Programming Assignment Report
Automata Theory Monsoon 2024, IIIT Hyderabad
Student Name: Ayush Sheta
Roll Number: 2023101057
Submission Date: 8 Sept 24

Table of Contents:

---

1. Introduction

This report covers the solutions to the programming assignment for Automata Theory Monsoon 2024. The assignment consists of three main questions:

1. DFA-based pseudo-random number generator (PRNG).
2. Simulation of L-systems to produce graphical outputs.
3. Deriving production rules based on given images of L-systems.

Question 1: DFA Based PRNG

1.1 Approach

For this question, I designed a Python program to simulate a DFA-based pseudo-random number generator (PRNG). The DFA transitions are processed iteratively to output the states, and the frequencies of each state are recorded.

1.2 Assumptions

- No dead states.
- All transitions are deterministic, meaning each state transitions to exactly one other state.
- For transitions that don't appear in the input, they are considered irrelevant and unreachable.

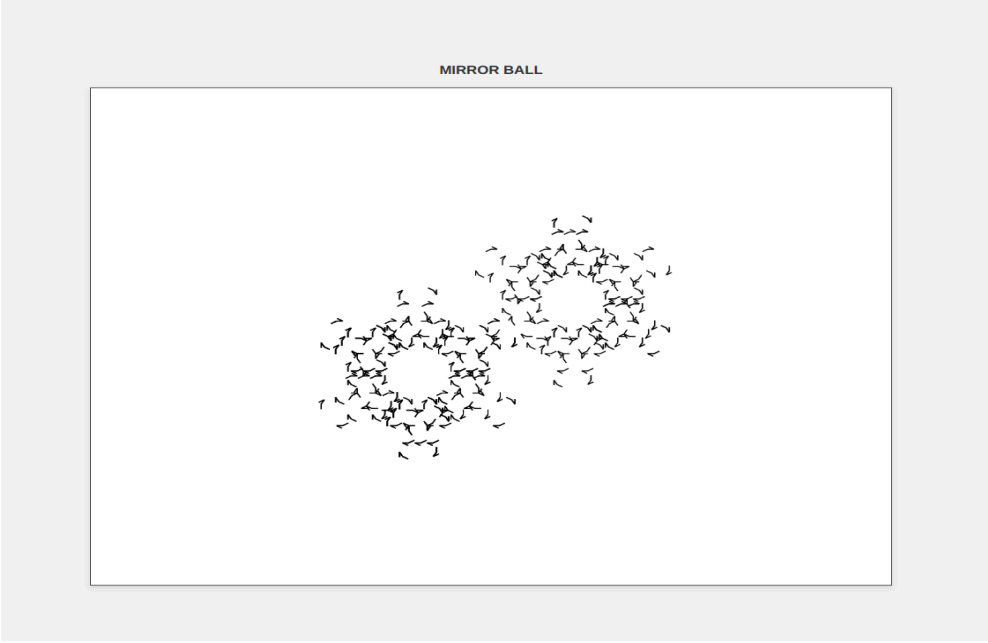
1.3 Results

The results for each test case include the frequency count of each state in the range [1, N].

Question 2: Simulating L-Systems

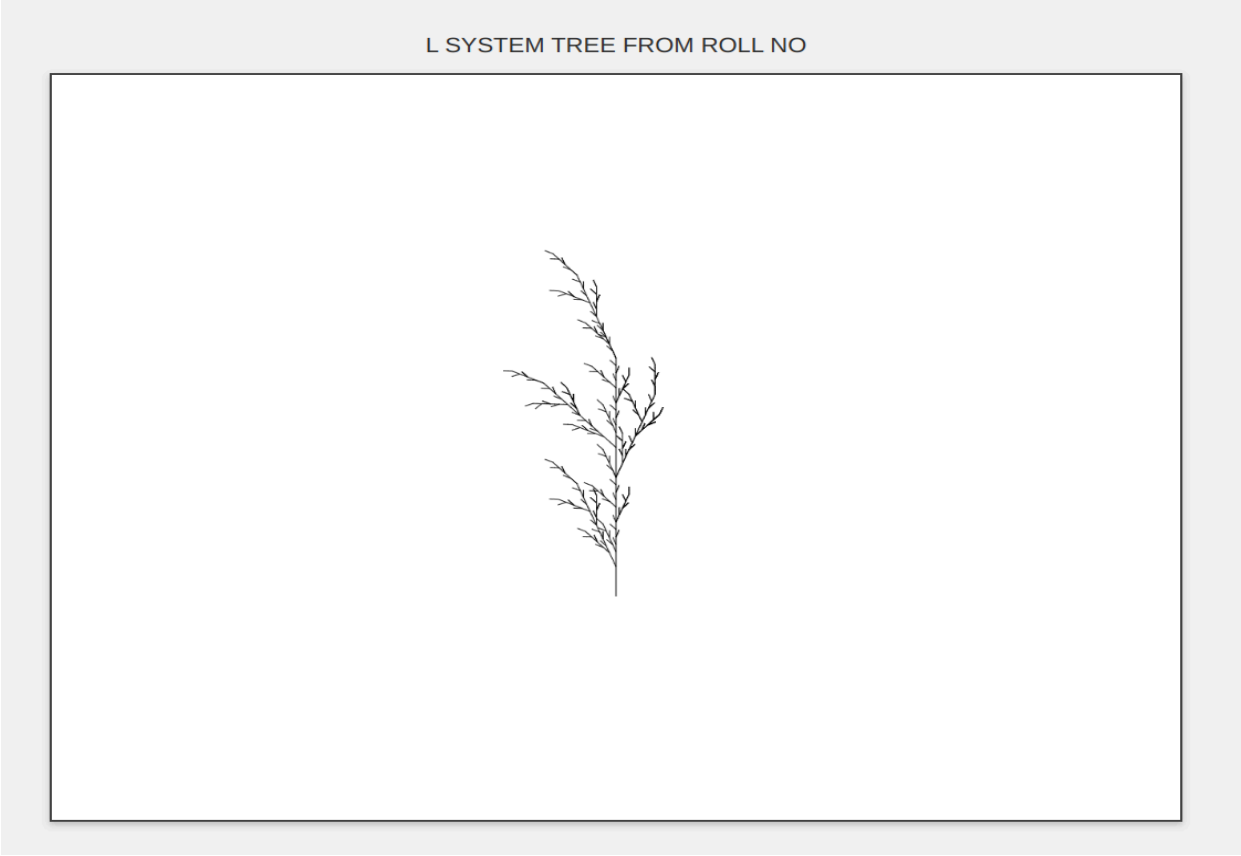
2.1 Mirrorball

For the "Mirrorball" L-system, the provided rules and axiom were iterated **9 times**, and the corresponding fractal-like image was generated.



