# Lindenmayer Systems in Shape Machine

**Zhen Li**
Georgia Institute of Technology, United States
zhenli.craig@gatech.edu

This project experiments with a certain type of Lindenmayer Systems (L-Systems) within Shape Machine to expand shape grammars' visual computational capabilities. L-Systems are well-suited for generating recursive, fractal-like structures. By bringing L-Systems into Shape Machine, this project aims to enable recursive shape-based rule applica-tions within a CAD environment, creating opportunities for intricate patterning in architectural, industrial, and visual design. Specifically, this project explores programmatically generating DrawScripts from L-Systems rules and visualizes the resulting designs. The integration of L-Systems in Shape Machine offers a novel approach to shape grammar design, combining the generative power of L-Systems with the formalism of shape grammars to create complex, visually engaging designs.

## 1 Introduction

Lindenmayer Systems (or L-Systems) are a mathematical formalism introduced by biologist Aristid Lindenmayer in 1968. [1], [2] They were initially developed to model the growth of plant structures but have since found applications in computer graphics, particularly in generating fractals and procedural natural forms.

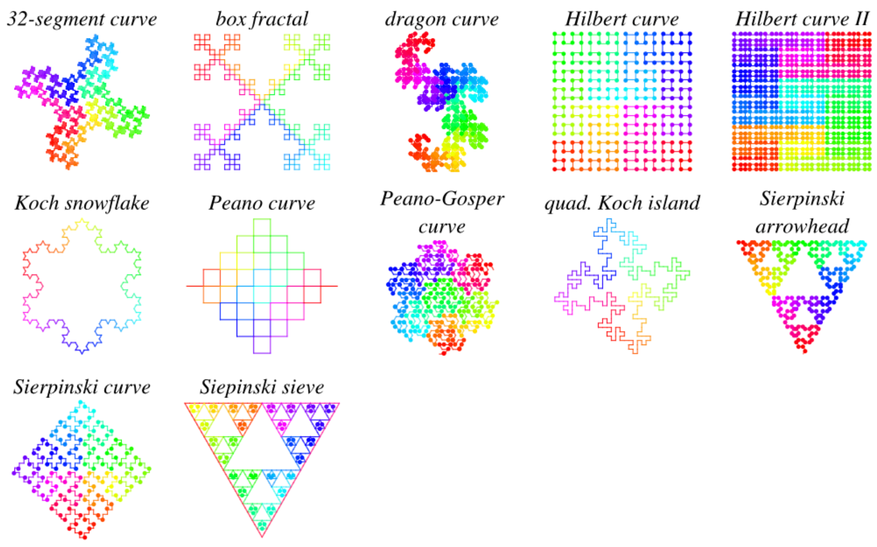


**Fig. 1** Fractal geometry produced by Lindenmayer systems

L-Systems operate as a set of rewriting rules (also known as gram-mar), where symbols in a string are replaced iteratively according to predefined rules.[2] These systems consist of three main components:

• ***Alphabet***: A set of symbols used to construct strings.

• ***Axiom***: An initial string or starting point.

• ***Production Rules***: A set of rules dictating how symbols are replaced or transformed in each iteration.

The integration of L-Systems with turtle graphics is a powerful tool in computer graphics, enabling the visualization of complex geometries and patterns. [3] Turtle graphics interprets the symbols of an L-System as commands to draw or manipulate the cursor on a Cartesian plane. [4] Commonly used commands include:

| | |
|---|---|
| `F` | Move forward while drawing a line. |
| `+` | Turn right by a specified angle. |
| `-` | Turn left by a specified angle. |
| `[` | Save the current position and orientation (used for branching structures). |
| `]` | Restore the last saved position and orientation. |
| `\|` | Turn around (180 degrees). |

**Table 1** An action table for L-systems

For example, consider the Koch curve: a string of characters (symbols) is rewritten on each iteration according to some replacement rules. The initial string (axiom)

$$F + F + F + F$$

And a rewriting rule:

$$F \longrightarrow F + F - F - FF + F + F - F$$

After one iteration the following string would result

$$\begin{aligned} &F + F - F - FF + F + F - F + F + F - F - FF + F + F - F \\ &+F + F - F - FF + F + F - F + F + F - F - FF + F + F - F \end{aligned}$$

They could be visualized as below:

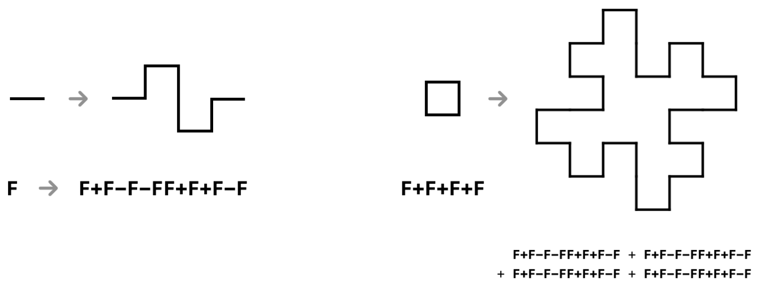

F → F+F−F−FF+F+F−F
F+F+F+F
F+F−F−FF+F+F−F + F+F−F−FF+F+F−F
+ F+F−F−FF+F+F−F + F+F−F−FF+F+F−F

**Fig. 2** Koch curve example

In this project, we will programmatically generate the shape grammars for L-systems, facilitating a richer integration between sophisticated fractal geometry and the ergonomic design environment provided by Shape Machine. [5] Specifically, we will be focusing on a certain type of geometry that can be generated through a unified logic where all the alphabet characters represent moving forward. [6]

## 2 Motivation

The integration of L-Systems into architectural and design workflows offers a transformative approach to generating and visualizing complex geometries. [1], [2] As a procedural fractal geometry generation method, L-Systems provide architects and designers with tools to explore fractal patterns, branching structures, and intricate geometrical forms that are often difficult to conceptualize through traditional means.

However, the sophisticated fractal geometries usually require a precise vector computation and can be difficult for human designer to implement. By leveraging the inherent logic of iterative rewriting and visualizing the results through Shape Machine and automated rule generation, designers can rapidly prototype intricate geometries and experiment with various configurations. This not only enhances creativity but also allows for the exploration of patterns and forms that align with natural principles, leading to more harmonious and functional designs.

## 3 Methodology

In this section, we will discuss the implementation of L-Systems in Shape Machine.

### 3.1 Implementing a state machine for turtle graphics in Shape Machine

In computer graphics, turtle graphics are vector graphics using a relative cursor (the “turtle”) upon a Cartesian plane ($x$ and $y$ axis). The turtle moves with commands that are relative to its own position, such as “move forward

10 units" and "turn left 90 degrees". The pen carried by the turtle can also be controlled, by enabling it, setting its color, or setting its width.

Unlike traditional turtle graphics environment, where a simple line implicitly carries the semantic of direction, a single line alone in Shape Machine does not carry such information. To implement a turtle machine in Shape Machine, the first thing we need to add is a point to represent the direction. For every line we draw a point with the same attributes (color).

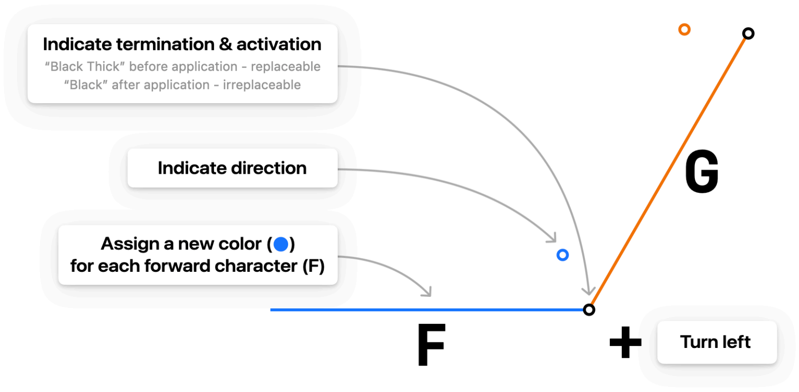

Indicate termination & activation
"Black Thick" before application - replaceable
"Black" after application - irreplaceable
Indicate direction
Assign a new color (●) for each forward character (F)
G
F
+
Turn left

**Fig. 3** Turtle graphics in Shape Machine

Another addition is a point that indicates termination. This is to avoid the unwanted matches as shown by the red point and line in Figure 4. The termination point uses the different colors so that Shape Machine doesn't confuse termination points with direction points.

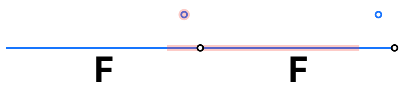


**Fig. 4** Unwanted match

## 3.2 Simulating parallel application in Shape Machine

Some L-System design might contain multiple rules, and they are usually parallelly rewritten. [1], [6] Consider the following rule for dragon curve:

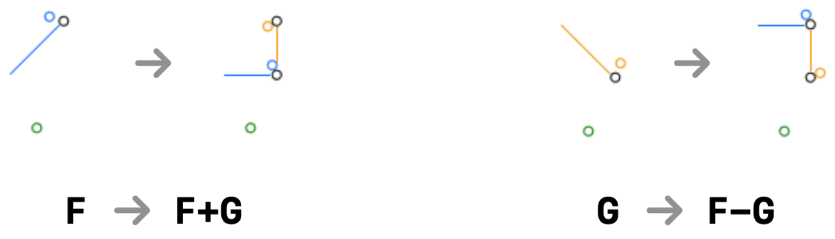


**Fig. 5** Dragon curve rules

For the axiom $F + G$, the L-system should produce $F + G + F - G$ after first application.

