ArtScript Semantics

Andrew Ansah Henok Misgina Fisseha

May 20, 2024

Presentation Video

Introduction

ArtScript, designed specifically for creating geometrically intricate artwork, particularly focusing on the use of polygons and points on a coordinate plane. ArtScript blends the elegance of mathematical shapes with the creative expression of visual art. ArtScript provides built-in functions for drawing basic shapes: like lines, circles, and polygon and empowers artists to encapsulate complex patterns into reusable functions. It's a playground for those who want to explore the beauty of geometry through code.

ArtScript serves as an excellent introduction to programming for aspiring artists and kids, offering a simplified syntax and intuitive commands for creating visually stunning geometric art and generative patterns. Its accessibility and ease of learning make it an ideal platform for beginners to explore the fundamentals of coding while unleashing their creativity through digital art. ArtScript will also encourage original artwork creation by users

Design Principles

ArtScript draws inspiration from the elegant simplicity and geometric artwork found in the Williams College Museum of Art. Inspired by these aesthetic principles, ArtScript is designed to replicate and extend upon these artistic styles through a programming language tailored for creating visually captivating geometric art and patterns. By providing intuitive commands and support for iterative patterns, ArtScript empowers artists to explore and express their creativity in the digital realm. We hope to be able to create crude replications of some of the artwork in the museum. The design is mainly based on using a pen to draw geometric shapes. ArtScript encourages freestyling with art and multiple renderings or similar renderings of an artwork can be easily achieved through its combined forms.

Language Concepts

Artscript is a specialized programming language designed to simulate drawing with a pen with mathematical precision. Users interact with Artscript through a set of 10 core commands: Forward, SetLocation, TurnRight, TurnLeft, Shift, Penup, Pendown, Rect, Circle, and Polygon. These commands provide the fundamental building blocks for creating a wide variety of shapes and patterns. With a basic understanding of geometry, users can effectively utilize these commands to produce complex drawings. Each command controls a specific aspect of the drawing process, whether it's moving the pen, rotating it, or drawing geometric shapes.

One of the most powerful features of Artscript is the repeat function, which allows users to execute a series of commands multiple times. This feature significantly reduces the amount of code needed to create intricate designs and patterns. By leveraging the repeat function, users can easily produce complex and repetitive shapes with minimal effort. Overall, Artscript combines mathematical accuracy with a user-friendly set of commands, making it an ideal tool for creating detailed and precise drawings programmatically.

Formal Syntax

⟨*expr*⟩ ::= ⟨*command*⟩ ⟨*expr*⟩ | ⟨*command*⟩ | ⟨*expr*⟩ ⟨*repeat*⟩ ⟨*expr*⟩ | ⟨*empty*⟩

⟨*repeat*⟩ ::= repeat ⟨*n*⟩ (⟨*expr*⟩)

⟨*direction*⟩ ::= up | down | left | right

⟨*color*⟩ ::= red | green | blue | purple | black | yellow | gold | white | pink | brown | orange | RGB(⟨*n*⟩ ⟨*n*⟩ ⟨*n*⟩) | none

⟨*num_pair*⟩ ::= ⟨*n*⟩ ⟨*n*⟩

⟨*command*⟩ ::= ⟨*command*⟩ ⟨*expr*⟩ | ⟨*command*⟩ | ⟨*empty*⟩
| go ⟨*n*⟩ ⟨*color*⟩
| setlocation ⟨*n*⟩ ⟨*n*⟩
| toright
| toleft
| penup
| pendown
| shift ⟨*n*⟩ ⟨*direction*⟩
| rect ⟨*n*⟩ ⟨*n*⟩ ⟨*color*⟩ ⟨*color*⟩
| circle ⟨*n*⟩ ⟨*color*⟩ ⟨*color*⟩
| poly ⟨*color*⟩ ⟨*color*⟩ ⟨*num_pair*⟩*

⟨*n*⟩ ::= (any positive integer)

Semantics

The program currently works by leveraging two facts. The fact that we can draw like a pen by following a point in any direction. And the fact that we can create polygons anywhere on the canvas by specifying their location, color, and size. Our primary primitives are commands which can be excecuted in order. The combining form is a expr which is a list of commands including the repeat function. The commands themselves utilize numbers, colors, and directions;however these can't be used outside of commands. The repeat element works by copying a given expression a specified number of times and adding it to the drawing list. Technically, commands are the most powerful because each one serves a specific purpose.

Forward (len, color)

- If the pen is down, draw a line from the current position in the current direction with the specified color.
- Update the pen's position based on the length and direction.
- If the pen is up, move the pen to the new position without drawing.

SetLocation (x, y)

Move the pen to the specified (x, y) coordinates without drawing.

TurnRight

Change the pen's direction 90 degrees clockwise.

TurnLeft

Change the pen's direction 90 degrees counterclockwise.

Shift (len, dir)

Move the pen in the specified direction by the specified length without drawing.

Penup

Set the pen's state to up, preventing it from drawing when moved.

Pendown

Set the pen's state to down, allowing it to draw when moved.

Rect (w, l, fill, color)

Draw a rectangle at the current pen position with the specified width, height, fill color, and stroke color.

