(SSOL) Simple Shape Oriented Language

Madeleine Tipp	Jeevan Farias	Daniel Mesko
mrt2148	jtf2126	dpm2153

Description:

SSOL is a programming language that simplifies the process of drawing shapes to SVG files. It will feature built-in types to represent geometric shapes and standard library operations to render them in SVG files. To output to an SVG file, the programmer must instantiate a Canvas object, and add at least one element object to it. The Canvas object can then be passed to the *draw()* library function with a String filename to create an SVG file.

Elements are added to a Canvas object programmatically-- the Canvas is essentially a queue. When a Canvas is drawn, the last element to be added to the canvas gets the highest z-index. There are four types that represent elements: Ellipses, Polygons, Lines, and Bezier curves. Ellipses and Polygons are defined by center point (a 2D pixel location) and a radius and/or sidelength. Lines are defined by two points. A Bezier curve is a parametrically defined curve with a beginning and ending anchor point and a control point for each. Our language encodes the data for each shape that the user creates. Upon running of the executable created by compilation, this data is rendered in SVG format.

Types of programs to be written in the language:

The intended use of SSOL is creating images. The user defines the size of their canvas along with the size, location, and orientation of the various shapes that they would like to put on it using the “draw()” function. Without calling draw(), SSOL functions as a minimal, general purpose programming language similar to C.

Types:

Primitives:

Type	Definition	Example
int	Signed integers	5, -5
float	Floating point decimals	3.2, -4.7

bool	Boolean values	True, False
char	ASCII characters	‘Z’, ‘7’
String	Array of chars	“abcd”, “hello”
Array	Sequential block of a variables of a particular type	int nums[5]; nums[3] = 100;

Complex Types:
The following built-in types are represented as objects. Each has its own constructor, that modify certain public fields of the object. For the types Line, Bezier, Ellipse, and Poly, there are private fields for fill and stroke colors (in RGB and Alpha values), that can only be modified by the fill() and stroke() functions. The Canvas object contains a private pointer to the first element (either a Line, Bezier, Ellipse, or Poly), and each element contains a private pointer to the next element in the Canvas. The last element has 0 for this value.

Object	**Description**	**Constructor Parameters**	**Functions**
Canvas	Represents the window in which the elements are drawn.	**length** (float) - pixel length **width** (float) - pixel width	
Line	Straight line between 2 points.	**x1** (float) - x coordinate of first point **y1** (float) - y coordinate of first point **x2** (float) - x coordinate of second point **y2** (float) - y coordinate of second point	**stroke(float, float, float, float)** - takes R,G,B, and Alpha values, respectively, for border colors.
Bezier	A bezier curve with 2 anchor points and 2 control points.	**x1** (float) - x coordinate of first anchor point (origin) **y1** (float) - y coordinate of first anchor point (origin) **x2** (float) - x coordinate of first control point	**stroke(float, float, float, float)** - takes R,G,B, and Alpha values, respectively, for border colors.

		y2 (float) - y coordinate of first control point **x3** (float) - x coordinate of second control point **y3** (float) - y coordinate of second control point **x4** (float) - x coordinate of second anchor point **y4** (float) - y coordinate of second anchor point	
Ellipse	Elliptical shape with a center point, x axis, and y axis. Center point is origin.	**x** (float) - x coordinate of center **y** (float) - y coordinate of center **xAxis** (float) - length along x axis **yAxis** (float) - length along y axis	**stroke(float, float, float, float)** - takes R,G,B, and Alpha values, respectively, for border colors. **fill(float, float, float, float)** - takes R,G,B, and Alpha values, respectively, for fill colors.
Poly	Regular polygon shapes defined by number of sides and side length. Center point is origin.	**x** (float) - x coordinate of center **y** (float) - y coordinate of center **sideLen** (float) - length of each side **numSides** (float) - number of sides	**stroke(float, float, float, float)** - takes R,G,B, and Alpha values, respectively, for border colors. **fill(float, float, float, float)** - takes R,G,B, and Alpha values, respectively, for fill colors.

Keywords:

// this is a comment (multi-line comments not supported)

Keyword	Usage
if	Control flow
while	Looping structure
for	Looping structure
return	Designates end of function, followed by value to be returned (if non-void function)

Operators:

Operator	Usage	Example
+, -, *, /, %	Addition, subtraction, multiplication, division, mod	5 + 4 7 - 2 6 * 3
=	Assignment	x = 5
==, !=	Equality	5 != 10
<, >, <=, >=	Comparison	6 > 3
&&, ||, !	Logical AND, OR, NOT operators	5
|	Sequentially group elements to add to Canvas object	Canvas x = e1 | e2;
|=	Add elements to existing Canvas object	x |= e3 | e4;

Library Functions:

Function	Description

draw(Canvas, String)	*Void*. This function outputs an SVG image with all the information stored in the Canvas object. The second parameter is the filename. When the executable is run, it will make as many output files as there are draw() calls.
rotate([Line, Bezier, Ellipse, Poly], float)	*[Line, Bezier, Ellipse, Poly]*. Can rotate any element around its origin point by a value specified in *degrees.* Returns a copy of element.
translate([Line, Bezier, Ellipse, Poly], float, float)	*[Line, Bezier, Ellipse, Poly]*. Reposition element with respect to its origin. Returns a copy of element.
random(int, int)	*Int.* Takes a range of integers between the values of 0 and 255 and picks a random integer value within that range as the RGB value. This will be linked to the C random function.

Sample Program:

Triangles and Squares

This program makes a simple design. A square canvas of 100x100px default size. 4 squares make up a frame, each with a random stroke color. In the center is a star made up of 10 triangles each with a random stroke and fill color that are rotated 36 degrees more than the previous and are layered on top of each other. Triangles have 50% opacity.

```
int main(){
    //create a canvas with default size of 100x100px
    Canvas myCanvas = Canvas(100,100);
    for(int i = 0; i < 10; i++){
      //create triangle and rotate
      Poly tri = Poly(50,50,20,3);
      tri = rotate(tri, i/10 * 360);
      tri.stroke(random(0,255),random(0,255),radom(0,255));
      tri.fill(random(0,255),random(0,255),random(0,255),0.5);
      myCanvas |= tri;
    }
```

```
    for (int i=0; i<4; i++){
     Poly square = Poly(50,50,100-i*3,4);
     square.fill(random(0,255),random(0,255),
          random(0,255),0.5);
    }
    draw(myCanvas, "myFile");
    return 0;
}
```