However, if it is rewritten sequentially, the resulting string will be $F + G + G$ after the first application of the left-hand side rule ($F \to F + G$), and then $F + F - G + F - G$ after the second application of the right-hand side rule ($G \to F - G$). This is no longer a dragon curve, and the resulting string's semantic would diverge in traditional turtle graphics environment and Shape Machine — In turtle graphics, $F$s should be of same length. However, if we apply the DrawScripts sequentially, the first $F$ will be $\sqrt{2}$ times longer.

To simulate parallel rewriting behavior in Shape Machine, we can borrow some compiler techniques [7] to make Shape Machine “blind” to the intermediate application result – we can rewrite rules to simulate parallel rewriting in a sequential manner, as shown in Figure 6.

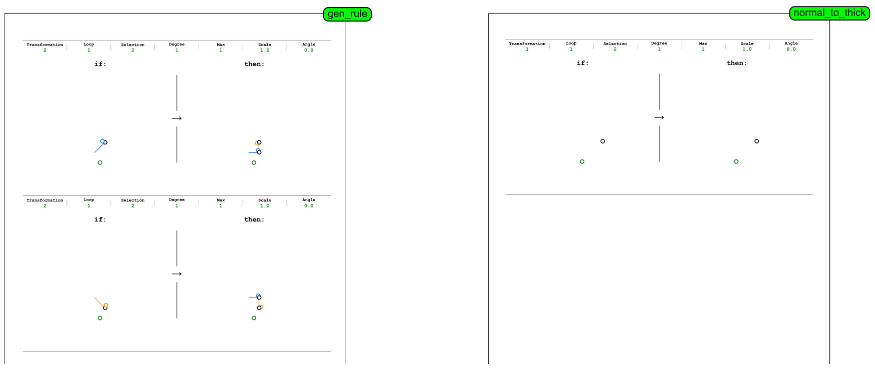
$$\begin{cases} F \to F + G \\ G \to F - G \end{cases} \implies \begin{cases} F \to F' + G' & \boxed{\text{gen\_rule}} \\ G \to F' - G' & \\ & \\ F' \to F & \boxed{\text{normal\_to\_thick}} \\ G' \to G & \end{cases}$$

**Fig. 6** Modified rules

To translate this change into Shape Machine, we can simply change the attribute of the terminating black points. In the current implementation, the terminating black point is activated ( `"Black Thick"` ) in the symbols without prime signs ($F$ and $G$), deactivated ( `"Black"` ) in the symbols with prime signs ($F'$ and $G'$).

Another DrawScripts block (normal_to_thick in Figure 8) for reactivate the terminating points to `"Black Thick"` is then added to the DrawScripts routine. This block will be executed in each iteration after all the generated rules have been executed.

After this change, we can simulate parallel application of multiple rules in Shape Machine.



| Transformation | Loop | Selection | Degree | Max | Scale | Angle |
|---|---|---|---|---|---|---|
| 2 | 1 | 2 | 1 | 1 | 1.0 | 0.0 |

| Transformation | Loop | Selection | Degree | Max | Scale | Angle |
|---|---|---|---|---|---|---|
| 2 | 1 | 2 | 1 | 1 | 1.0 | 0.0 |

| Transformation | Loop | Selection | Degree | Max | Scale | Angle |
|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 1 | 1 | 1.0 | 0.0 |

**Fig. 7** Parallel application in Shape Machine

### 3.3 Automated generation of DrawScripts from given grammar

Now we can generate DrawScripts for simple L-systems. Figure 8 is an overview of the DrawScripts program. To generate the rules, the designer can simply provide the rules and apply the leftmost block lsys-rule-generator to a selection of dark green points inside the gen_rule block, which is provided by Shape Machine templates.

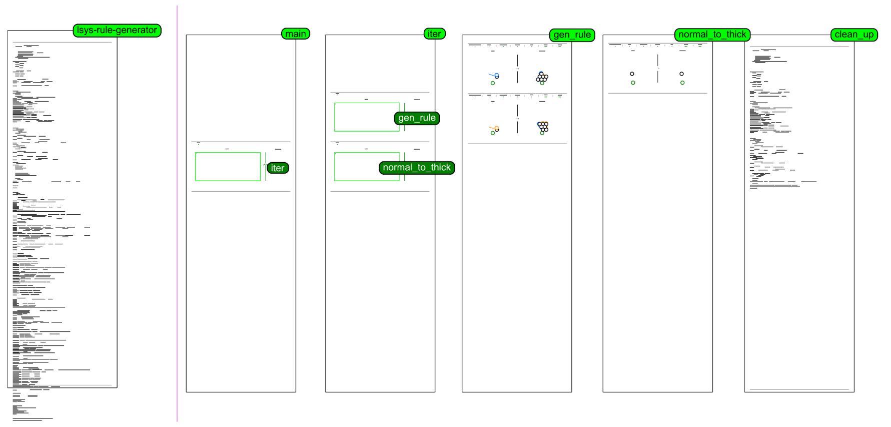


**Fig. 8** The DrawScripts program

## 4 Results

Below are some of the experiments made with the rule generator.

