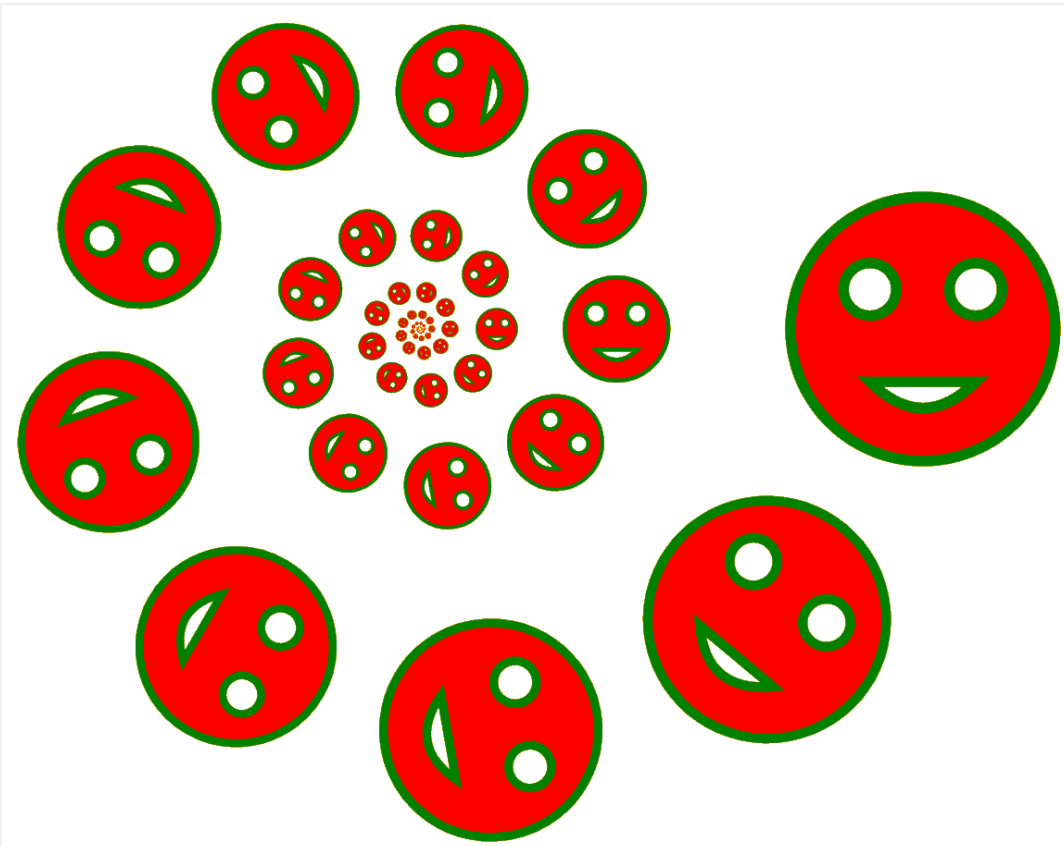


```
SVG500,500
#"smiley"
M250,250 F"RED"
CIR50 F"ORANGE"
M230,270 Q250,290 270,270 Z
M230,235 CIR10
M270,235 CIR10
RTS
#"main"
M0,0 F"ORANGE" RCT500,500 SC"GREEN"
SW4
$D=1.05 $N=60 $W=0
#"loop"
ROT250,250,$W
SCL250,250,$D
TRL190,0
JSR"smiley"
POP POP POP
$N--
$W+=40
$D*=1.11
BNE"loop",$N
```

SvgScript

Leonard Ilanius 27 april 2022

Introduction

SvgScript is an *ad hoc* solution to certain musings and problems pertaining to efficient semantic coding of graphics. As of now the script may be somewhat lacking regarding legibility as it is perhaps too concise.

The front page example illustrates what the "language" has to offer, it is essentially a spiral of rotated receding smileys. Here is a line by line description of the front page.

