Introduction to Artistic Programming

Robert J. Krawczyk
College of Architecture, Illinois Institute of Technology
3360 South State Street, Chicago, IL, 60616 USA
E-mail: krawczyk@iit.edu

Abstract

The goal of this workshop is learn how to develop simple graphic shapes into complex patterns and transformations using a programming language called Logo. Learn how to create a range of designs based on simple polygons to complex multi-colored overlays. Begin to see how computer graphics can be used to create and investigate a variety of artistic designs. See how a programming language can be used to explore the underlying methods of mathematics, geometry, and art.

1. Introduction

Even though the artistic element of computing will be highlighted, this workshop uses a very simple learning model to introduce the visualization of mathematics and geometry. All the designs developed only require the understanding of some very basic geometric concepts. Programming requires a step-by-step planning of procedures, includes the concept of trial and error, allows for teaching in an incremental approach to solve problems, is based on the concept of working from success, and has the ability to repeat previous ideas over and over. Having a method which can immediately display the results of an operation visually is very powerful. The workshop also shows how computers can be used to investigate an idea, not just produce an answer to a specific question.

The programming language used in this workshop is a version of Logo. Logo was originally developed by Daniel Bobrow and Wallace Feurzeig at Bolt, Beranek and Newman, Inc., and Seymour Papert, at the Massachusetts Institute of Technology in the 1960's. The goal was to allow people to use computers to manipulate things more familiar than numbers and equations. Originally, simple commands were used to control a robotic "turtle" which held a pen. It quickly grew to become a teaching tool for children for geometry and programming through drawing.

```
forward 100
right 90
forward 100
right 90
forward 100
right 90
forward 100
right 90
```

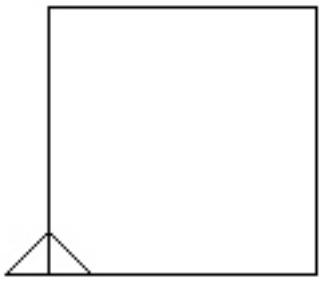


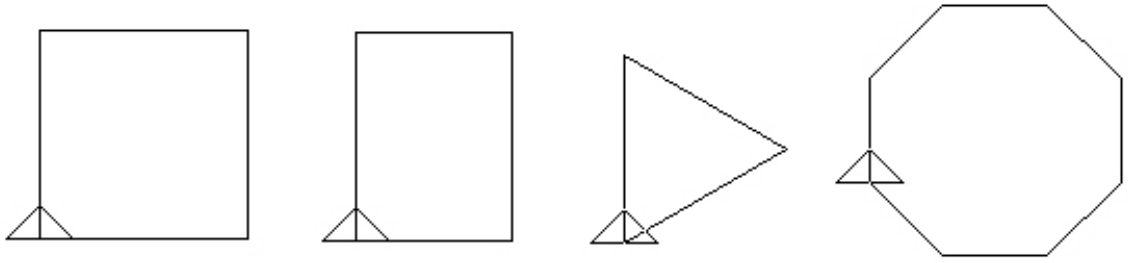
Figure 1: *Instructions for a square*

Logo is based on a simple method called "turtle graphics". Turtle graphics consists of a series of commands that instruct a turtle, now the cursor on the screen, to move from one position to another. As it

moves it leaves a trail behind it, this becomes a drawing on the screen. Turtle movements begin with a series of very simple commands: forward, back, turn right and turn left. For these a distance or angle is required. All moves are relative to the current location of the turtle. For example, to draw a square a series of forward and right turns instructions are required, as shown in Figure 1. As each instruction is entered, the corresponding drawing action is immediately displayed. In addition to such simple actions, powerful computer science concepts of procedures, recursion, programs-as-data are built into Logo.