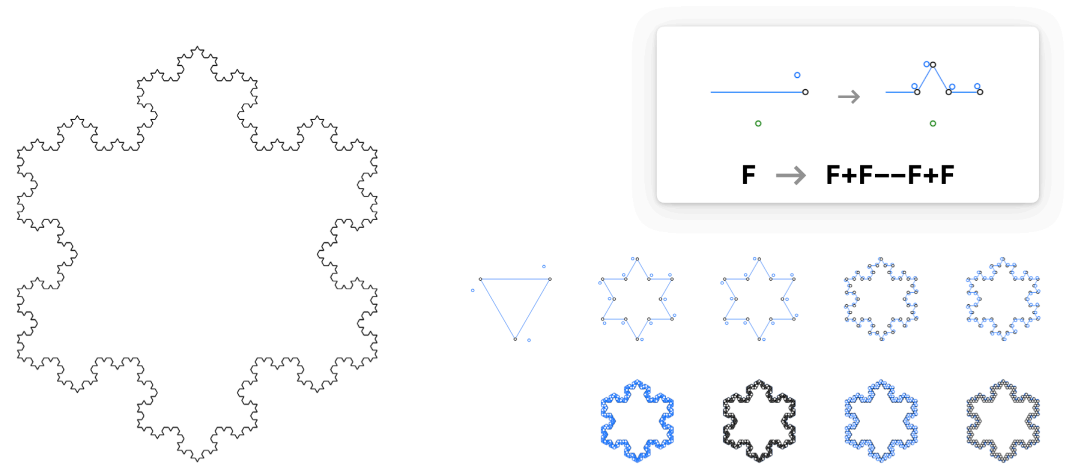


**Fig. 9** Koch snowflake

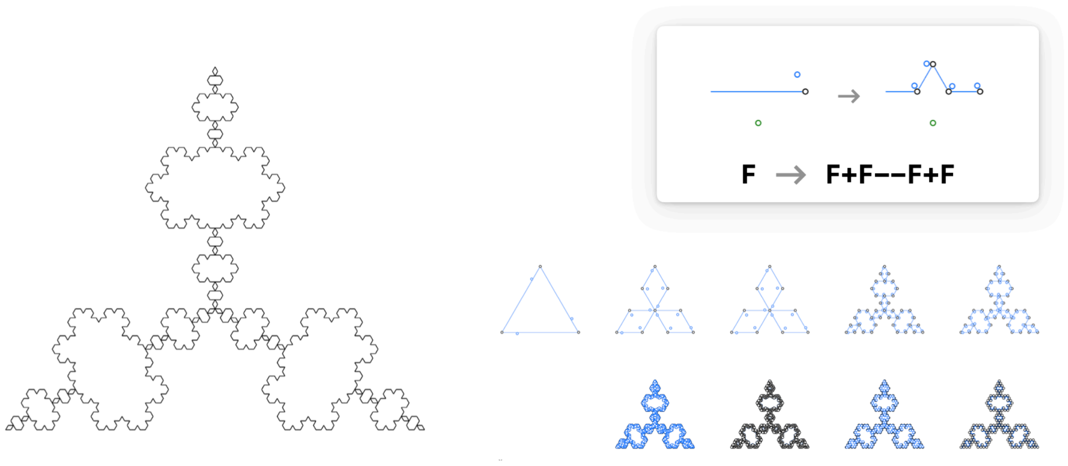


**Fig. 10** Koch anti-snowflake

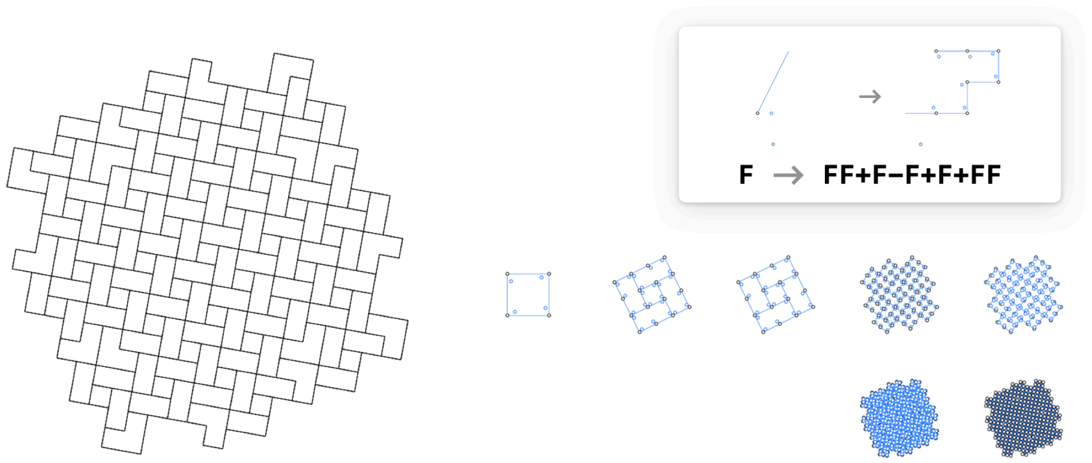


**Fig. 11** Tiles

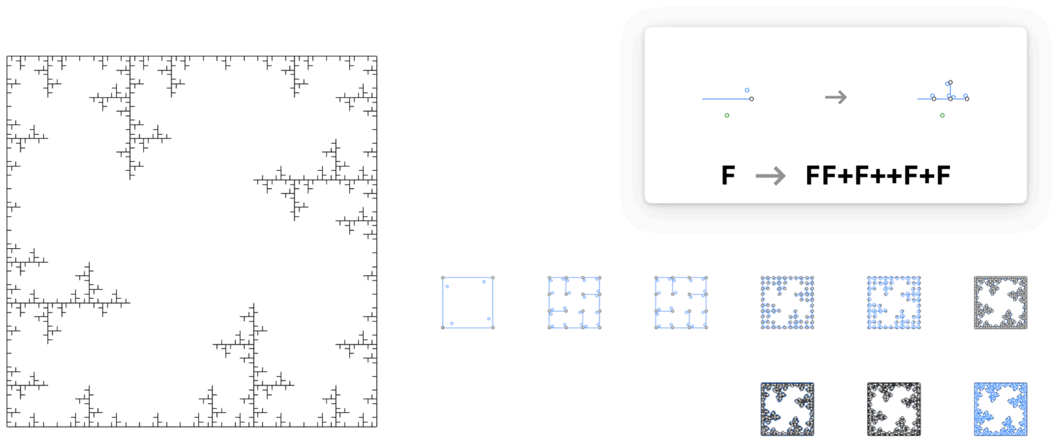


**Fig. 12** Crystal

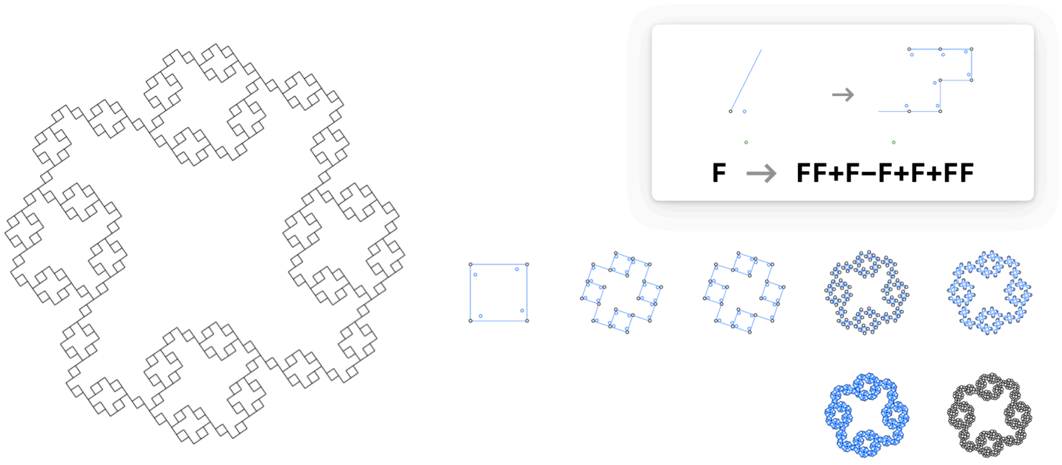


