The L-Systems Explorer: An Introduction to Generative Art

A Thesis
Presented to
The Division of Mathematics and Natural Sciences
Reed College

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Arts

R Jacob Hoopes

May 2023

Approved for the Division
(Computer Science)

_____

Dylan McNamee

Acknowledgements

This thesis has been a joy to put together. I've reveled in each opportunity to engage with the material and explore more about this idea whose depth I feel I have only begun to understand. Engaging with my peers and mentors about L-Systems and Generative Art has opened my mind to new ways of thinking about these ideas, not to mention the joy I feel when they respond with their interests in kind. I've attempted to make this thesis multidisciplinary, in the spirit of L-Systems as a child of both Art and Computer Science, but also in the spirit of Reed College as a Liberal Arts institution. For this, and in all things, I have found that no idea is complete without a range of perspectives.

I'm grateful to Sky Peterson, Jon DeVries, Jaden Nichols, Aria Killebrew Bruehl, Izzy McKenna, Keith Ng, Will Howes, Astrid Lilly, Nathan Hagan, Izzy Laun, Ariana Fleet, Calvin Beeman-Weber, Ross Tidwell, Miles Silvey, and all the others who I've talked with about the construction of this thesis. You've kept me afloat.

Unending thanks to my roommates, Henry Wilson, Brent Ellis, and Gabe Fish for your support in everything. This would not be possible at all without your presence. With Bridge, Root, and whatever other new thing we have going on, this has only happened because I've been able to retreat back to a loving home when I'm done with my work. My gratitude really surpasses expression.

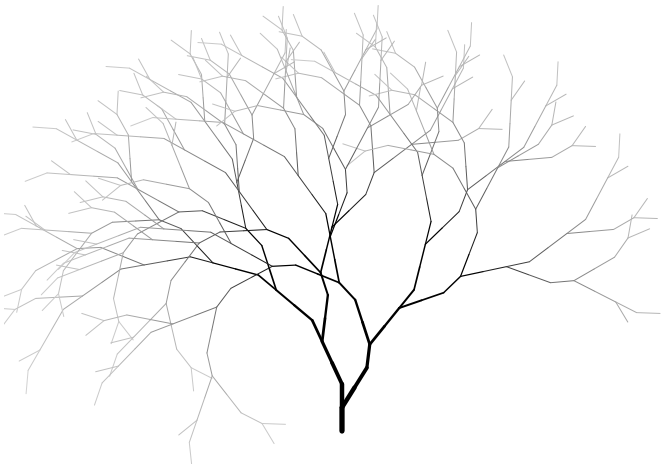


Figure 1: Gratitude[1]

[1] A; A → F[F+FA]-FB, B → F+F[–FA]+FB; $\theta = 32°$, ϕ = -7°; Iterations = 7; IW = 4.5, WC = 0.85

Table of Contents

List of Figures

Abstract

The world is a mess of systems. I hope to establish the context for one system that interests me and build it up for you so that you may feel the same wonder that I have felt. There will be an overview of the structure and function of L-Systems, later followed by the brief introduction of some different types of L-Systems. I'll explain the program that allows users to learn about and explore L-Systems, followed by a demonstration of its capabilities and its development. I will also discuss some related work, and go on to explain possible extensions. Wrapping things up, I take a step back from the program to ask some larger questions.

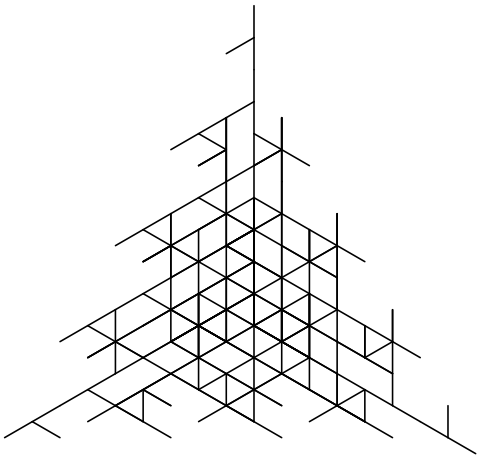


Figure 2: Three-point Mirror[2]