2.2 Tree

For the "Tree" L-system, the rules were iterated **4 times** based on the angle calculations. The pattern produced closely resembles the branching structure of a tree.

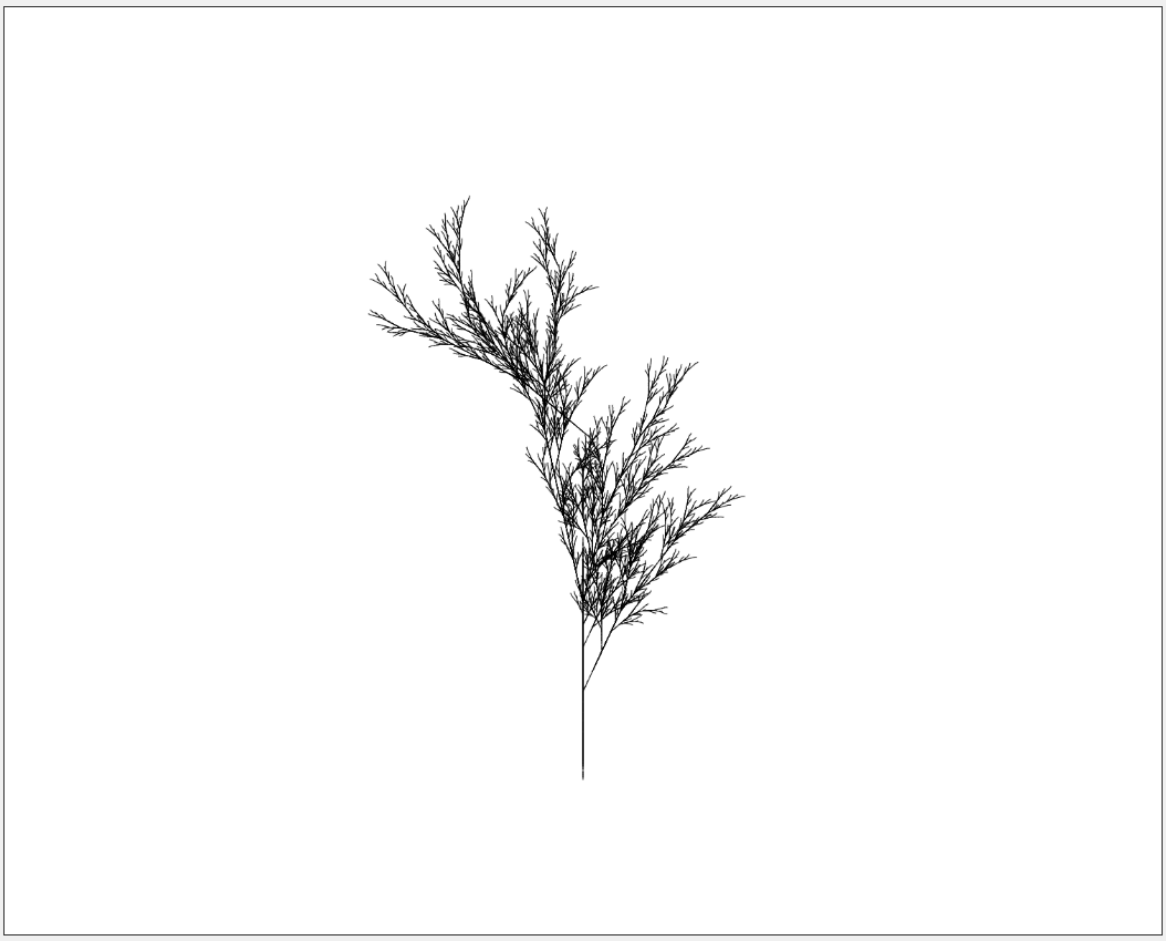


2.3 Anything that can happen will happen

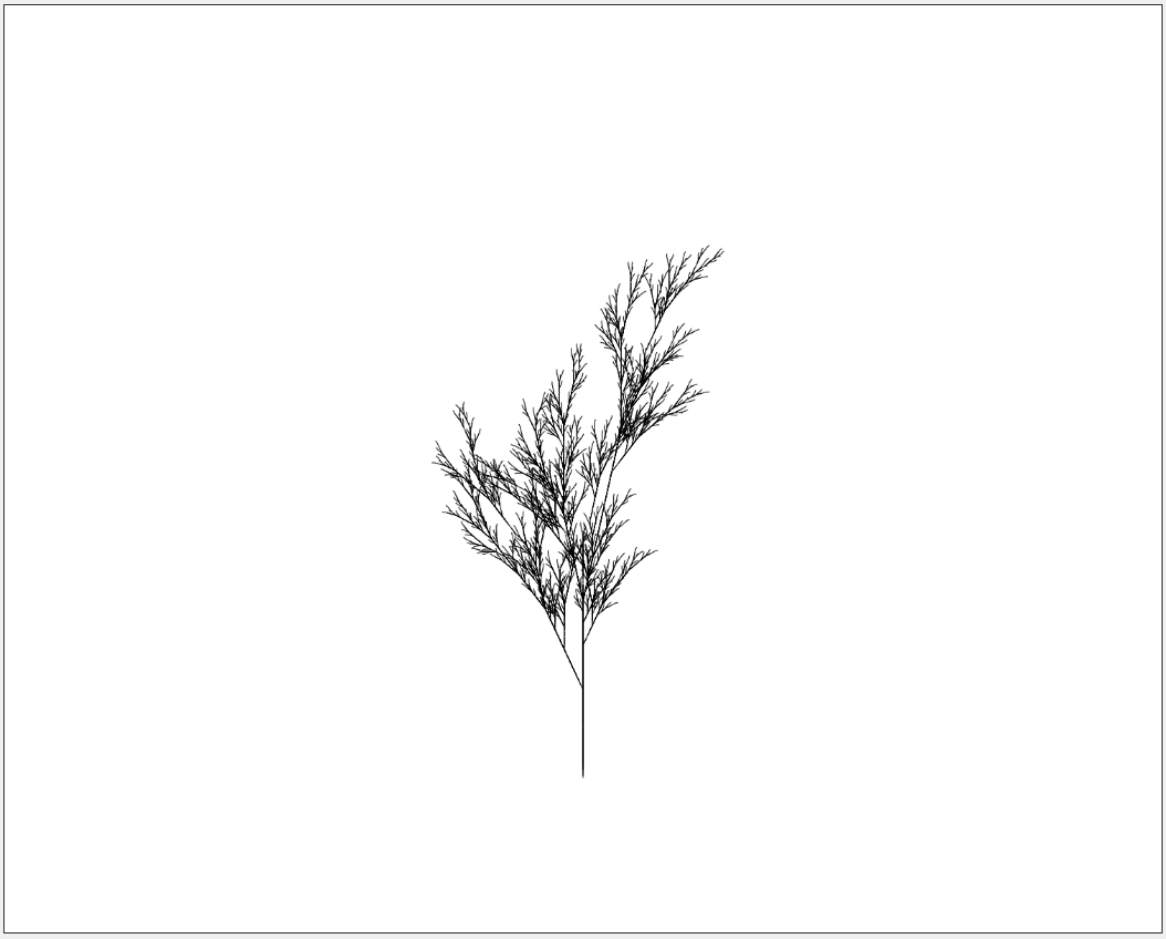
The random nature of the rules generated 5 different configurations for this part, and their images were captured after **4 iterations.**

