# gTangle

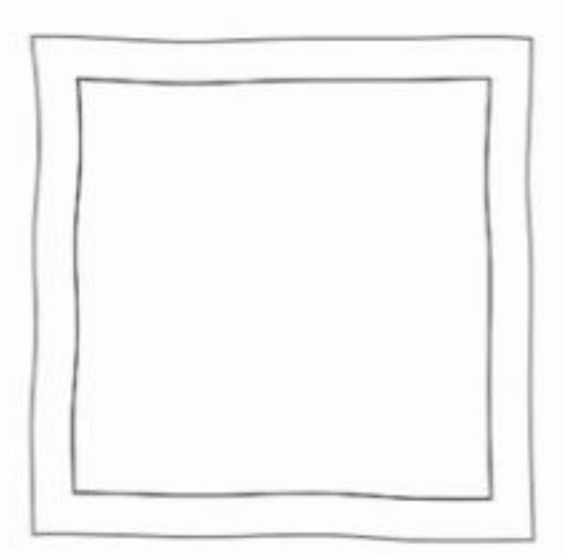
## A Grammar for the Procedural Generation of Tangle Patterns

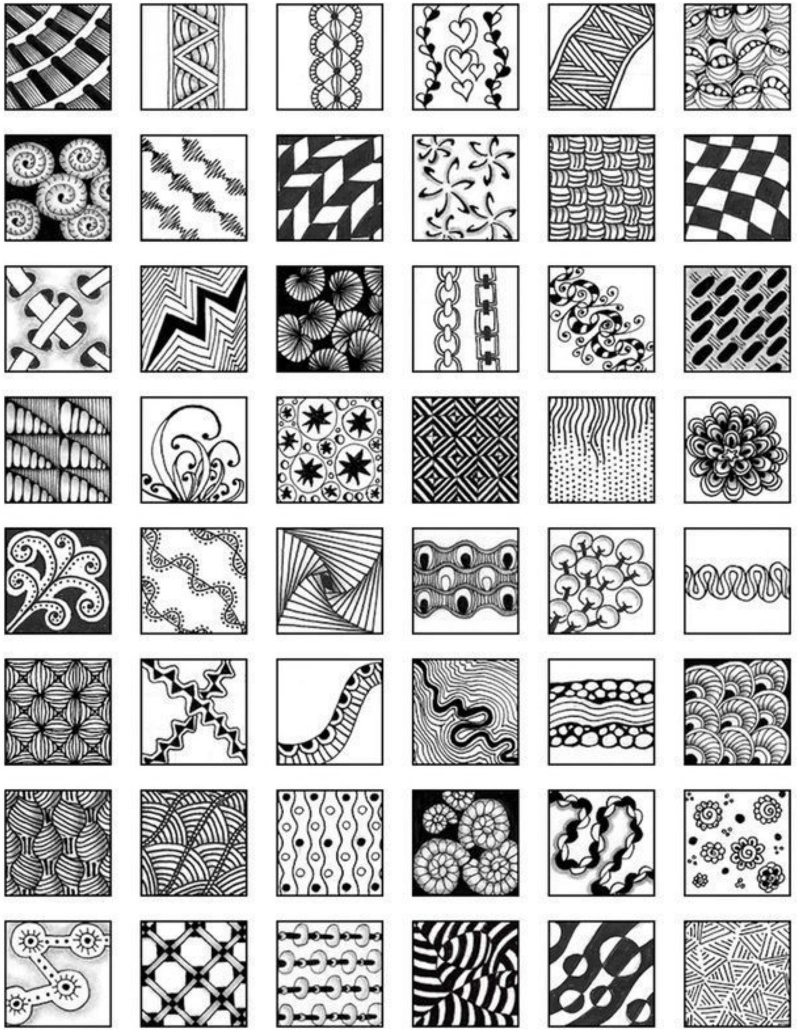
| | |
|---|---|
| **Lab Team** | Aman Goel<br>Ammar Ahmed<br>Aryamaan Jain<br>Jyoti Sunkara |
| **Mentor** | Prajwal Krishna Maitin |