[2]A; A $\rightarrow$ [F[–FA]FA]+A; θ = -120°; Iterations = 5

Dedication

To the unending search for understanding.
for worlds. in worlds and over worlds

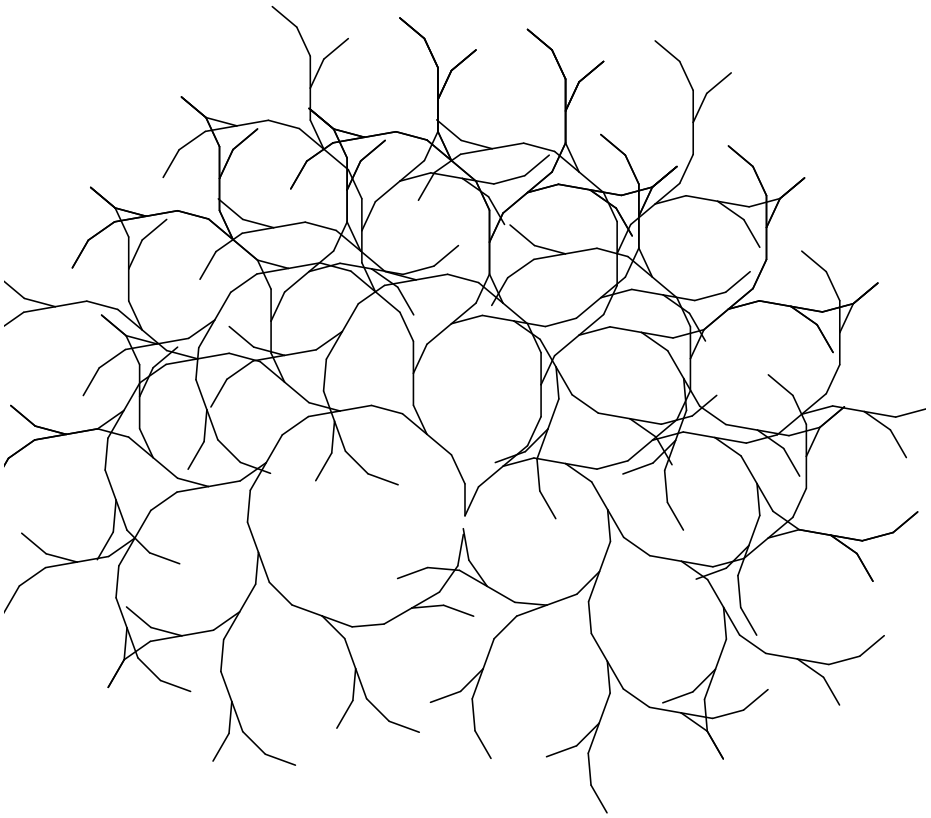


Figure 3: Transcendent Spirals[3]

[3] A; A → [F+F+FA]–F–FA; $\theta = 25°$; Iterations = 7

Introduction:

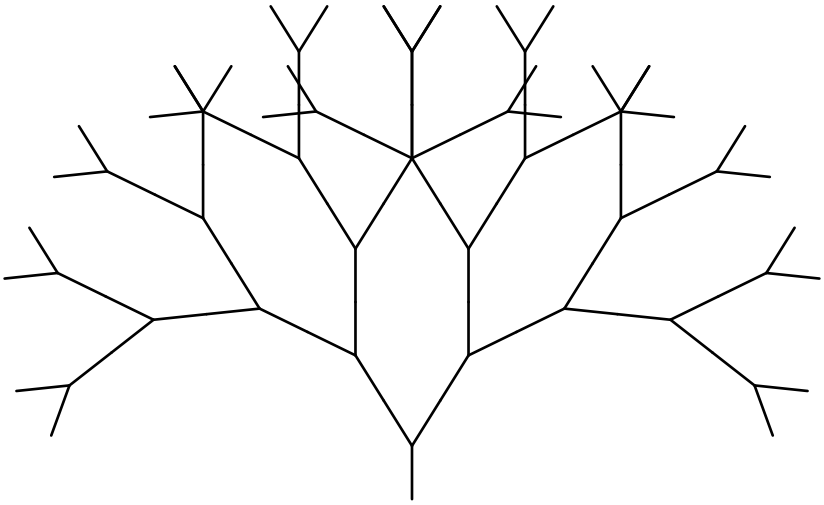


Figure 4: Classic Tree[4]

The world of Computer Science can seem like a wilderness, teeming with complex and specialized topics like Zero-knowledge proofs,[5] NP-Completeness,[6] Finite State Machines,[7] and the Halting Problem,[8] but I hope to help show how approachable this discipline can be. Specifically at the hands of those from distant disciplines, like Anthropology, Economics, and Psychology, as well as the locals, pure Math, Bio, Chem, and Physics. The tool I introduce here will demonstrate the accessibility of this powerful, subtle, yet familiar discipline: Computer Science. The ideas that I employ in the program were named L-Systems, and their capabilities and beauty can astound.