1		**SVG500,500**	We declare an image of width 500 and height 500 pixels
2		**#"smiley"**	Subroutine "smiley" starts here. The name is preceded by a #
3		**M250,250 F"RED"**	**M**ove to coordinate 250,250. Set **F**ill color to "RED"
4		**CIR50 F"ORANGE"**	Draw a **CIR**cle of radius 50px. Set fill color to "ORANGE"
5		**M230,270** **Q250,290 270,270 Z**	We draw the mouth of the smiley here. First we move to 230,270. Q and the following four parameters describe a quadratic bezier curve. Z tells the browser to close the loop.
6		**M230,235 CIR10**	
7		**M270,235 CIR10**	
8		**RTS**	**R**e**t**urn **S**ubroutine
9		**#"main"**	#"main" indicates where execution of code starts, just as in C or Java.
10		**M0,0 F"ORANGE"** **RCT500,500 SC"GREEN"**	Drawing orange background (not shown on front page)
11		**SW4**	SW4 means stroke-width is 4 pixels.
12		**$D=1.05 $N=60 $W=0**	D is distance. N is iterations and W is angle
13		**#"loop"**	Start of loop
14		**ROT250,250,$W**	All coordinates are rotated an angle W around the point 250,250.
15		**SCL250,250,$D**	All coordinates are scaled from perspective point 250,250 to a distance of D.
16		**TRL190,0**	All coordinates are translated 190 pixels in x-direction and 0 pixels in y-direction.
17		**JSR"smiley"**	**J**ump **S**ub**R**outine. Draw the smiley.
18		**POP POP POP**	"Popping" the transformations ROT,SCL and TRL
19		**$N--**	Decrease iteration count N by one
20		**$W+=40**	Increase angle of rotation by 40 degrees
21		**$D*=1.11**	Increase distance from viewer by 11%
22		**BNE"loop",$N**	Branch to "loop" if N is not zero. Code exits here.

Background

The SvgScript interpreter converts its script into SVG-tags that the browser can understand. When doing this it tends to compress most instructions for rectangles, ellipses (circles) into the path tag.

Svg-Tags

Most browsers support the use of SVG tags that can be embedded in HTML. The following code can be embedded inside an HTML document's body tag:

```
<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" stroke="green"
stroke-width="4" fill="yellow" />
</svg>
```

Similar tags can be written for rectangles, polygons, ellipses. However if you want to write a triangle or perhaps rotate your rectangle or ellipse you find that you will have to resort to using the more general purpose tag called path. The following code draws a black triangle:

```
<svg height="210" width="400">
  <path d="M150 0 L75 200 L225 200 Z" />
</svg>
```

The code inside the d attribute defines the triangle inside an svg image of width 210 and height 400.

The following commands are available for path data:

- M = moveto
- L = lineto
- H = horizontal lineto
- V = vertical lineto
- C = curveto
- S = smooth curveto

- Q = quadratic Bézier curve
- T = smooth spline curve
- A = elliptical Arc
- Z = closepath

As stated on W3Schools site: “All of the commands above can also be expressed with lower letters. Capital letters means absolutely positioned, lower cases means relatively positioned.”.

The above commands have doubtless been inspired by PostScript (http://paulbourke.net/dataformats/postscript/) where for example the commands moveto or lineto are aliased using M or L. We find PostScript is used in PDF files for graphics.

Why SvgScript?

I wanted to explore how far down the path of terseness one could go without losing semantics or readability.

Brevity is important because there will always be applications where memory is scarce. Also, brevity means that the coder writes and reads less code.

It should be considered that when we make languages verbose this also makes the code less readable. In this rant verbosity has been driven to extremes, http://steve-yegge.blogspot.com/2006/03/execution-in-kingdom-of-nouns.html :

For the lack of a nail,

throw new HorseshoeNailNotFoundException("no nails!");

There are more verbose examples. The ideologues behind these constructions do not like context based solutions. For example instead of (x, y) for denoting a coordinate they write (circleBehindWheelCoordinateX, circleBehindWheelCoordinateY). That way these variables will be context independent and can never be confused with any other coordinate variable pair, but for the common person readability is reduced.

SvgScript is eclectic and *ad hoc*. Some sources of inspiration are listed below.

6502 assembler

- jsr, jump subroutine
- rts, return subroutine
- bne, branch if not equal to zero
- rzr, return if zero - my own invention

OpenGL

A pipeline of matrix operations for transforming coordinates

- TRL, translation
- REF, reflection
- ROT, rotation
- SCL, scale
- POP removes transformation from pipeline

Perl

Using $ in front of the variables simplified parsing and also prevented overlap between variable and path commands. For example the variable $z and the path command z.