Tangles are a form of two dimensional structured pen and ink art created by a small set of basic strokes:

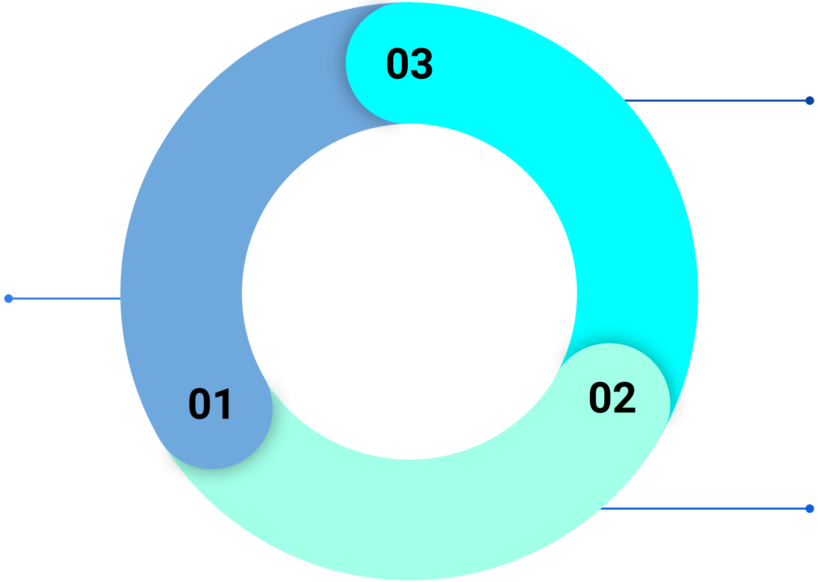
- Dots
- Straight Lines
- Curves
- Shapes

The tangles are generated by recursively splitting and combining initial set of polygons, using group grammars that perform operations on the polygons.





# Purpose



## Group Grammars Simplicity

Expressing complex tangles patterns and sub-pattern distributions, with relatively simple grammars.

## Artistry

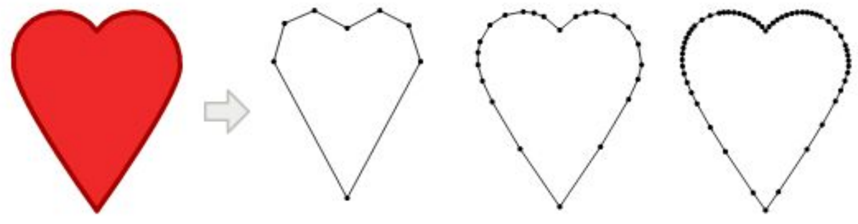
Tangles take artists hours of tedious and time-consuming work which can now happen quicker and with fine control.

## Speed

Group grammars can, in few tens of seconds, produce a wide variety of patterns!

# Grammar

## Shape

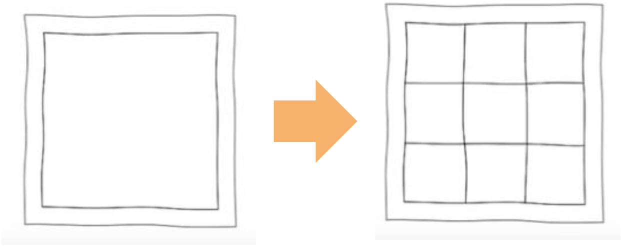


*Geometric Shape*

$$S = \langle t, g, s; \Theta, \mathbf{d} \rangle$$

*Grammar Shape*

## Rules



*Regular Split Rule*

$$R = O(\{p_o\}) : t_m \rightarrow [\langle t_0, g_0 \rangle, \ldots, \langle t_k, g_k \rangle]$$