Before we jump in, there are some things that must be said. This is a Computer Science thesis, and so (essentially by necessity,) there will be code. I would like to keep this thesis as accessible as possible to those who don't code, and so I

[4] A; A $\rightarrow$ F[+FA]–FA; $\theta = 32°$; Iterations = 5

[5] Goldwasser et al. (1989)

[6] Cook (1971)

[7] Ben-Ari & Mondada (2018)

[8] Turing (1936)

have tried to design it so that anyone with the ability to feel wonder and joy can find something of value between these covers. The code is accessible on my github, `github.com/RJacobH/Thesis` as well as my website, `rjacobh.github.io/website`. Some parts of the code of the central project will also be discussed at a later point in the thesis. However! I have included pictures at every moment I could, and I'd like to imagine that they tell enough of a story by themselves that to some extent the words are ultimately unnecessary.

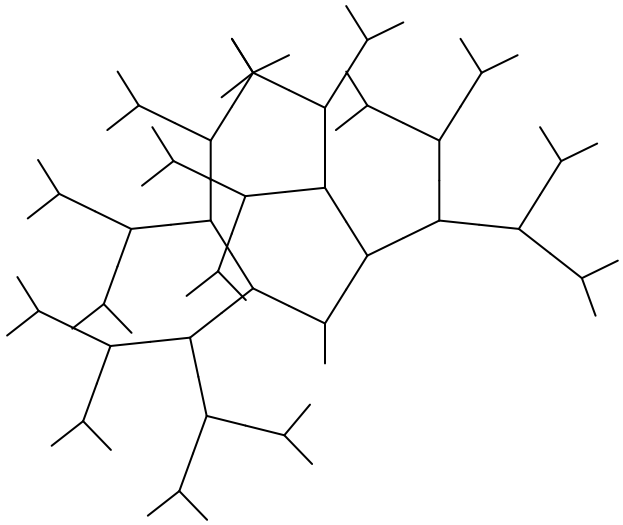


Figure 5: Tilted Classic Tree[9]

The structures that I will be discussing throughout this thesis are *Lindenmayer Systems*, usually called *L-Systems*. They are a type of "rewriting" system that have a distinct visual flavor. Introduced by Aristid Lindenmayer in 1968[10] , they were first imagined as a framework for understanding the development of simple multicellular organisms. They were soon applied to larger systems, such as the structures of plants. A publication of Lindenmayer and others, *The Algorithmic Beauty of Plants*,[11] was written in 1990 and has been an invaluable resource for me in the formation of this thesis. It introduces L-Systems as a tool to celebrate and understand the beautiful complexity of plants, as well as to understand complex developments with limited arguments. Beyond this book, there have been a number of other vital resources in this process which I will discuss as they become relevant.

The images that appear throughout this thesis that look like Figure 5 are specific demonstrations of L-Systems. If you look at the footnotes that I've included with all of these images, you'll see a line of strange characters. I'll briefly describe here what those are, though I'll go into more detail in Section 1.1. Right now, all that's important to know is that they're fundamentally recipes for creating each of the

[9] A; A $\rightarrow$ F[++FA]–FA; $\theta = 32°$; Iterations = 5

[10] Lindenmayer (1967)

[11] Prusinkiewicz & Lindenmayer (1990)

visualized L-Systems.[12] If you'd like, you can create any of these images on your own or in my program by following the recipe. Also, unless otherwise noted, every image in this thesis was created or generated by me. Except where noted, all the L-System images that I show have been created by me in one of my programs.

L-Systems are fundamentally a multi-subject topic, they cannot be understood from the perspective of just one discipline. An idea of their structure and abilities can only be sufficient if a variety of fields are consulted. I've tried to reflect this awareness in the form of my thesis as well as its content. I've tried to write something that would be as interesting for an artist to read as a computer scientist. If you're here for the pictures, please enjoy! But if you're here for the discussion of the systems that create those pictures, I urge you to stay and engage with some of the larger questions that I've asked about life, computers, and the place this thesis has in that discussion. But regardless of what brought you here, I hope you have a fun time looking through this thesis, and it warms my heart to know that there are people interested in this work which has captured by imagination so completely.

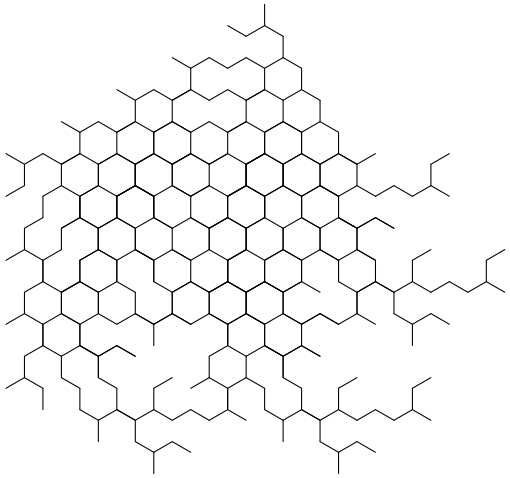


Figure 6: Aspirational Honeycomb[13]