2. Shapes

There are two basic prerequisites needed to develop a shape: the shape has a known and consistent geometry and the shape can be expressed by a few simple measurements. For example, the square, the side dimension is the only measurement needed to define it; we also know that the adjacent sides are perpendicular and that opposite sides are parallel. Other considerations include the location of the starting point for the shape, the individual steps to complete it, and that the cursor needs to return to its starting position. One of the best ways to determine the steps needed is to draw the shape for yourself and record what you do. Continuing with the square: once the steps are confirmed individually we can consolidate them into a procedure. A procedure may optimize the steps and change any constant measurements to ones that are variable. For the square, we see that just two steps can be repeated four times. Figure 2 shows the procedure *square*. To draw one you only need to specify the procedure name and the values for the parameters, in this case the side dimension for the square: *square 100*. The individual steps do not need to be entered again. The procedure square can now be used to draw a square of any size by changing the parameter not the actual steps. Figure 2 also shows the procedures for other common shapes: rectangle, triangle, and polygon. The small triangle displays the shape's starting point also referred to as the origin.



a. *b.* *c.* *d.*

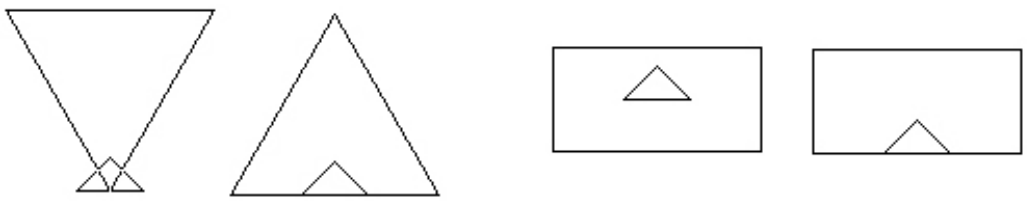
```
to square :side
repeat 4 [forward :side right 90]
end
```

```
to rectangle :side1 :side2
repeat 2 ~
[forward :side1 right 90
 forward :side2 right 90]
end
```

```
to triangle :side
repeat 3 [forward :side right 120]
end
```

```
to polygon :n :side
repeat :n [forward :side right 360/:n)]
end
```

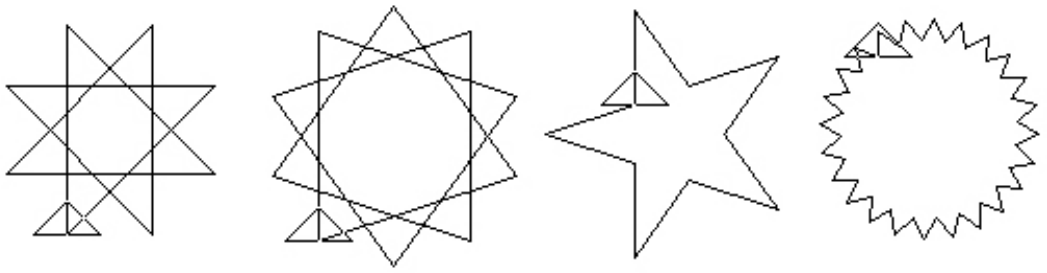
Figure 2: *Basic shapes*



a. *b.* *c.* *d.*

Figure 3: *Shapes with alternate origins*

The selection of a starting point may determine the overall orientation of the shape or may be determined due to a specific intended use. The triangle in Figure 2 has its origin at one the corners, for some drawings that might be inconvenient, so Figures 3a. and b. show the same triangle based a revised procedure that constructs the triangle with the its starting point at the apex and the midpoint of the base. These shapes could also be called arrowdown and arrowup. The second set of shapes, Figures 3c. and d., show a rectangle with its origin at the center and at the midpoint of the bottom edge.