SVG

Svg is written in verbose XML style and can be hard on the fingers when typing. More complex creations may require a large amount of text that perhaps should be produced by another program. If you want to transform certain tags you need to embed them in a transform SVG tag as in the example below or use CSS transform:

```
<svg viewBox="-40 0 150 100"
xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
  <g fill="grey" transform="rotate(-10 50 100)
    translate(-36 45.5) skewX(40)scale(1 0.5)">
    <path id="heart" d="M 10,30 A 20,20 0,0,1 50,30 A 20,20
0,0,1 90,30 Q 90,60 50,90 Q 10,60 10,30 z" />
  </g>
  <use xlink:href="#heart" fill="none" stroke="red"/>
```

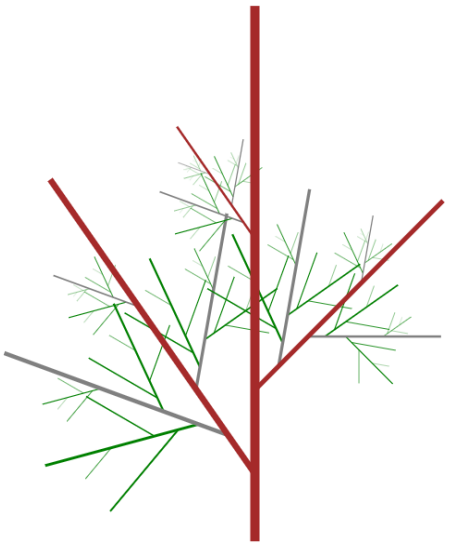
```
</svg>
```

Implementation

SvgScript is coded in javascript and makes heavy use of regular expressions and templates. Appendix contains a list of keywords (lexemes) used in SvgScript.

Fractal tree example

This example motivated the implementation of function calls with arguments and functionality for declaring and invoking arrays. The instruction RZR was needed in order to break and return if depth parameter d equaled zero. Here we also see locally and globally scoped variables. The former are lowercase and the latter are upper case. We see that before each function call to drawTree we load the transformation pipeline with scaling and rotation instructions. These are “POPPED” after the call. The parameters for scale and rotation were found by trial and error. The scale (SCL) is supposed to aid perspective drawings by reducing stroke width proportionally. The fractal idea is to draw a line and then add three branching trees at specific angles. The operation is continued for each branch.



```
SVG500,500
#"drawTree"($d)
RZR,$d
$d--
ROT250,422,-35 SCL250,300,1.5
JSR"drawTree"($d) POP POP
ROT250,350,45 SCL250,220,2
JSR"drawTree"($d) POP POP
ROT250,220,-35 SCL250,130,4
JSR"drawTree"($d) POP POP
SC $COLS[$d]
M250,480 L250,20
RTS
#"main"
$COLS=["green","green","gray","brown","brown"]
SW8
JSR"drawTree"(5)
```

Appendix. Lexemes.

	Keyword / regular expression	
	SVG	SVG width, height
	ID	ID
	#	comment or address position
	".*?"	string
	\+\+	++, increment operator
	\+=	+=, add operator
	-=	subtract operator
	--	decrement operator
	-?(\d+\.\d+|\d+)	a number including negative and decimals
	*=	multiplication operator
	/=	division operator
	=	assign operator
	\(| \)	left or right parenthesis
	\$[a-zääö]+[0-9]*	Local scope variable. Lower case with optional number suffix, for example $z12
	\$[A-ZÅÄÖ]+[0-9]*	Global scope variable. Upper case with optional number suffix, for example $Z5
	TXT	Text command. E.g. TXT”HELLO WORLD”
	TRL	TRansLation. E.g. TRL 10,10
	SCL	SCaLe. E.g. SCL 100,100,2 means perspective point is 100,100 and distance is 2. Compared to distance 1 this means that object at coordinate 100,100 is halved in size.
	REF	REFlection. E.g. REF0,0,0,100 reflects objects vis a vis y-axis. 0,0,0,100 denotes a vector from 0,0 to 0,100.
	ROT	ROTation. E.g. ROT100,100,20 Rotates image 20 degrees around point 100,100
	POP	Removes last transformation (TRL,SCL, REF, ROT) from transformation stack
	SHIFT	Removes first transformation from transformation stack

	JSR	Jump SubRoutine
	RTS	ReTurn from Subroutine
	RZR	Return if ZeRo
	BNE	Branch if Not Equal zero
	T[ra][ra]	Triangle. E.g.
	RCT	ReCTangle
	CIR	CIRcle
	ELL	ELLipse
	ARC	ARC
	SC	Stroke Color
	SW	Stroke Width
	F	Fill Color
		The following commands have been copied from https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute/d#path_commands
	m, M	m: relative move, M: absolute move
	l, L	l, lineTo relative coordinates L, lineTo absolute coordinates
	t, T, c, C, q, Q	Bezier commands
	a, A	Arc
	z, Z	close path
	s, S	smooth spline