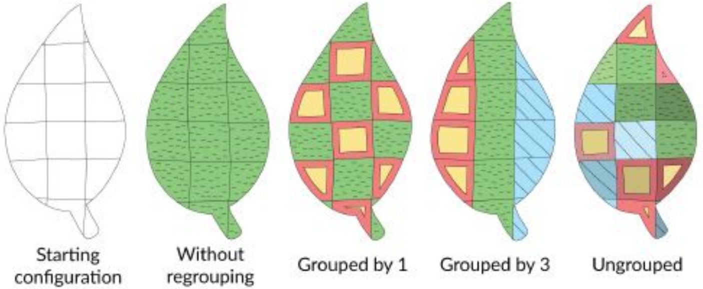
*General Production Rule*

# Operators

## Grouping Operators

$$ungroup() : t_m \rightarrow [\langle t', g_0 \rangle, \langle t', g_1 \rangle \ldots \langle t', g_k \rangle]$$

$$regroup(k) : t_m \rightarrow [\langle t', g_0 \rangle \ldots \langle t', g_0 \rangle \ldots \langle t', g_k \rangle \ldots \langle t', g_k \rangle]$$



*Grouping Operators Geometry*

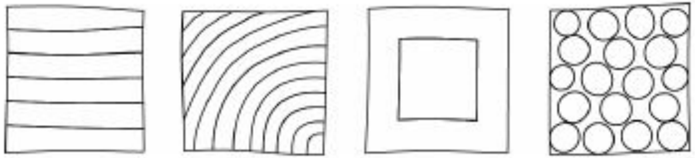
## Geometric Operators

$$regularSplit(c, o, x) : t_m \rightarrow [\langle t_0, g_0 \rangle, \ldots, \langle t_0, g_0 \rangle]$$

$$streamlineSplit(type, o) : t_m \rightarrow [\langle t_0, g_0 \rangle, \ldots, \langle t_0, g_0 \rangle]$$

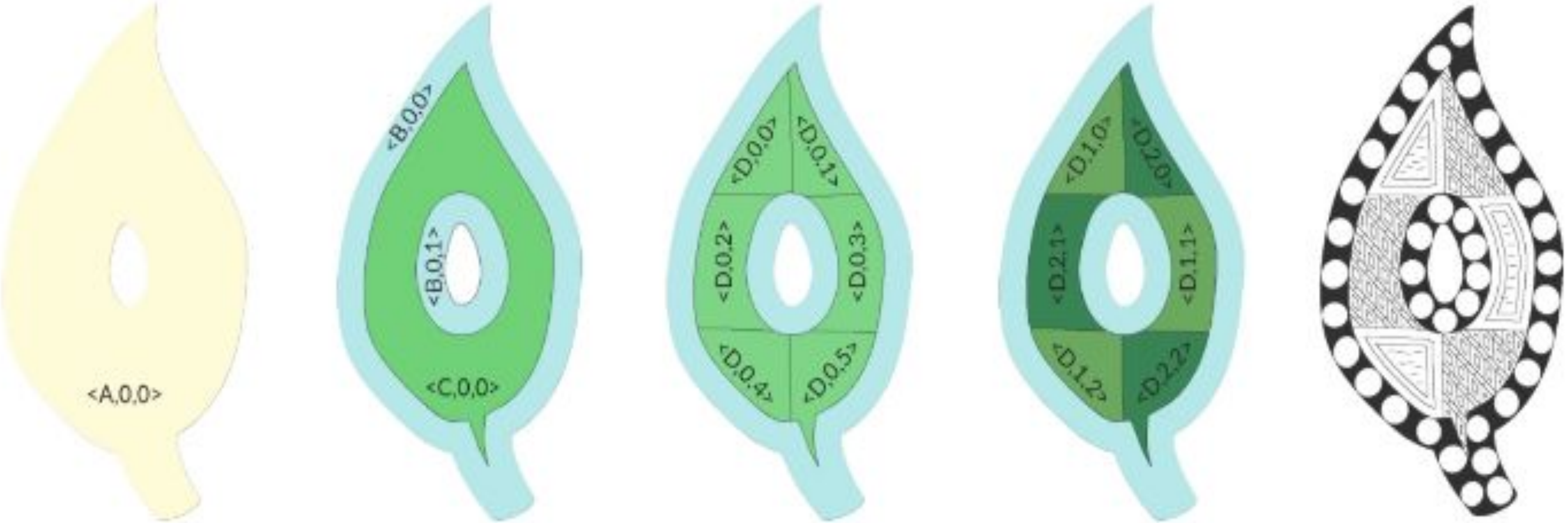
$$outline(d) : t_m \rightarrow [\langle t_{out}, g_0 \rangle, \langle t_{in}, g_1 \rangle, \ldots, \langle t_{in}, g_1 \rangle]$$