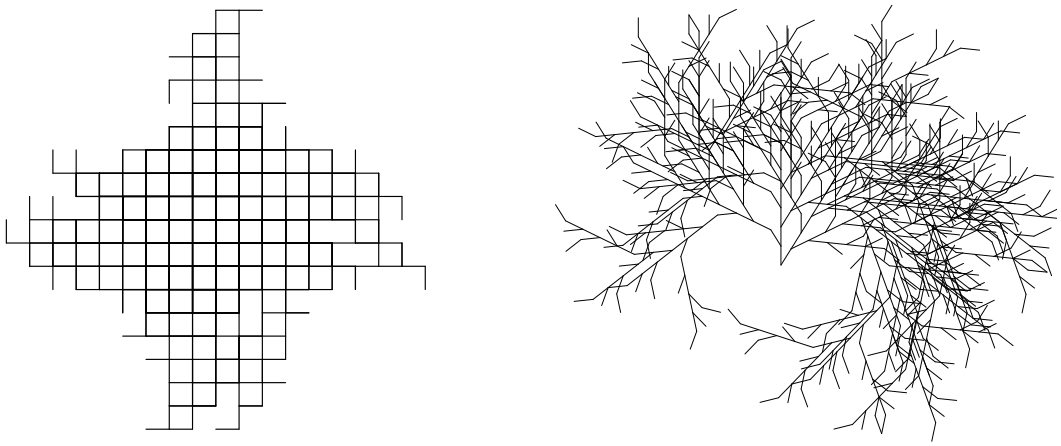
Motivations

The primary motivation, and the reason that I have become so personally invested in this topic, is the idea of *pattern*. Patterns that I demonstrate with L-Systems and other generative arts appear at different scales throughout life, the world, and our universe. I was in near constant conversation with my friends and acquaintances in other disciplines while I was writing this, and I was amazed at how often questions

[12]Beyond the discussion in Section 1.1, I direct you to Section 4.3.1 to learn more about about the fifth part specifically.

[13]A; A $\rightarrow$ +F[A+F–FA]–FA; $\theta = 60°$; Iterations = 5

of self-reference, universality, and beauty from simple rules arise in areas beyond just Computer Science.



(a) 90 degrees[14] (b) 32 degrees[15]

Figure 7: Nearly Identical L-Systems

The second motivation is the simple beauty of many of the patterns that emerge when working with L-Systems. This is the motivation that I imagine might be the most interesting to artists, be they visual artists or otherwise. It is also the reason I became aware of L-Systems and their relatives in the first place, so it is as essential a part of this story as the first motivation. It should be noted that it feels like the division between these fields is not nearly as dramatic as one might imagine. There are varied connections between the Art aspects of this work and the Computer Science and Math aspects, some of which I will explore later in the thesis.

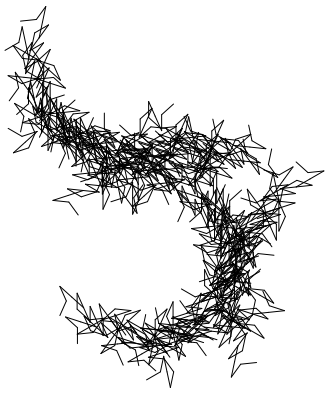


Figure 8: Fishhook[16]

[14]A; A → [F[+F+FA]FA[F–FA]+FA]-FA; $\theta = 90°$; Iterations = 4
[15]A; A → [F[+F+FA]FA[F–FA]+FA]-FA; $\theta = 32°$; Iterations = 4
[16]A; A → +F[A]F–FA–FA; θ = -156°; Iterations = 6

The third motivation is tied in with a philosophical understanding of these structures. As I delved deeper into L-Systems, I came across the claim that these structures themselves were *alive*.[17] This felt absurd, but I understood how a person could see life within the symmetry and complexity. I found that it was extremely difficult for me to draw the line between living and non-living matter without the choice feeling somewhat arbitrary. A conventional definition, that the smallest unit of life is the cell due in part to its ability to self-replicate, felt insufficient. These structures I was in the process of investigating do replicate themselves to some extent, so why shouldn't they live? These questions on the nature of life stuck with me, and I found myself asking the same questions of life in the context of other machines. I ask address larger issues of the place of machines in the creation of art and the creative process. It also feels like these questions about the border between living and non-living things are essential when we're experiencing what may be the first great wave of AI systems.