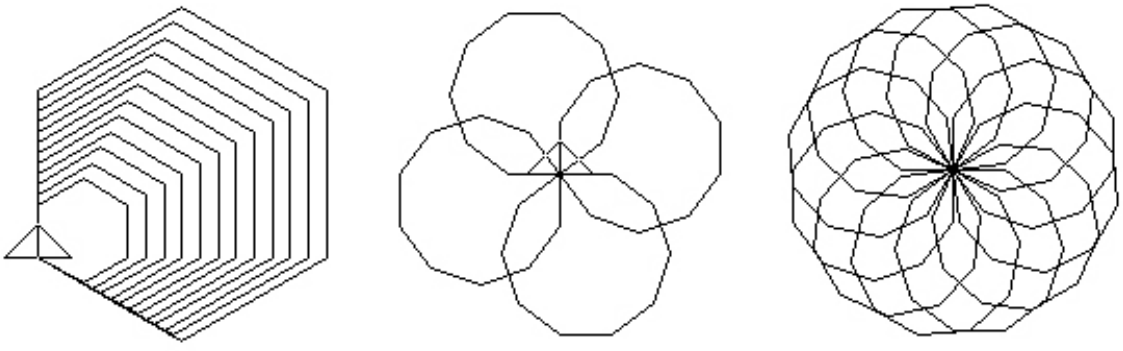
a. *b.* *c.* *d.*

Figure 4: *Variations on the polygon procedure*

Interesting results can be obtained when you try to expand the normal geometric definitions of shapes. For example, the polygon procedure is based on a number of sides equally distributed over 360 degrees. The question could be asked is what would happen if the turning angle does not add up to 360 degrees. Figure 4 shows a modified polygon procedure that has as its parameters number of sides, turning angle, and radius. In these cases the procedure had to be repeatedly executed until the overall figure closed. An alternative method would be to develop a computation to predict such closure. The procedure could then be used to test the computation.

2. Repetition

Once a shape is developed into a procedure, there are a number of operations and transformations that can be performed with it or on it. A shape can be repeated: in a lineal manner, in a grid pattern, in a circular manner, rotated, scaled, or mirrored. Figure 5a. shows one example of scaling: a hexagon is drawn twelve times, each time the radius is increased a constant amount. Note the starting point becomes the stationary point for scaling. Figures 5b. show a decagon drawn four times, each time rotated 90 degrees. A more interesting application of rotation is Figure 5c., a decagon drawn twelve times, each time rotated 30 degrees.