a.com

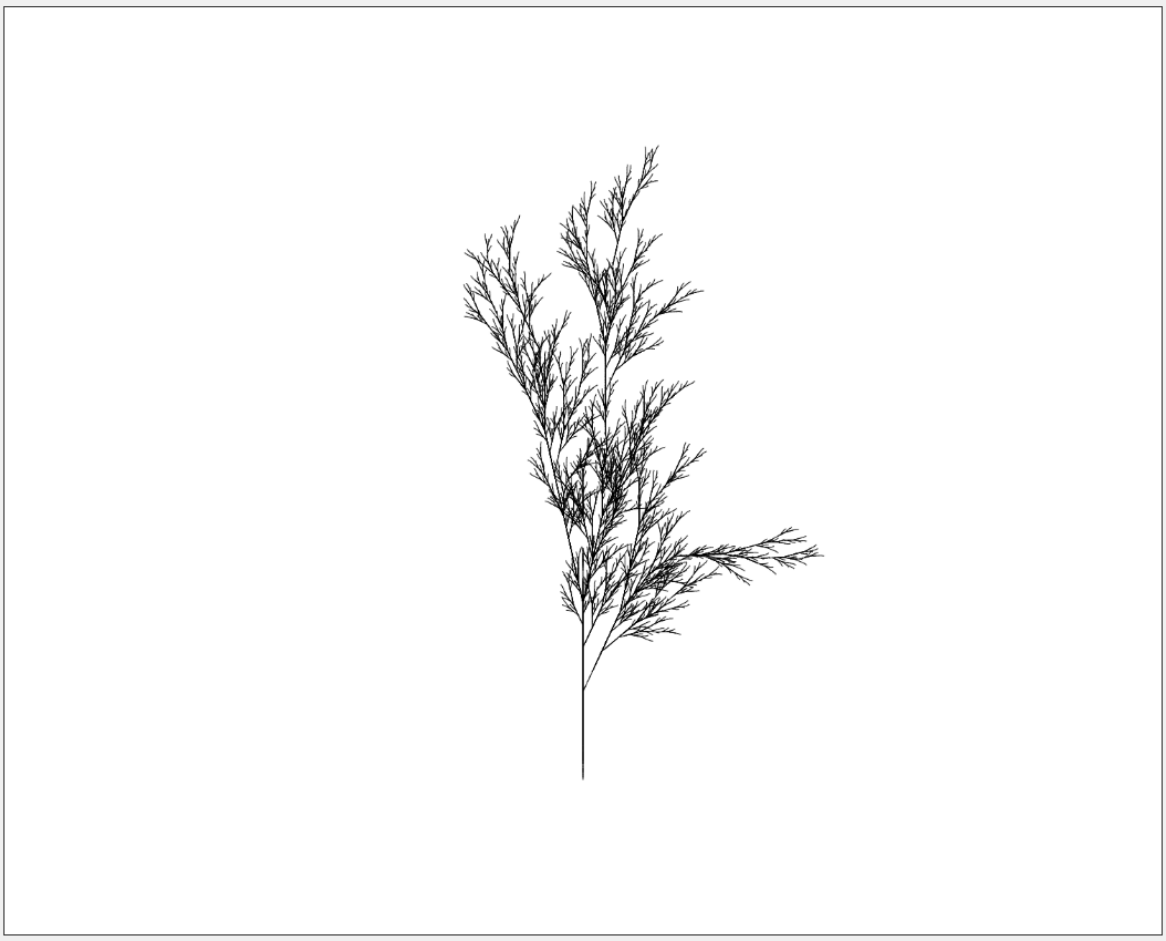
L-System RANDOM TREE Generator



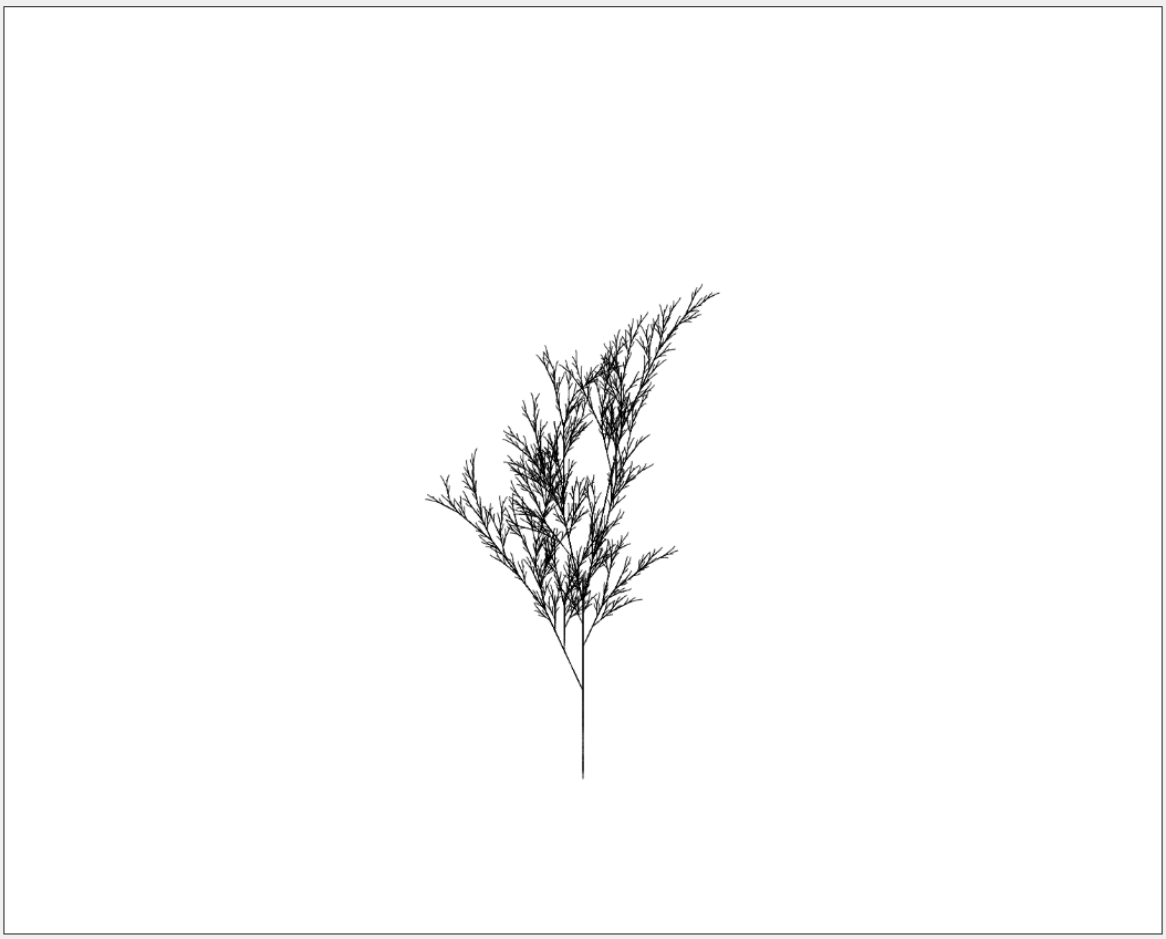
L-System RANDOM TREE Generator