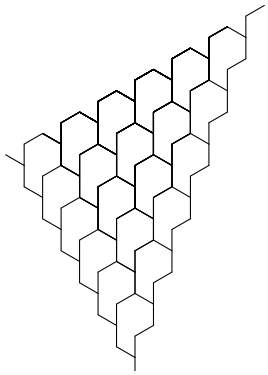


Figure 9: Not Quite 3d[18]

This thesis does not aim to answer any of these questions. I ask them and suggest that they are important questions to be asking, but mainly state them so that you see what I've been thinking about as part of this process. I present my research as an open book for you to read and find whatever is valuable for you. Although, I do assert the importance of L-Systems and other simple, powerful tools that a person can hope to understand, especially in a time when so many of the systems that control our lives are invisible to us. These questions have been a foundational piece of my work, and have shaped the development of this thesis as much as the content itself.

I hope that L-Systems will find a broader audience, either as a result of the work that I've done here or as part of some other exploration of their abilities. This came to be a motivating project for me for more reasons beyond those outlined above, and I imagine that someone who sets new eyes on this work could see it in a way I never could have imagined. I go into more detail near the end of the thesis about what I believe I have contributed to the conversation around L-Systems and their relatives, as well as what I believe the program I wrote could do if more time is given to its evolution.

[17] Kelty & Landecker (2004)

[18] A; A → F[+F–A]F–F+A, B → F–FA; $\theta = 60°$; Iterations = 7

The Path We Will Follow

The first thing I will do in this thesis is formally introduce L-Systems. I will give different perspectives on the construction of these systems to help the reader see that they are incapable of being understood from any single one of these perspectives. The only way to get the whole picture is to know that a single explanation will always be insufficient.

From there, I'll begin to demonstrate the abilities of L-Systems. I've chosen to display them as a 2d or 3d visual structure, though there are other ways to express them, which are discussed in Section 4.3.1. While they can look compelling, the process of creating them has been hard to access and engage with for those outside of Computer Science. I wanted to make it so that people with any knowledge about them could play with them. I wanted to bring wonder and teach about these systems, so I wrote a program to do just that. I describe the structure of the program, some steps I took in its development, and how to use it, and I direct interested readers to try the program themselves. How to access the program is detailed in Section 2.4.

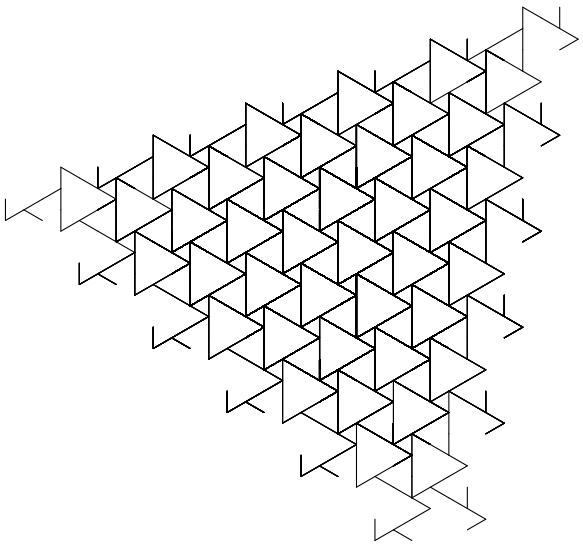


Figure 10: Triangle Heaven[19]

In Section 3.2 I go on to talk about *generative art* and the broad crossover between Art and Computer Science that has fueled my explorations and the work of many other contemporary artists. I talk about some of my earlier explorations into L-Systems and generative art, and connect those experiments with the resulting thesis. After that, I talk about some of the questions I've asked during this research, in Computer Science and in areas beyond. I place this thesis within its historical moment and suggest some broad extensions that aren't direct continuations of this work.

I explore the origin of L-Systems and their early connection to simulations of living

[19] A; A $\rightarrow$ F[F+FA]–FA; θ = -120°; Iterations = 12

structures, especially those of plants. *The Algorithmic Beauty of Plants*,[20] or *ABOP*, written by Aristid Lindenmayer, their progenitor, and Przemyslaw Prusinkiewicz, has been an invaluable resource for me in the development of this thesis. In this section, I touch on some of the ways that this book guided parts of my research and helped me make judgement calls about what I'd be focusing on. There are some types of L-Systems that I don't include in my program. This is the point where I talk about those types of L-Systems and why I decided to not include them.

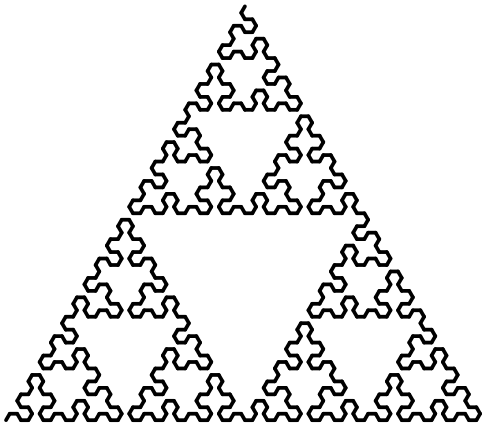


Figure 11: Sierpinski Gasket[21]

In Section 4.2 I discuss possible extensions of the program. Some of these extensions are little more than user interface changes while others might require underlying structural changes to function. In order to more competently think about the second of these goals, I reference the choices I've made regarding the content of *ABOP*. I address some of the considerations that might have to be made in order to introduce some of the structures in *ABOP* to my own program.