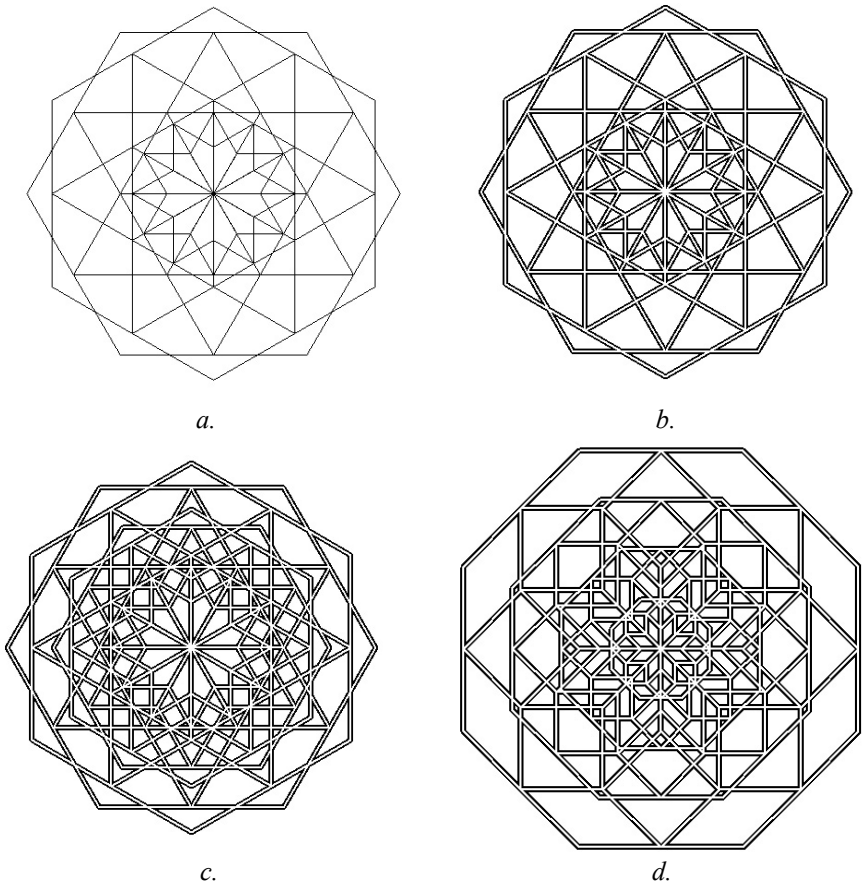


a. *b.* *c.*

Figure 5: *Repetition with scaling and rotation*

3. Moving Towards an Artistic Expression

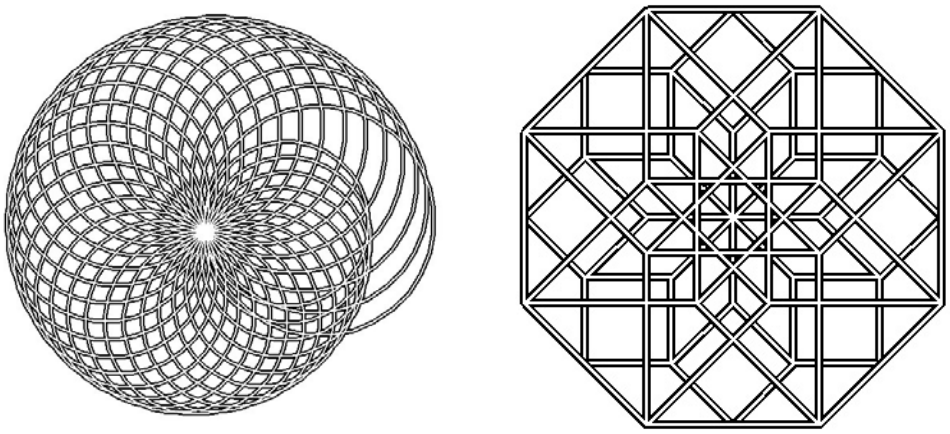
After we have developed a library of basic shapes and have reviewed some simple operations, we can discuss how these simple figures could be artistically represented. One method is to overlay one figure over the other. Figure 6a. shows one hexagon at radius 50 rotated twelve times drawn over another hexagon at radius 100 rotated twelve times. It still remains a simple but a more complex line figure. We can further enhance this arrangement by separating the figures in a foreground and background manner. This can be done by setting an appropriate line thickness and color as shown in Figure 6b. In this case, a thinner version of the pattern is placed over a thicker one. Figure 6c. shows the overlay of three hexagons patterns and Figure 6d. shows the overlay of four patterns.



