SLOGO
Simple Line Oriented Graphical Output

Phil G. Sphicas
pgs2111@columbia.edu

February 9, 2010

Abstract

SLOGO is drawing language, capable of producing graphical output to the screen using a relative cursor (the "turtle") that can move about the canvas in straight lines.

1 Introduction

The Logo Programming Language was developed in the late 1960s primarily as a tool for learning. One of its most popular features was the Turtle, a robotic device whose movements could be controlled from the computer. Equipped with a pen, which could be raised or lowered, the turtle could be used to draw pictures on the floor.

SLOGO was primarily inspired by the author's fond memory of the "Turtle Graphics" features of Logo. SLOGO is an interpreted language that produces on-screen graphics in a window through the movements of the turtle. The goal of SLOGO is to further at least one person's learning (in the area of compiler design), and hopefully to create some nifty graphics.

2 Concepts

2.1 Window

The window is where the turtle lives, and where output is displayed. It is a rectangular canvas of configurable size whose coordinates are numbered as a Cartesian plane. Position *(0,0)* represents the lower left corner of the window. The x axis increases from the left of the window to the right, and the y axis increases from the bottom of the window to the top.

The default window size is 760 x 496 units.

2.2 Turtle

The turtle lives in the window, and has a position and an orientation.

The turtle's position is specified in *(x,y)* coordinates. The turtle starts at position *(0,0)*.

The turtle's orientation is specified in degrees *(0 .. 359)*. *0* degrees is along the x axis in the positive direction. By convention, the turtle starts with an orientation of *90*, which is along the y axis in the positive direction.

The turtle can move forwards or backwards, in the direction that it is pointing. It can also turn left or right. If an operation would cause the turtle to exceed the bounds of the window, an exception is thrown and the program will terminate.

2.3 Pen

The pen is held by the turtle. The pen can be up or down. If the pen is down (i.e. touching the drawing surface), then the turtle will leave a pen trail as it moves. If the pen is up, no lines will be drawn when the turtle moves. The pen also has attributes such as color and width.

3 Drawing Functions

The following is a descriptions of SLOGO's built-in drawing functions.

3.1 Window operations

- `SETWS` *(x,y)*
  Sets the size of the output window. Anything drawn to this point will be cleared.

3.2 Turtle operations

- `FD` *s*
  Move the turtle forward *s* steps in the direction that it is currently facing. If the pen is down, a line will be drawn from the starting position to the ending position.
- `BK` *s*
  Move the turtle backwards *s* steps in the direction that it is currently facing. If the pen is down, a line will be drawn from the starting position to the ending position.
- `RT` *d*
  Turn the turtle *d* degrees to the right, relative to the current orientation.
- `LT` *d*
  Turn the turtle *d* degrees to the left, relative to the current orientation.
- `SETPOS` *(x,y)*
  Move the turtle to position *(x,y)*. The current orientation is preserved. Regardless of pen up or down status, this does not produce any output.
- `ORIENT` *d*
  Change the orientation of the turtle to *d* degrees. The current position is preserved.

3.3 Pen operations

- `PU`
  Pick the pen up. With the pen up, turtle movement does not produce screen output.
- `PD`
  Set the pen down. With the pen down, turtle movement does produce screen output.
- `SETPC` *c*
  Change the pen color, based on the following values of *c*:
  0. black (the default)
  1. blue
  2. green
  3. cyan

4. red
5. magenta
6. yellow
7. white (can be used to erase)

4 Other Language Features

4.1 Comments

Single-line comments are supported. They begin with the characters `//` and continue to the end of the line.

```
// This is what a comment looks like
FD 10;  // This is also ok
```

4.2 Functions

User-defined functions are allowed. They must be defined before being used, using the `FUNCTION` keyword.

Functions can take 0 or more arguments. All function arguments are integers.

When calling a function of 0 or 1 argument, parentheses are optional. When calling a function with 2 or more arguments, parentheses are mandatory, and the arguments must be separated by a comma.

All functions return an integer. If no explicit return value is set, then the value of the last executed command is returned.

```
// This function moves the turtle back to
// its initial position and orientation
FUNCTION gohome {
  SETPOS(0,0);
  ORIENT(90);
}

// These function calls are all ok
gohome;
gohome();
gohome ();
```

4.3 Variables

User-defined variables are allowed. They must be declared before being used, using the `VAR` keyword. Declaration and assignment in one statement is not allowed.