Circle (r, fill, color)

Draw a circle at the current pen position with the specified radius, fill color, and stroke color.

Polygon (fill, color, coords)

Draw a polygon with the specified fill color and stroke color using the provided list of coordinates.

Semantics Table

Syntax	Abstract Syntax	Type	Prec./Assoc	Meaning
n	num of int	int	N/A	n is a primitive. We represent integers using the 32-bir F# integer data type $(int32)$
color	color of line	str	N/A	Color is the wide range of colors a user can use to draw lines. The user can also specify his own RGB values.
Red	color of Red	str	N/A	red is the primitve that will represent the color red for a line. Similar commands can be carried out for the wide range of colors
RGB	color of RGB	num*num*num	N/A	rgb allows the user to be as specific as possible about their choice of colors. users are given the choice to input their own rgb values
expr	Expr List	str	N/A	expr is the series of commands to be carried out by the language
repeat	Expr List	str.	N/A	repeat repeats expr by n times
direction	direction	North | South. East | West	N/A	This defines a new type named direction, which can take one of four possible values: North, South, West, or East Each of these values represents a distinct direction, and no other values can be a Direction aside from these four.
command	command List	**Forward** of (Int*Color), **TurnRight**, **TurnLeft**, **Shift** of (Int*Direction), **Rect** of (Int*Int*Color*Color), **Circle** of (Int*Color*Color), **Penup**, **Pendown** **Polygon** **Setlocation**	1 / Left	Series of commands to be carried out Types of turns of the pen forward movement of the pen setting location of the pen draw a circle,rect or a polygon Move pen up Move pen down draw a polygon set the location of the pen

Remaining works

If we had more time we would add variables, recursion trees, and for loops. A feature to help the user locate and designate the coordinates for the pen will be helpful. We would also add more shapes like ellipses. Text support would also be implemented. It would be nice to have text on the canvas. Transformations of shapes would be implemented. It would be nice to rotate, mirror, etc shapes.

Examples

There are example inputs provided in the ”code/examples” folder in the repository. Refer to the end of this document for more visuals.

```
setlocation 10 10
rect 200 200 yellow yellow
setlocation 50 50
rect 100 100 gold gold
setlocation 70 70
rect 50 50 none yellow
```

Figure 1: The name of file in examples folder is rect.txt and the output is rect.svg. The above lines of commands saved as a txt file when run with dotnet run rect.txt outputs below:

Figure 2: Joself Albers similar artwork example

```
circle 100 RGB(100 100 100) red
```

Figure 3: The name of file in examples folder is circle.txt and the output is circle.svg. The above line of commands saved as a txt file when run with dotnet run circle.txt outputs below:

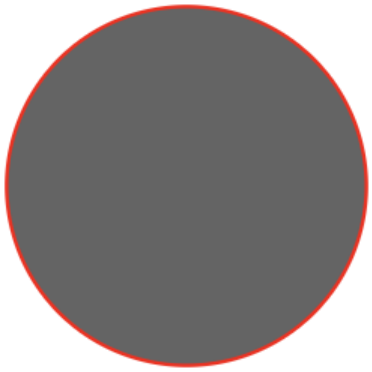

Figure 4: circle

```
setlocation 20 380

repeat 9 (

go 360 blue
toright
go 10 green
toright
go 360 purple
toleft
go 10 red
toleft

go 360 green
toright
go 10 blue
toright
go 360 red
toleft
go 10 purple
toleft )
```

Figure 5: The name of file in examples folder is example.txt and the output is example.svg. The above line of commands saved as a txt file when run with dotnet run example.txt outputs below:

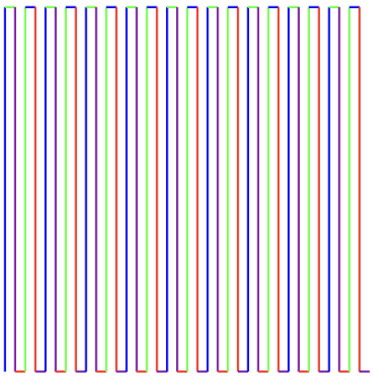

Figure 6: repeating pattern

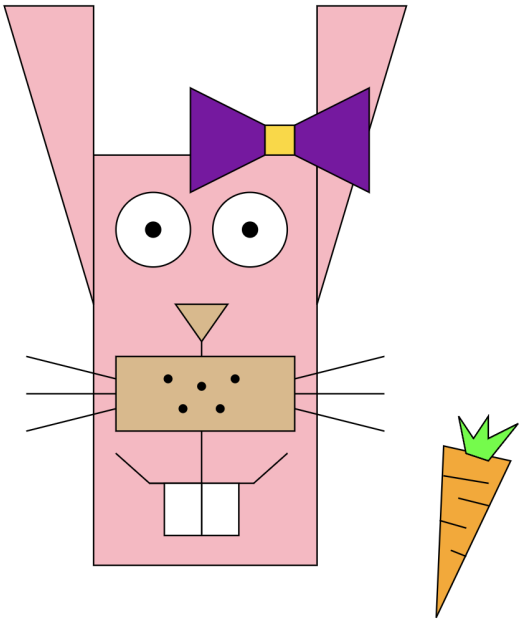

Figure 7: Image of Bunny using ArtScript