a. *b.* *c.* *d.*

Figure 6: *Overlaid figures*

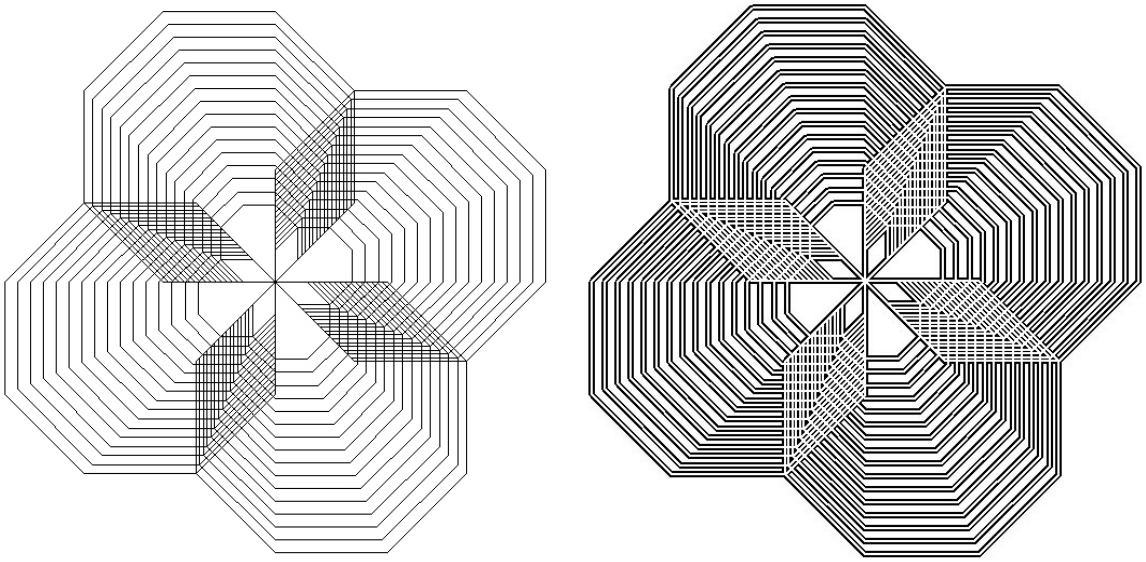
Figure 7a. shows a procedure that combines scaling and rotation 32 times on a 32-sided polygon and Figure 7b. combines a series of squares and octagons placed along a larger octagon. This case demonstrates the concept of placing one shape over the control points of another and opens the possibility of an infinite number of patterns of repetition.



a. *b.*

Figure 7: *Variation of repetition*

Figures 8 and 9 show a sequence of design steps to produce a final image. Figure 8a. shows the first step, using a combined scaling and rotation procedure an octagon is rotated four times and scaled sixteen times. To better highlight the lines, Figure 8b., a duplicate series is placed in the background for contrast, with a line thickness change. For the final image, Figure 9, the background series is increased to eight rotations and the foreground series remains at four. The locations, where one series intersects the other, form solid lines and areas that give the image a very complex and richly layered pattern. The actual image is red and black.



a. *b.*

Figure 8: *Initial steps for an overlaid image*

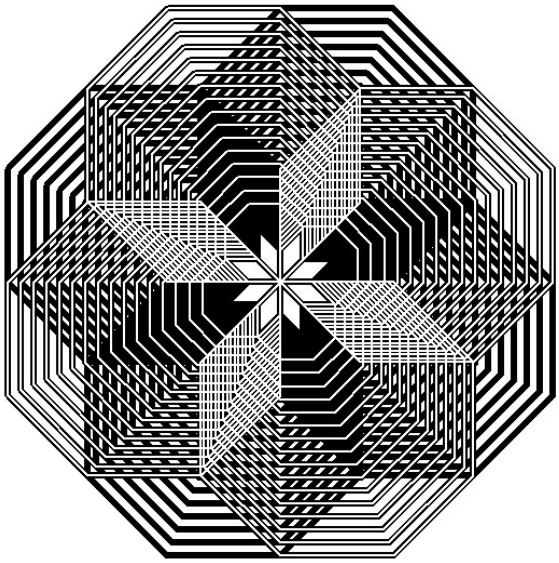


Figure 9: *Final image combining overlays*

Resources

Logo background and resource website:

S. Papert, Mindstorms: Children, Computers and Powerful Ideas, Basic Books. 1980.

The Logo Foundation, http://lcs.www.media.mit.edu/groups/logo-foundation/

Books on the development of turtle graphics:

H. Abelson and A. diSessa, Turtle Geometry, MIT Press. 1968.

J. Clayson, Visual Modeling with Logo, MIT Press. 1981.

MSWLogo download and reference information:

MSWLogo, http://www.softronix.com/download/mswlogo65.exe (1.7MB)

MSWLogo, http://www.softronix.com/logo.html

Logo tutorials:

P. Dench, A Turtle for the Teacher.
http://www.ecu.edu.au/pa/ecawa/sig/logo/paul_dench/turtle//turtle-6.html, Logo resources

J. Muller, The Great Logo Adventure, Doone Publications. 1997.
http://www.doone.com/grt_logo_advntre.html, CD book and a copy of MSWLogo