$$place(s, d) : t_m \rightarrow [\langle t_{rem}, g_{rem} \rangle, \langle t_0, g_0 \rangle, \ldots, \langle t_0, g_0 \rangle]$$



*Geometric Operators Geometry*

# Recursive Operators



$R_a$ = outline : A -> [B, C]

$R_b$ = regularSplit: C -> D

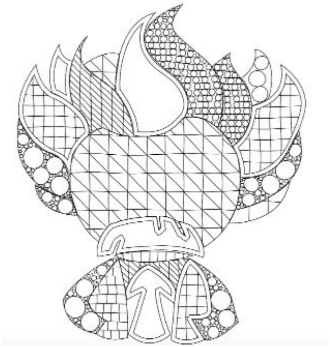
$R_c$ = regroup : D

Final result

# Iteration

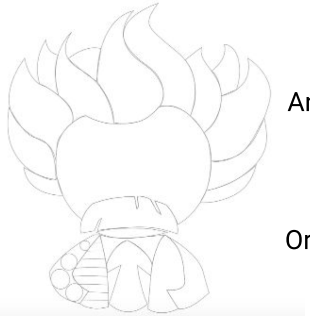
**Shape Perturbation**

A coherent noise function that is applied to all shapes with the same group id



**Grouping And Geometric Operators**

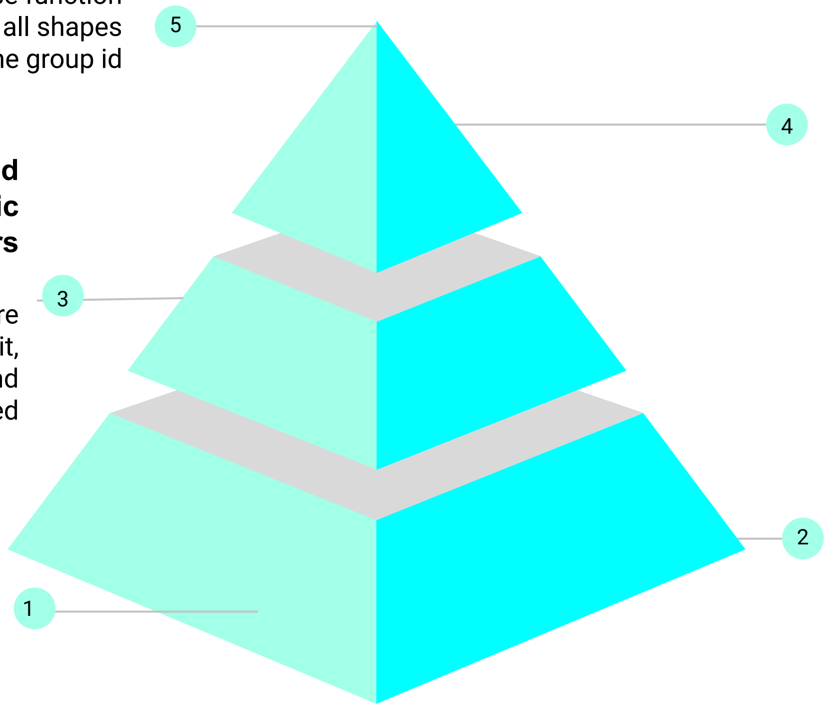
The shapes are grouped, split, placed and outlined