**Fig. 13** Rings

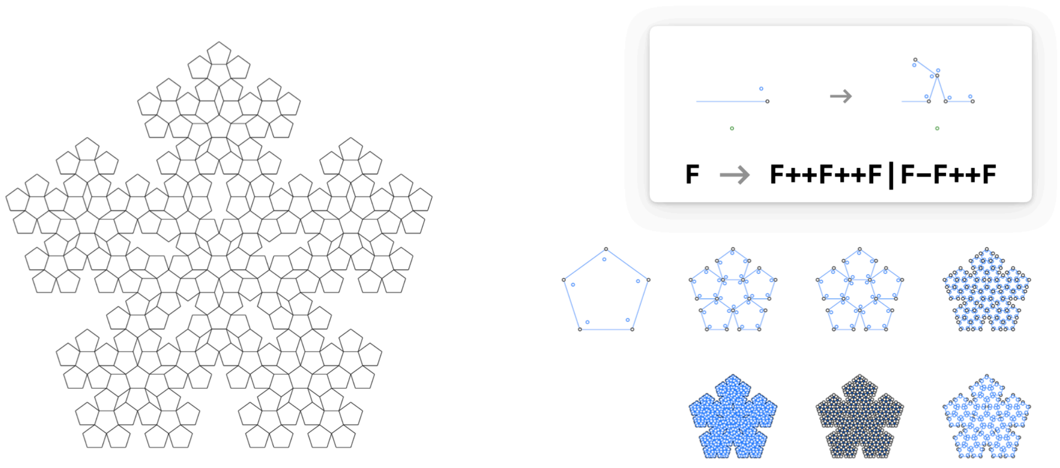


**Fig. 14** Pentaplexity

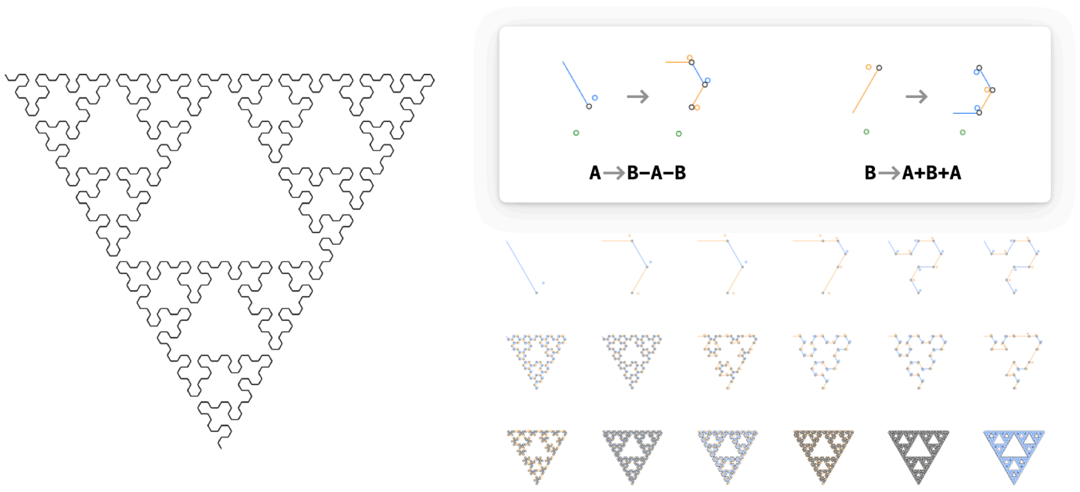


**Fig. 15** Sierpiński arrowhead

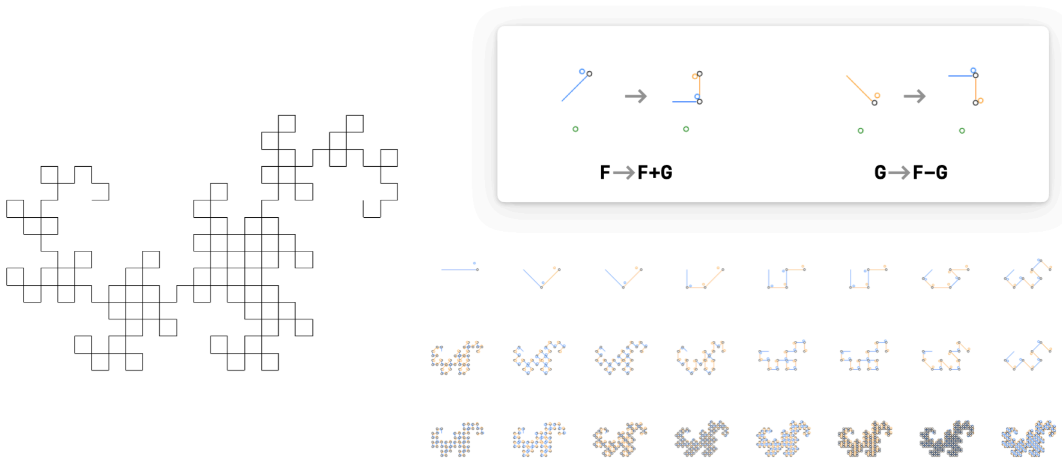


**Fig. 16** Dragon curve

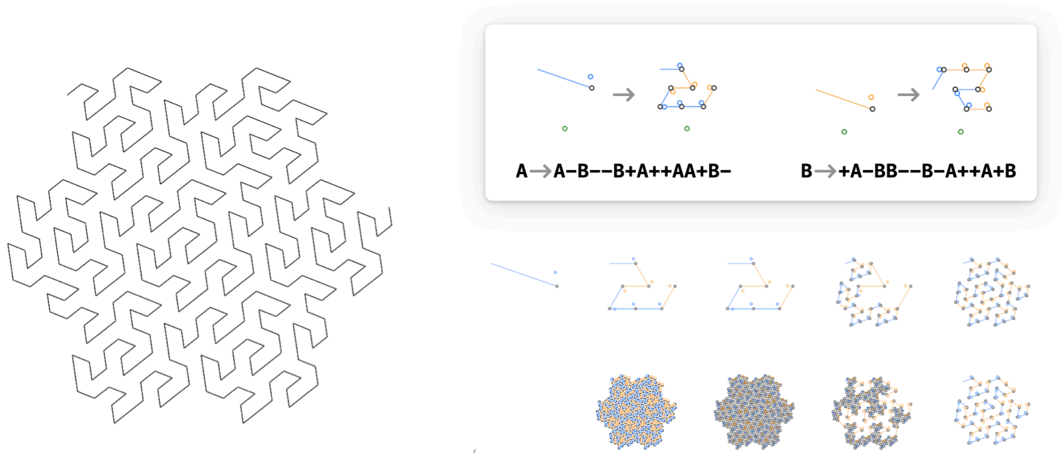


**Fig. 17** Gosper curve

A fast and convenient way to generate