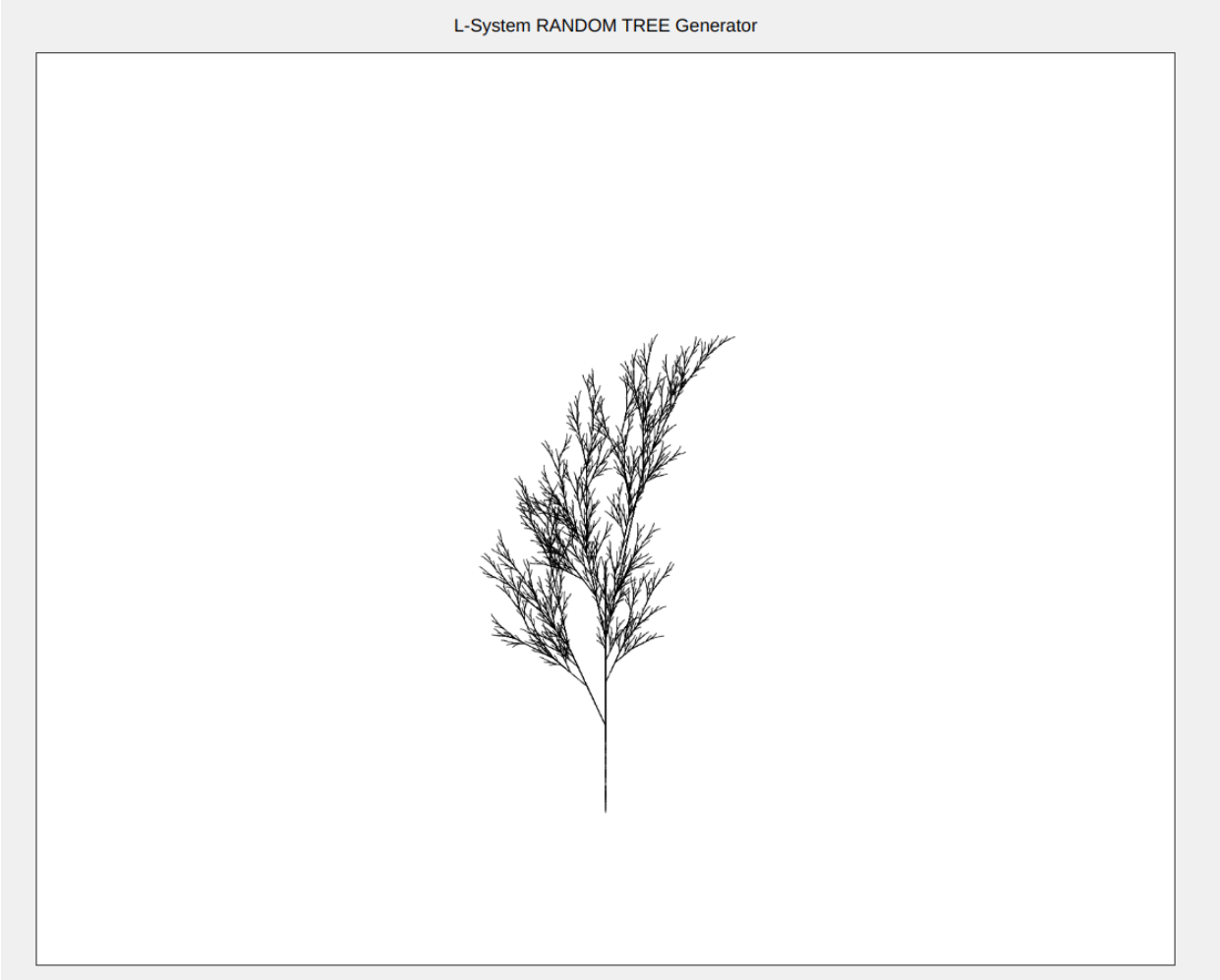


L-System RANDOM TREE Generator



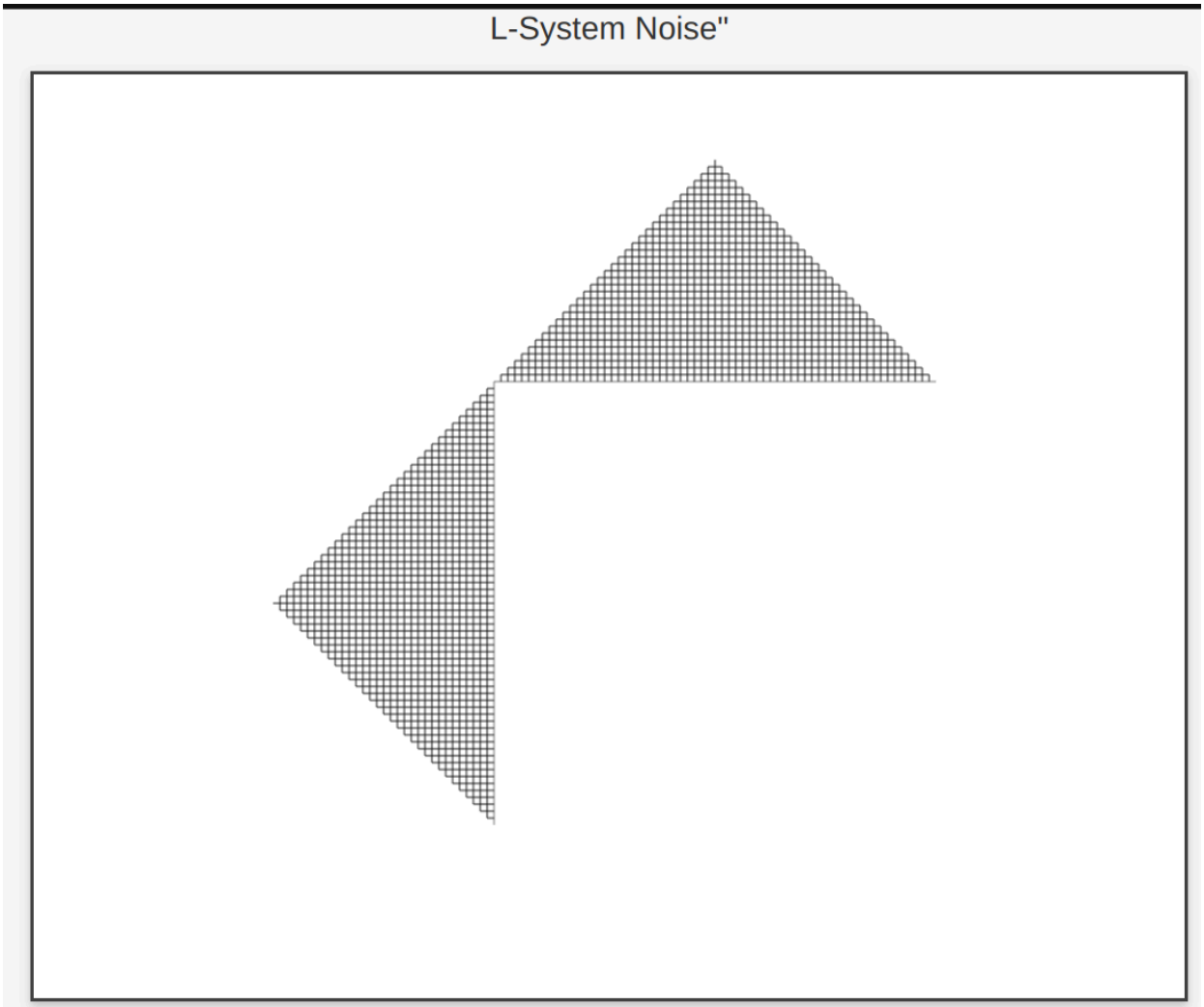
L-System RANDOM TREE Generator





2.4 Content without context is noise

Using the axiom and production rules, I iterated the system 9 times to generate the final pattern.