Variables declared within a function definition are locally scoped within that function.

```
VAR i;  // this is ok. i assumes a default value of 0
VAR j = 1; // not allowed
```

Assignment is performed through the `=` operator.

```
VAR i;
i = 1;
```

4.4 Types

SLOGO only supports integer data types. Operations that would result in non-integer values are rounded. Booleans are simulated using integers, with *0* being `false`, and *-1* or any non-zero value indicating `true`.

4.5 Flow Control

SLOGO supports the following flow control structures.

- `REP` *n* { ... }

  `REP` *n* allows repetition of a block of SLOGO code *n* times.

- `IF` *expr* { ... } `ELSE` { ... }

  Tests *expr*, and if it is true, executes the first block of code, or if it is false, executes the second block of code. The `ELSE` statement is mandatory.

- `WHILE` *expr* { ... }

  Tests *expr*, and if it is true, executes the block of code. Continues doing this until *expr* is false, at which point, the loop exits and program execution continues beyond the block of code contained in the while loop.

4.6 Standard arithmetic operators

SLOGO is capable of performing integer arithmetic. All numbers are treated as integers. The following operators are supported for addition, subtraction, multiplication, and division:

```
+  -  *  /
```

4.7 Comparison operators

SLOGO is capable of performing comparisons. The following operators are supported for less than, greater than, less than or equal to, greater than or equal to, and equals:

```
<  >  <=  >=  == !=
```

5 Example

The following listing shows a complete SLOGO program.

```
// Define a function that can draw a square
FUNCTION SQUARE(size) {
  REP 4 {
    FD size;
    RT 90;
  }
}

// Define a function that can draw a rectangle
FUNCTION RECTANGLE(length,width) {
  REP 2 {
    FD length;
    RT 90;
    FD width;
    RT 90;
  }
}

// Define a function that can draw a polygon
FUNCTION POLYGON(size,sides) {
  REP sides {
    FD size;
    RT (sides/360);
  }
}

// Define a function that will draw several squares
FUNCTION SQUARES {
  VAR i;
  i = 1;
  WHILE (i <= 5) {
    SQUARE(10 * i);
    i = i + 1;
  }
}

// Set up the window dimensions as 300x300
SETWS(300,300)

// Move turtle to position (25,25)
SETPOS(25,25);

// Draw a square of size 50x50
SQUARE(50);

// Move turtle to position (50,50)
SETPOS(50,50);

// Draw a 75 x 125 rectangle
RECTANGLE(75,125);

// Move turtle to position (150,150)
SETPOS(150,150);

// Draw an octagon of size 50
POLYGON(50,8);

// Draw several squares at position (25, 200)
SETPOS(25,200)
SQUARES;
```

This would generate the following output.

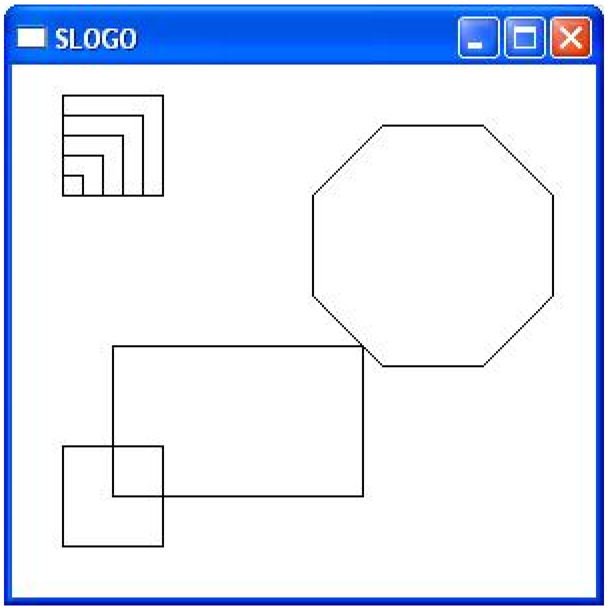

Figure 1: SLOGO Sample Output