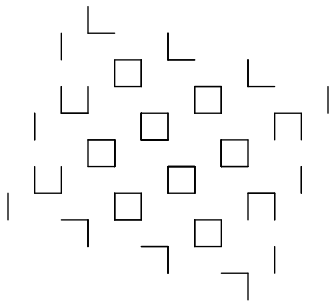
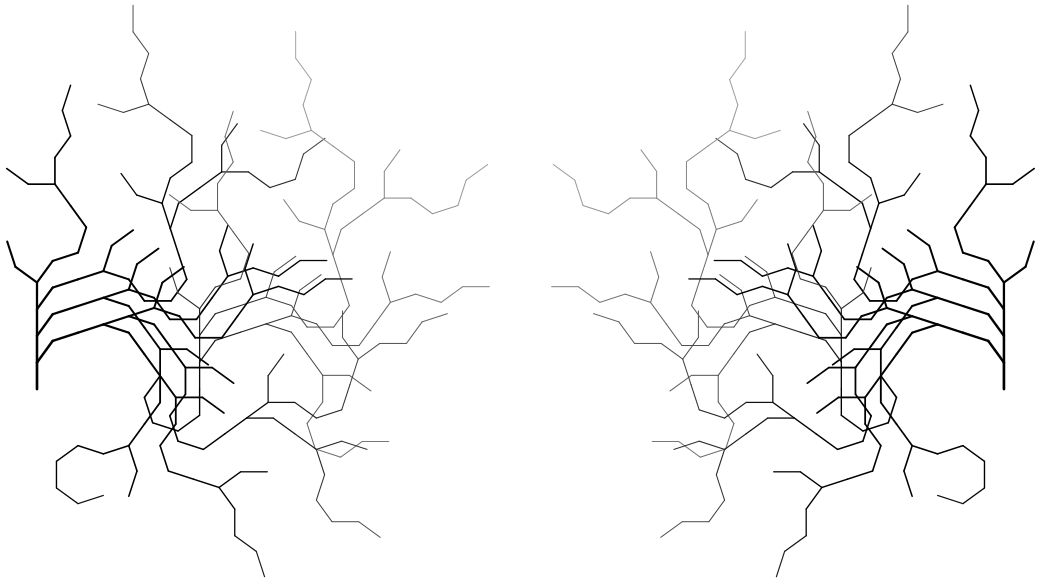


Figure 12: L-Swiss[22]

[20]Prusinkiewicz & Lindenmayer (1990)
[21]A; A → B+A+B, B → A–B–A; $\theta = 60°$; Iterations = 6 (A and B draw)
[22]A; A → F[+A]f–A; θ = -90°; Iterations = 7

This last section returns to explore more carefully bound L-Systems and articulate their capabilities. I discuss how every depiction of an L-System depends on choices made by a designer and that it's hard to understand an L-System in its raw text form. I talk about some different possible interpretations that can and have been used to make L-Systems comprehensible.

I then conclude the thesis with a brief summary of the understanding I've gained through this process, the concrete objects that I have created, and the exploration of the larger ideas that lie under this whole process. I summarize my ideas of how L-Systems might be investigated in the future and what there might still be to be learned about these structures. I end by encouraging my readers to mess around with my program and explore L-Systems and the larger world of generative art.



(a) Mirror Left[23] (b) Mirror Right[24]

Figure 13: Mirrored L-Systems

[23]A; A → F[A+FA–F]–F–FA+F; θ = 36°, ϕ = 18°; Iterations = 4; IW = 2, WC = 0.95
[24]A; A → F[A+FA–F]–F–FA+F; θ = -36°, ϕ = -18°; Iterations = 4; IW = 2, WC = 0.95

Chapter 1

What is an L-System?