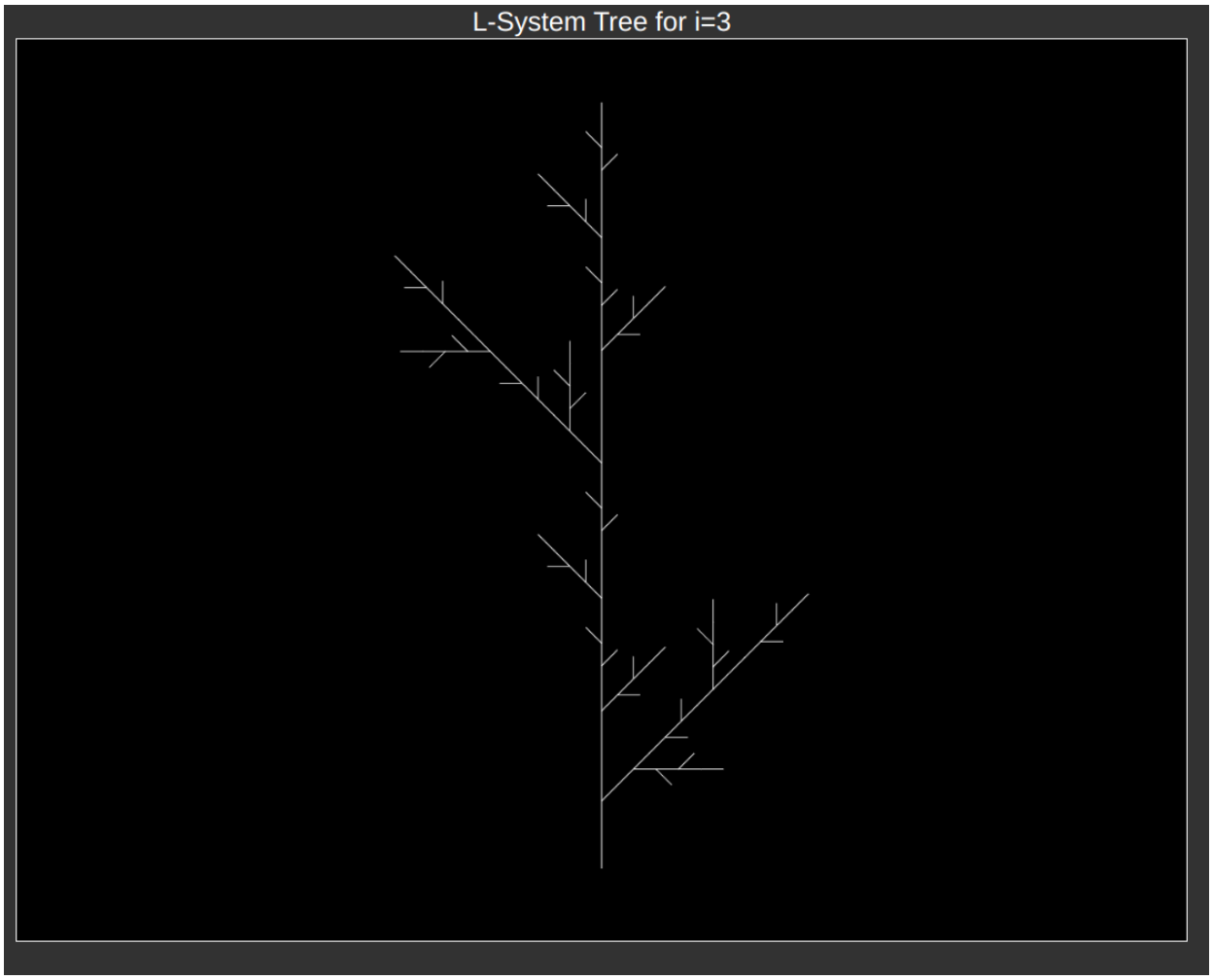
2.5 Assumptions

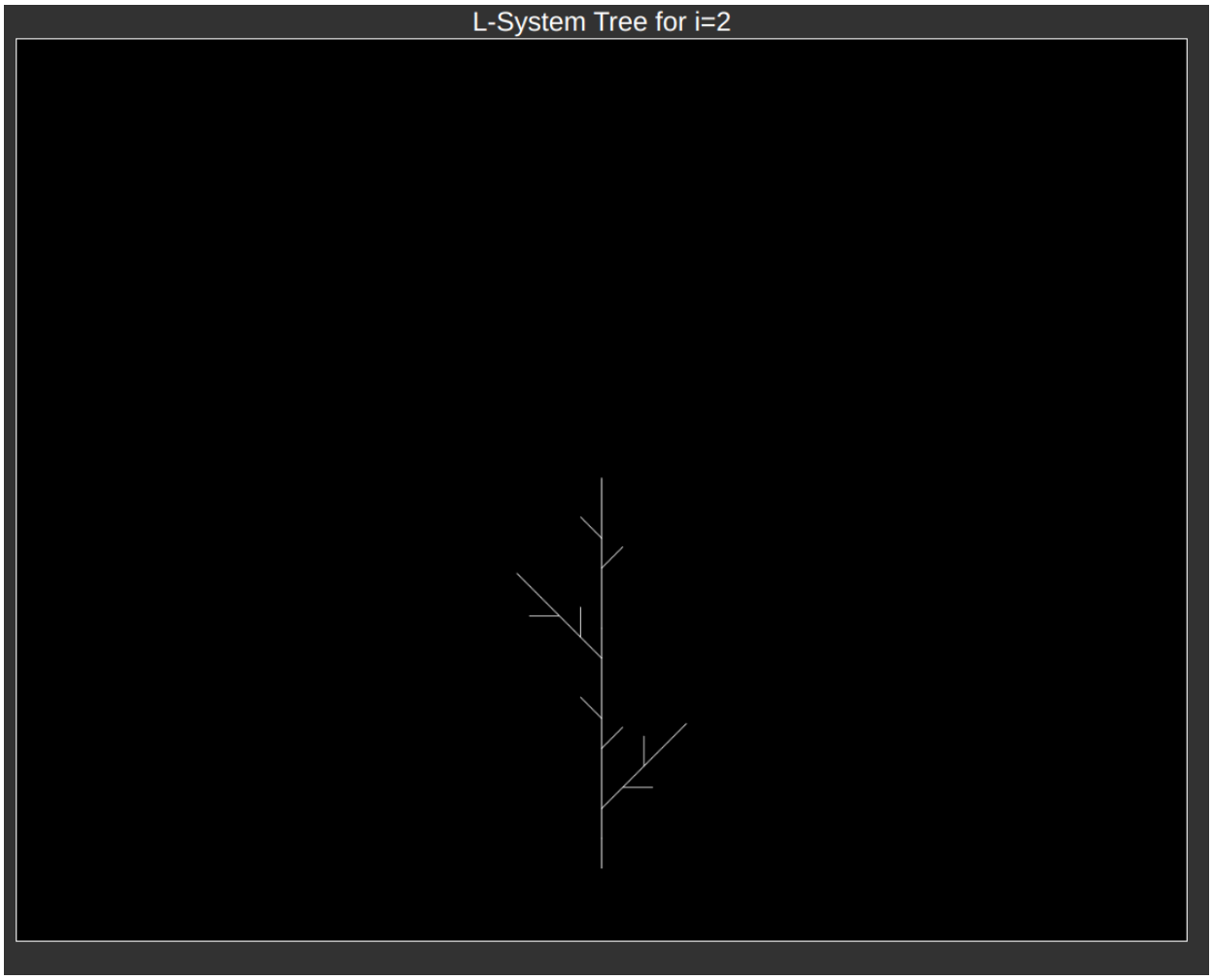
- I assumed that the provided rules for each sub-part were complete and accurate, meaning no further adjustments to the rules were needed.
- The angle calculations for the "Tree" system were computed using the provided formula and were tested for correct application during iterations.