**Input**

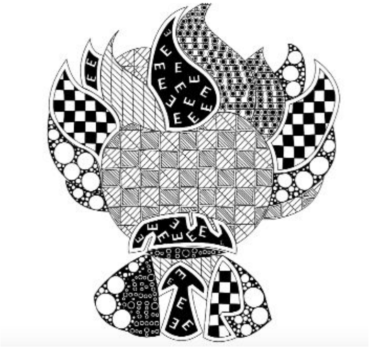
An outline sketch in SVG format is provided as input

Or an SVG outline is created by us.



**Decorative Operators**

The shapes are filled and stippled

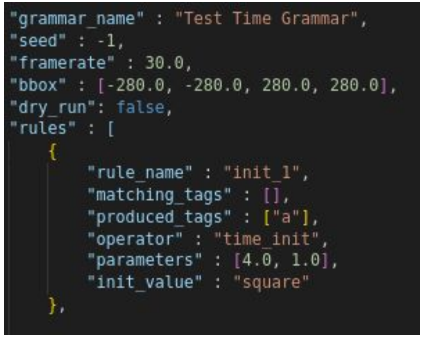


**Tree**

The SVG is stored as a tree data structure after being bisected into the elemental structures

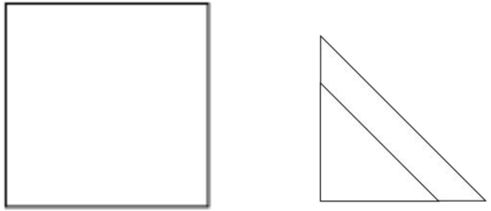
# Results

## JSON

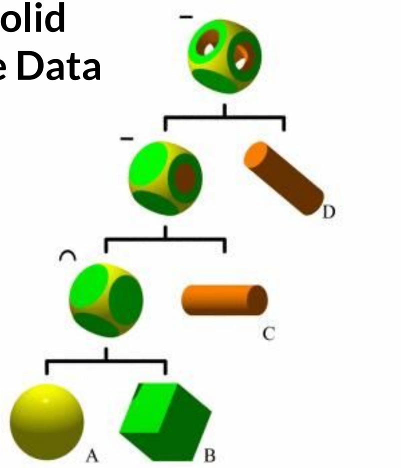


```
"grammar_name" : "Test Time Grammar",
"seed" : -1,
"framerate" : 30.0,
"bbox" : [-280.0, -280.0, 280.0, 280.0],
"dry_run": false,
"rules" : [
    {
        "rule_name" : "init_1",
        "matching_tags" : [],
        "produced_tags" : ["a"],
        "operator" : "time_init",
        "parameters" : [4.0, 1.0],
        "init_value" : "square"
    },
```

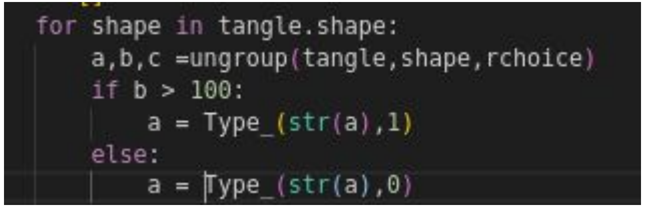
## SVG Drawing



## Constructive Solid Geometry Tree Data Structure



## Grouping Operators



```
for shape in tangle.shape:
    a,b,c =ungroup(tangle,shape,rchoice)
    if b > 100:
        a = Type_(str(a),1)
    else:
        a = Type_(str(a),0)
```

# Challenges

*Challenge 1*

### Collaboration

With the online semester underway the difficulty of communication, debugging, pair programming and explaining code has become manifold.