fractal geometry in Shape Machine also enabled some fun variations. There are 3 basic ways to create variation:

1. Add additional replacement rules after generating fractal shapes. Figure 18 shows a Gosper curve variation generated by replacing all lines into arcs.

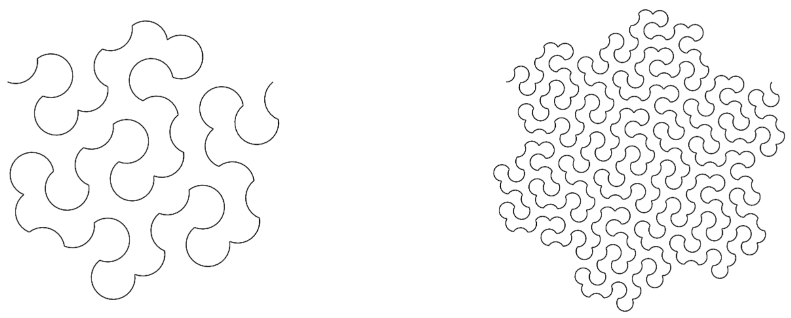


**Fig. 18** Gosper curve variations

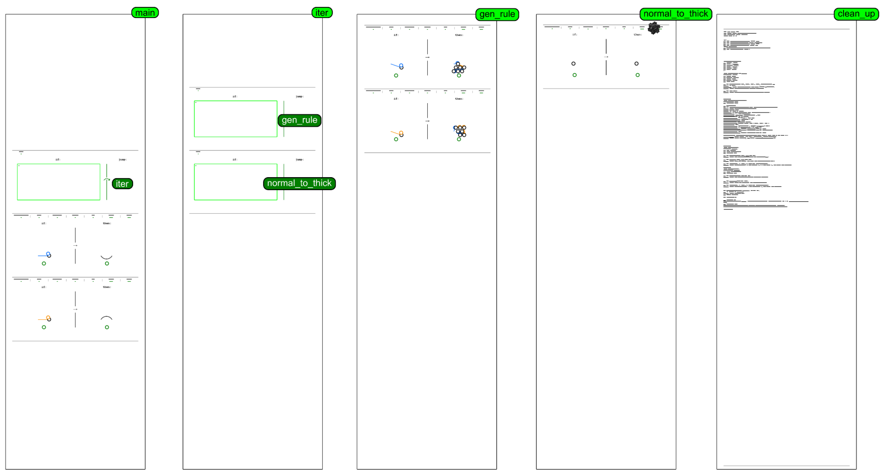


**Fig. 19** The modified program for generating Gosper curve variations

2. Modify left-hand side in generated rules. For example, the rule $F \longrightarrow FF + F - F + F + FF$ will generate an entirely different tessellation when the left-hand side $F$ is mirrored.