This is an L-System:

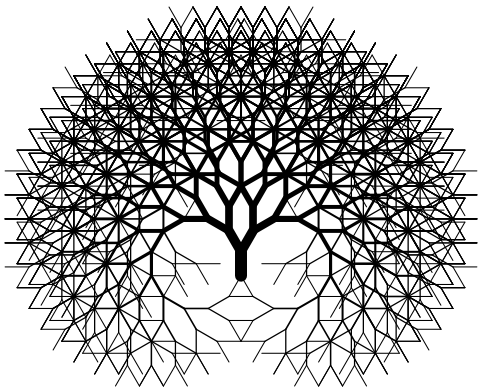


Figure 1.1: Magnificent Tree[1]

Well, it would be more correct to say that this is the *output* of an L-System. L-Systems are a *technique* with which one can describe a complex, changing system with a few letters and symbols. This chapter will cover a few main ideas. The first, and most central, is a discussion of the mechanics of how they work, how to interpret them, and how to create them. I'll illustrate particularly effective ways these systems can be used, how I've used them, (including some early experimentations before the central project,) and end with a lead-in to the central coding portion of this thesis, the L-Systems Explorer.

There are a few effective ways I've found to introduce L-Systems to people. I'll go through each of these different strategies to give you some solid footing before we really dig into the material.

1.1 Understanding through Definitions

Before I describe the structure of an L-System, I'd like to introduce you to the idea of a *string*. A *string* in Computer Science is a more succinct way of saying *a sequence*

[1] A; A $\rightarrow$ F[+A]–A; $\theta = 30°$; Iterations = 12

of symbols. Any word is a string, but a string does not need to be in any dictionary, and it can include essentially any character that can be written on a computer. This includes numbers, symbols like `&`, `%`, and `+`, as well as some other special characters. When I say "string", I mean a word that consists of some combination of these characters, for example: `DFS&2*|:09` or `!!n./?;`. A string can also be "empty", meaning that it contains no characters, though it is still a string. They may also be only one character long.

L-Systems have four essential parts. We'll break apart the recipe for Figure 1.2 to clarify our example. The first part is the `start` variable. It is usually just a character; in Figure 1.2 it is `A`. This will be the initial value of the structure, something akin to a "seed". The second part is a set of rules that describe how the L-System grows (which is explained more in the next paragraph). These rules are usually written like `A` $\rightarrow$ `AB` or `B` $\rightarrow$ `AA`, where each of these is a different rule mapping a character to a string. In Figure 1.2, the one production rule is `A` $\rightarrow$ `F+A`. In this thesis, I will call `A`, `B`, and other values with rules, "productions". Similarly, each of these rules is called a "production rule". As `+` does not have a production rule in Figure 1.2, it is therefore not a production. The third part of an L-System is the angle or angles that it uses to change its state. In Figure 1.2, this is θ `= -45`°. Some of the L-Systems I show in this thesis use a second angle ϕ in addition to the required θ. This ϕ establishes a baseline difference for values that use θ so that an L-System isn't just reduced to a line when $\theta = 0$. The fourth part is the `Iterations`, which is the number of times the production rules are applied to a string. This is `Iterations = 7` in Figure 1.2. Some L-Systems I made for this thesis use an entirely optional fifth part, which has two values and governs the width of the lines. Figure 1.2 does not use this part. Initial Width (IW) is the pixel width of the first line. Width Change (WC) is the multiplier applied to the line width of each successive line. Unless otherwise stated, the value for ϕ is 0 and the value for both IW and WC is 1.

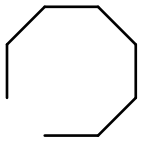


Figure 1.2: Almost an Octagon[2]

To draw an image from an L-System, we follow a set of carefully stated instructions. First, we save the start variable as a special string, we'll call it the L-string. If the start variable was `A`, then the L-string is also `A`. Then, we repeat a specific step some number of times determined by the value of `Iterations`. If `Iterations` is 5, then we will repeat the following step 5 times:

> - For every character in the L-string, if that character has an associated production rule, then that character is replaced in the L-string by its production rule. (For example, if the L-string is `A`,

[2] A; A $\rightarrow$ F+A; θ = -45°; Iterations = 7

and there is a production rule `A` → `BA`, then the new L-string after this step is `BA`.) If that character does not have a specified production rule, then it is implied that its production rule consists of itself.

With this simple step as a guideline, we can construct even the most complicated L-Systems. It might seem strange that this one rule is capable of such complexity, but that surprise is the mood that I'm encouraging you to see as a fundamental feature of these structures. There may be more opportunities to be surprised!

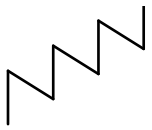


Figure 1.3: A Simple Squiggle[3]

While these images are intended to help describe the process of building an L-System, keep in mind that they are ultimately just *visualizations*. The L-Systems are only the *strings*. Thinking that L-Systems are the images might not be entirely accurate, but I admit that the image and the string are so closely tied that the distinction isn't all that important. Yet still, there *is* a difference.