*Challenge 2*

### SVG Graphics

SVG defines the graphics in XML format.

The intricacies of SVG images, how they are drawn, stored structured and created and how to create them from code are all new problems for us.

*Challenge 3*

### Constructive Solid Geometry Trees

While grammars bring mathematical finesse to the tangles on paper. The representation and manipulation  of shapes using a graph in code is a hard problem to solve.

# Progress

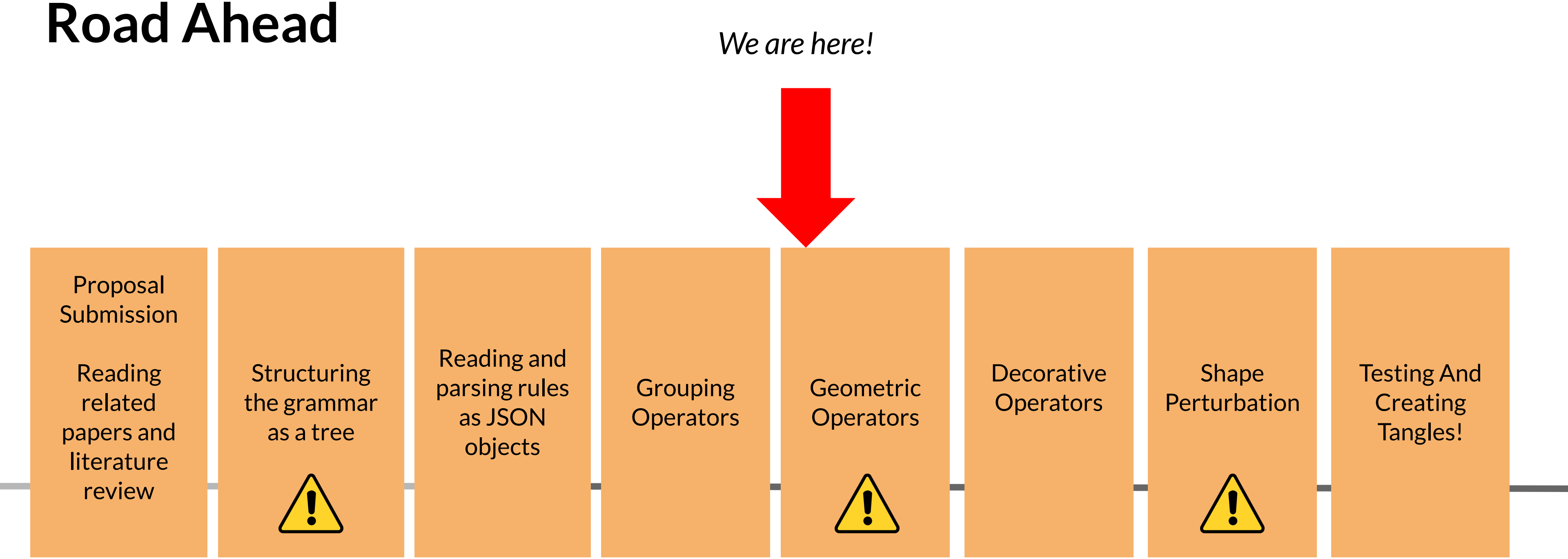
- Grammar tree data structure to store shapes and take tessellated polygons as input and output in SVG format
- Structuring rules as JSON files and parsing them
- Grouping Operators

Overall:

43%

Completed! 🎉

# Road Ahead



*We are here!*

Proposal Submission

Reading related papers and literature review

Structuring the grammar as a tree ⚠

Reading and parsing rules as JSON objects

Grouping Operators

Geometric Operators ⚠

Decorative Operators

Shape Perturbation ⚠

Testing And Creating Tangles!