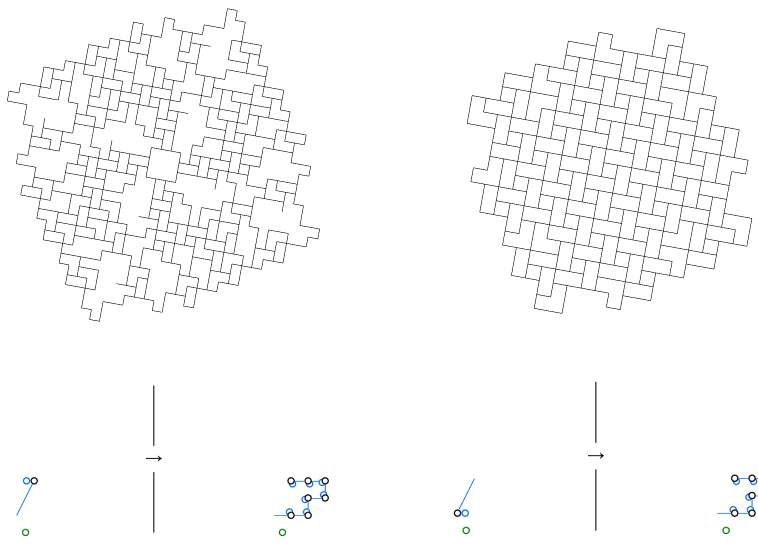


**Fig. 20** Tile variations

3. Modify right-hand side in generated rules. The rule of Pentaplexity, $F + +F + +F|F - F + +F$, will result in different fractal stars shown in Figure 21, if added with some arcs in right-hand side. This is different from simply adding additional replacement rules after generating fractal shapes, because the modification here will remain in the geometry between iterations.

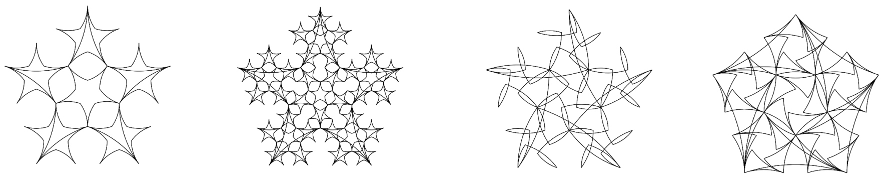


**Fig. 21** Pentaplexity variations

## 5 Future Work

We have demonstrated the integration of L-Systems into Shape Machine, enabling the generation of complex fractal geometries through programmatically generated DrawScripts. Future work could focus on transforming the Python logic into a more user-friendly interface within Shape Machine, allowing designers to interact with L-Systems directly. Additionally, expanding the generation rules and exploring more sophisticated fractal patterns could enhance the creative potential of this integration. Further research could also investigate the application of L-Systems in architectural design, industrial design, and graphic design contexts, exploring how these generative systems can inform and enrich the design process.

## 6 Conclusion

The integration of Lindenmayer Systems (L-Systems) into Shape Machine demonstrates a novel and powerful approach to computational design, bridging the generative capabilities of recursive grammars with the precision and versatility of CAD environments. This project has successfully shown how programmatically generated DrawScripts based on L-System rules can produce intricate fractal geometries and enable rich visual exploration in architectural, industrial, and graphic design applications.[8]

By implementing features such as parallel rewriting simulation, termination point adjustments, and automated rule generation, the project has addressed key challenges in adapting L-Systems to the Shape Machine framework. The results showcase the potential of this integration to not only enhance the efficiency of complex pattern generation but also introduce opportunities for creative experimentation through modifications in grammar rules.

The ability to create and customize fractal patterns, as demonstrated through examples like the Koch snowflake, Dragon curve, and Gosper curve variations, opens up avenues for innovative applications in design disciplines. This work highlights the value of combining mathematical formalisms like L-Systems with user-friendly computational tools, enabling designers to prototype and iterate complex forms with ease. Future developments could focus on expanding the library of generative rules, enhancing visualization capabilities, and exploring real world implementations in various design contexts.

Overall, this project lays the groundwork for more sophisticated interactions between shape grammars and generative systems, fostering new possibilities in computational design and visual aesthetics.

## References

[1] A. Lindenmayer, “Mathematical models for cellular interactions in development I. Filaments with one-sided inputs,” *Journal of Theoretical Biology*, vol. 18, no. 3, pp. 280–299, 1968, doi: https://doi.org/10.1016/0022-5193(68)90079-9.

[2] P. Prusinkiewicz and A. Lindenmayer, “The Algorithmic Beauty of Plants,” in *The Virtual Laboratory*, 1990. [Online]. Available: https://api.semanticscholar.org/CorpusID:168626

[3] P. Bourke, “L-Systems: A Mathematically Elegant Framework for Plant Growth.” [Online]. Available: https://paulbourke.net/fractals/lsys/

[4] S. Papert, *Mindstorms: children, computers, and powerful ideas*. USA: Basic Books, Inc., 1980.

[5] G. Stiny and J. Gips, “Shape Grammars and the Generative Specification of Painting and Sculpture,” in *IFIP Congress*, 1971. [Online]. Available: https://api.semanticscholar.org/CorpusID:36431081

[6] G. Rozenberg and A. Salomaa, *The mathematical theory of L systems*. Academic press, 1980.

[7] A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools (2nd Edition)*. USA: Addison-Wesley Longman Publishing Co., Inc., 2006.

[8] M. Leyton, “A process-grammar for shape,” *Artificial Intelligence*, vol. 34, no. 2, pp. 213–247, 1988, doi: https://doi.org/10.1016/0004-3702(88)90039-2.