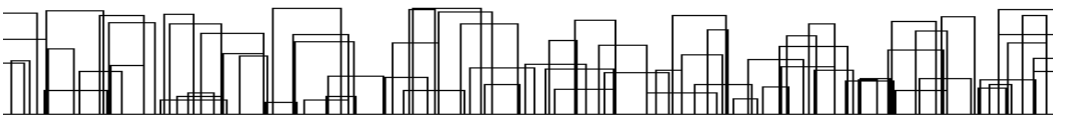


Figure 15-8

To create the moving arrow effect, we could try the following steps:

1. Draw a block arrow.
2. Wait for a short time.
3. Clear the screen.



16. Turtles that Climb Trees

Introduction

You probably noticed the Fern branch and Tropical tree drawn on the very first page of this book. Here they are once more:

Figure 16-1

Believe it or not, these figures have been drawn using Logo programs – programs containing not more than 15 instructions each! They are based on an interesting idea called *Recursion*.

Continue reading this chapter to learn about *recursion* and how to draw these sort of designs in Logo.

Hint: The following instructions will draw a single curved line and return the Turtle to its base (this code would actually be level 1 of the recursion):

```
REPEAT 10 [FD 25 RT 5]
REPEAT 10 [LT 5 BK 25]
```

You just have to recursively implant the same curved line at a number of places on both sides.

Logo Challenge

The trees shown next have the following new features:

- The tree looks different every time you call the instruction (even if all inputs are the same)
- The tree shows fruit at some places. The fruit shows up at random places, and the amount of fruit is controlled through an input.

Hints: (1) Obviously, we have used `RANDOM` to make things happen differently every time. (2) To draw the fruit, add code just before the `STOP` command, because that's where the branches end.

Figure 16-18