Question 3: L-Systems Axiom

3.1 Stick Plant

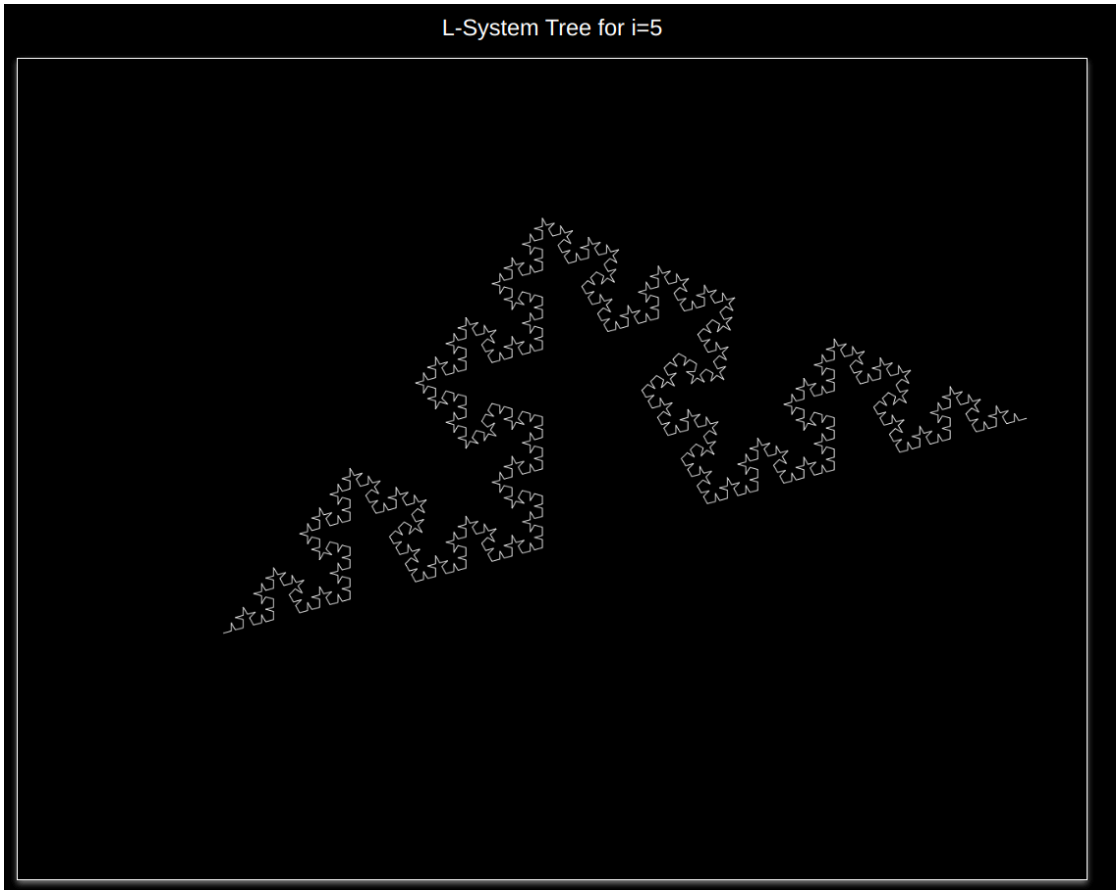
After examining the graphical outputs, I derived the following axiom and production rules for the Stick Plant system.

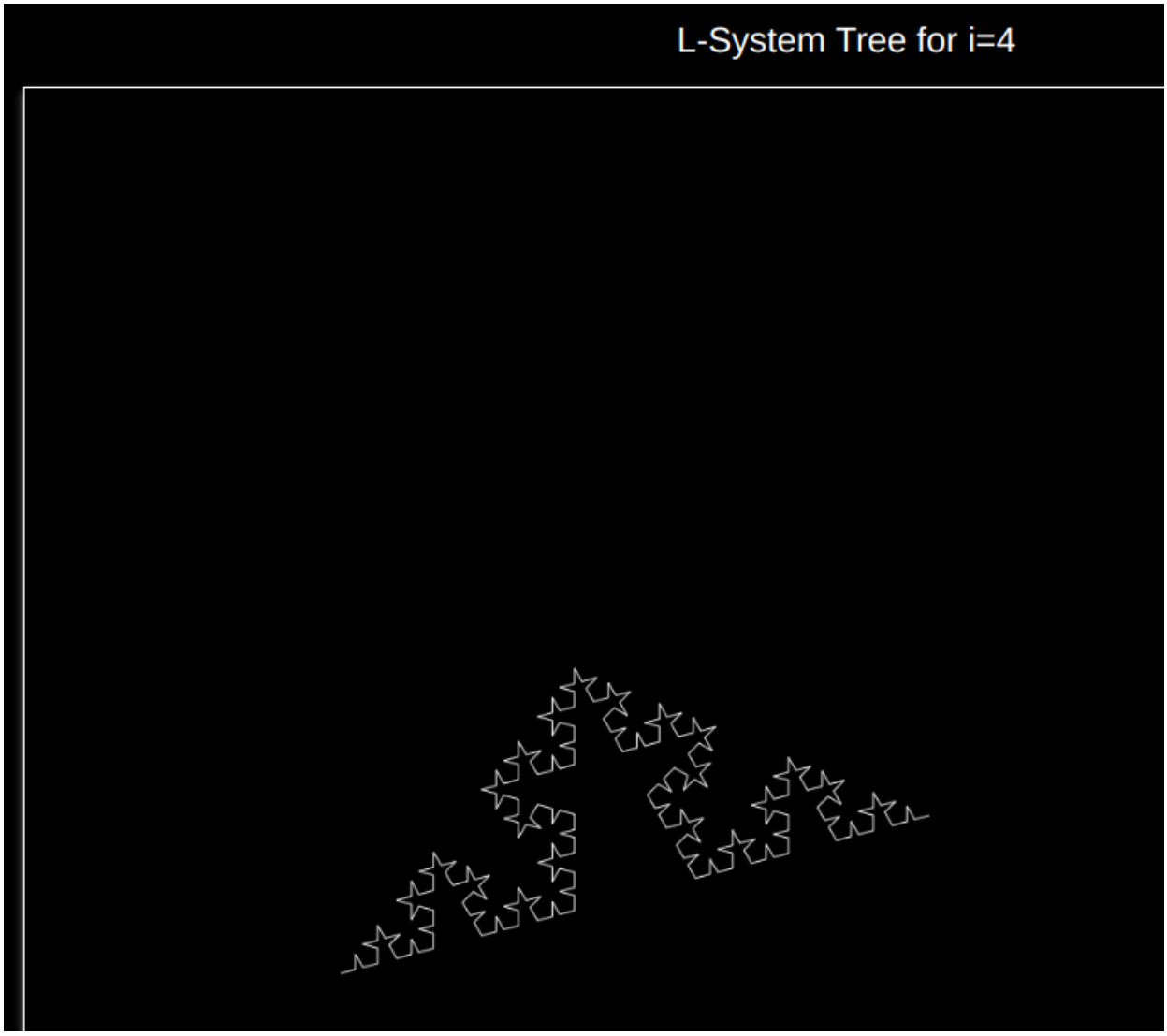

L-System Tree for i=3



3.2 Santa K(l)osh

For the Santa K(l)osh pattern, I identified the axiom and the production rules by analyzing the iterations shown.


