Language Specifications

Sammy Sasaki, Mico Mendoza

0.1 Introduction

Geometric abstraction is an abstract art form that primarily uses simple geometric shapes painted with flat colors and placed in non-illusionistic planar space. Many people enjoy geometric abstract art, but sometimes they can be hard to create or find, especially for people with little to no artistic skills.
This is a domain specific programming language that offers users a way to easily create geometric abstract art. It lets users create art that is customized to their own liking, in terms of color scheme, size, shapes, and other parameters.

0.2 Design Principles

The main guiding principles in this programming language are simplicity and usability. We want our language to be simple and easy to use, and give the user as much power as possible to create the art that they are envisioning for themselves. We designed our language so that the users have the power to easily customize the primary visual features of a geometric abstract art – the 2D geometirc shapes and colors/color scheme. Other visual features that we are keeping in mind as we develop our language further are the number of edges for each shape, the relative sizes of the shape, the positions of the shape, the colors of adjacent shapes, the size of the canvas that we are filling up with shapes (and gaps between shapes), etc.

0.3 Examples

```
5 shapes, scheme: greyscale, max 8 edges
```

This should generate 5 shapes on a canvas colored shades of grey, each with between 3-8 edges.

```
7 shapes, scheme: rainbow, max 6 edges
```

This should generate 7 shapes on a canvas with colors from the rainbow, each with between 3-6 edges.

```
2 shapes, scheme: red green blue purple , max 3 edges
```

This should generate 2 shapes on a canvas, each colored one of red green blue, or purple, each with 3 edges.

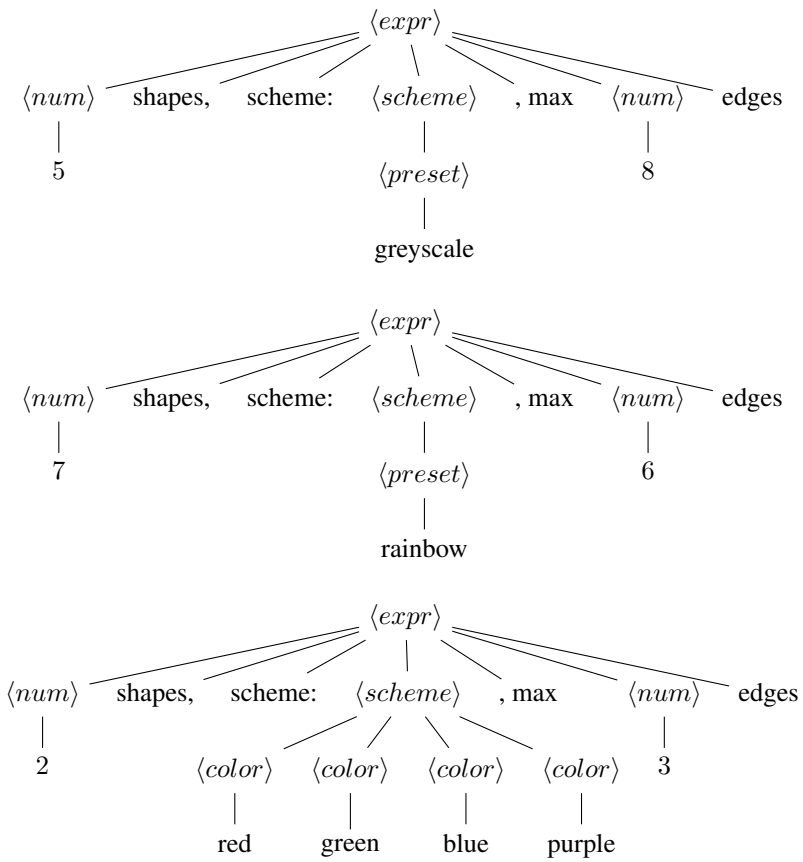
0.4 Language Concepts

Each program takes 3 parameters to specify the geometric abstract art a programmer wants to create. The number of shapes and the number of maximum edges for any given shape are primitive types (int) in this language, while the color scheme is a combined form of the primitive type color (string), where a programmer can choose from preset color schemes or customize their own (must be 4 or more distinct colors).

0.5 Formal Syntax

```
<expr>   ::= <num> shapes, scheme: <scheme>, max <num> edges
<num>    ::= <digit>+
<digit>  ::= 0 | ... | 9
<scheme> ::= <preset>
           | <color>4+
<color>  ::= red | orange | yellow | green | blue | indigo | purple
<preset> ::= greyscale | rainbow
```

Example ASTs:



0.6 Semantics

(i) num has an abstract syntax of Num of int, and is an int, represented by Int32.

(ii) Color is abstract data type which can be one of many predetermined color options. This is used to specify the colors of shapes and the background.

(iii) Preset is an abstract data type which currently has two options: greyscale or rainbow. This is one option for specifying the colors used in a particular artwork.

(iv) Scheme is an abstract data type of either a Preset of Preset, or a Colors of Color List, a list of 4 or more colors. This determines the color scheme of an artwork.

(v) Expr has an abstract syntax of Expression of int * Scheme * int. This encapsulates one piece of art, specifying the number of shapes, the colors to be used, and the max number of edges per shape. It has precedence 1 and is left associative.

(vi) The programs don't take any user input outside of the original program lines. The programs output svg files that when opened display a colored canvas containing the art piece. The evaluation would be done using svg. First a string of svg commands would be created when evaluating the tree, and that would be fed into a file which can be opened.

0.7 Limitations/Remaining Work

Here are limitations/items that need to be worked on further:

1. We have to assign distinct colors for adjacent shapes

2. We would like to have a better way of generating shapes, perhaps using a forbidden zone or growing the shapes from each other, the make them more spread out and similar to our original vision.

3. We are considering adding more customization options for the program writer, perhaps regarding edge lengths, or the ability to explicitly add shapes in certain positions.