L-System Tree for i=5



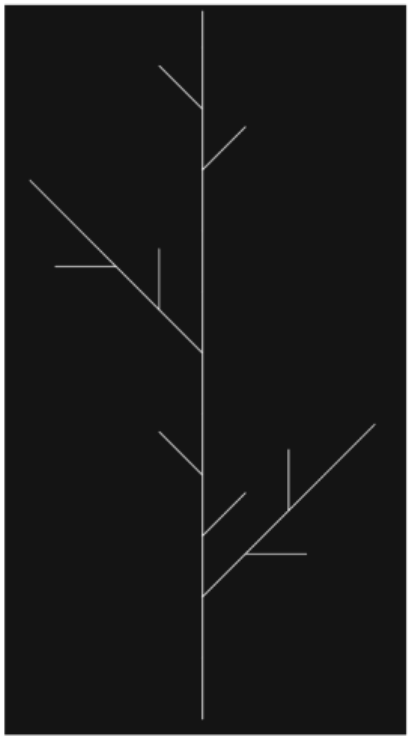
3.3 Assumptions

- **In stick_plant.json I have made a new rule ‘G’ which is the same as ‘F’ (as in the question).**
- **In santa.json I have modified the ‘F’ rule. After modification, if ‘F’ appears then first it will rotate right by 75 deg, then move forward and draw a line on that path and then rotate left by 75 deg.**

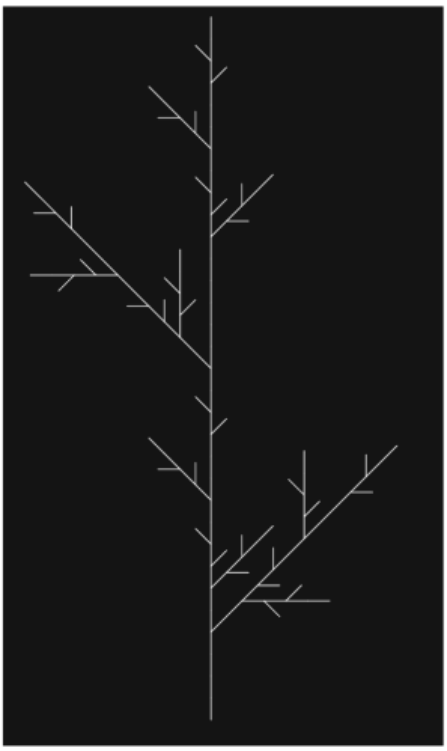
3.4 Side by Side Comparison

Given images :

For stick

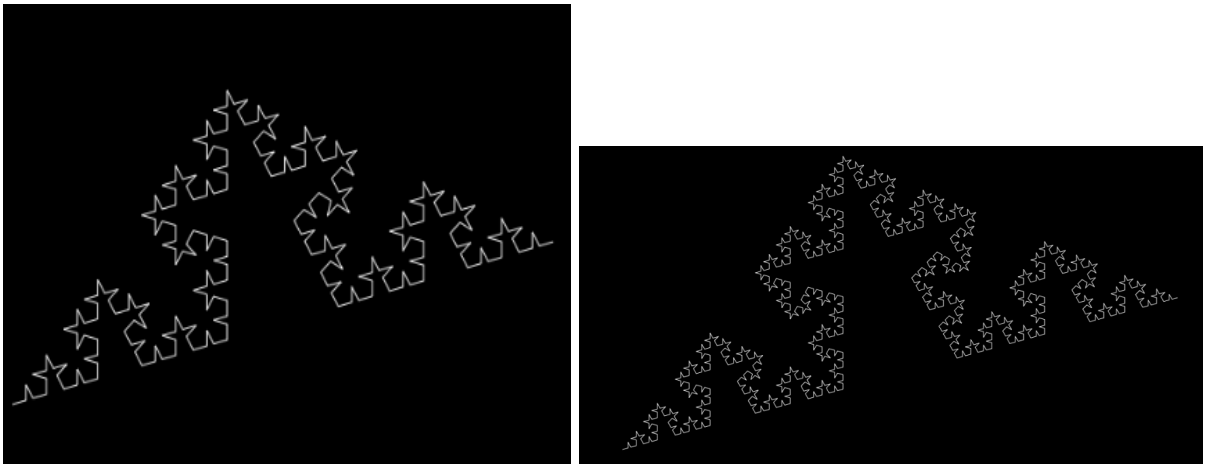


(a) after 2 iterations



(b) after 3 iterations

For Santa:



(a) after 4 iterations

(b) after 5 iterations

Figure 2: Santa K(l)osh

How to Run the Code

Question 1: PRNG

To run the PRNG code:

1. Navigate to the `q1` directory.

Execute the following command:
`python3 prng.py`

2. Provide input according to the sample input format.

Question 2: L-Systems

To generate the images:

1. Use the appropriate software (i use vs code -> live server , open with live server or as a normal html file)
2. Navigate to the corresponding directory (e.g., `q2/mirrorball.html`) and run the program to generate the image.

Question 3: L-Systems Axiom

To generate the JSON output for the axiom:

1. Navigate to the `q3` directory.
2. The JSON files for Stick Plant and Santa K(l)osh are located in `stick_plant.json` and `santa.json`.

Conclusion

This report summarizes my approach and solutions for each question in the Automata Theory assignment. The output for each section was tested and verified based on the problem requirements.