1.2 Understanding through Images

L-Systems can be opaque if there's no visual output to latch onto. I think that one of the best ways to engage with these structures is through connecting the images with the strings that describe them. One can see that everything in the image has an origin in the string, and that there is nothing lost between the image and the string. I demonstrate alternative visualization strategies in Section 4.3.1.

Figure 1.4 shows the appearance of an L-System as the number of iterations increases from 0 to 3. There is a sense that the L-System is "growing" here, fittingly tied to their origin as plant-modeling software. The text below each set of images is the L-string that constructed those images.

This L-System's info: "start: A; rules: A → –FB, B → +FA; angle: $\theta = 45°$".

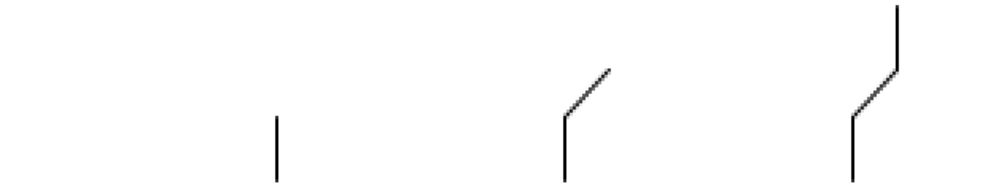
(a) A (b) –FB (c) –F+FA (d) –F+F–FB

Figure 1.4: First Simple L-System

[3] A; A → F+B, B → F–A; θ = -122°; Iterations = 7

Figure 1.5 uses the same start variable and angle, but different production rules. Each image still displays the appearance of the L-System as the number of iterations goes from 0 to 3.

This L-System's info: "start: A; rules: A → F–B, B → F+A; angle: $\theta = 45°$".



(a) A (b) F–B (c) F–F+A (d) F–F+F–B

Figure 1.5: Second Simple L-System

Look at the L-System in Figure 1.6 and see how the different productions can be seen in the output. (The long straight sections are generated by `A` and the branch joints come from `B`.) The starting point is the open-ended branch in the middle of the image.

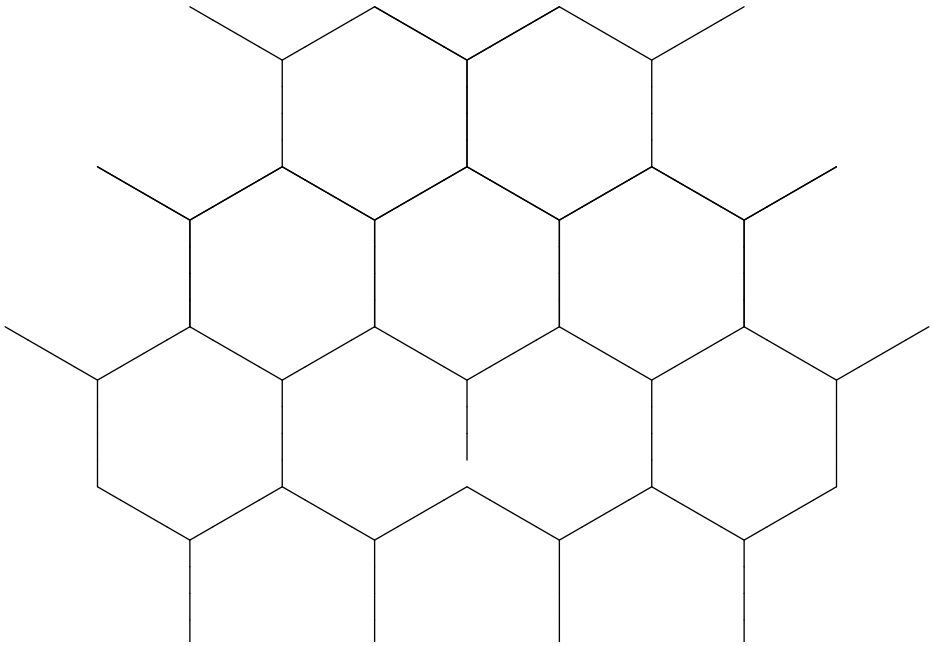


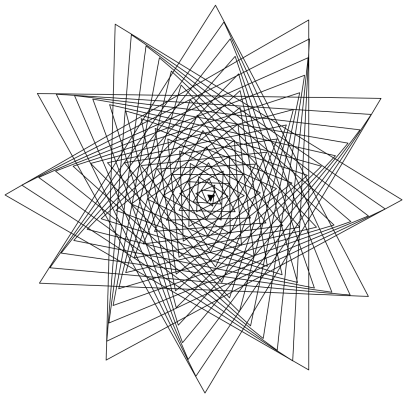
Figure 1.6: Distinct Productions[4]

[4] A; A → FFFB, B → +[– –FA]FA; $\theta = 60°$; Iterations = 11

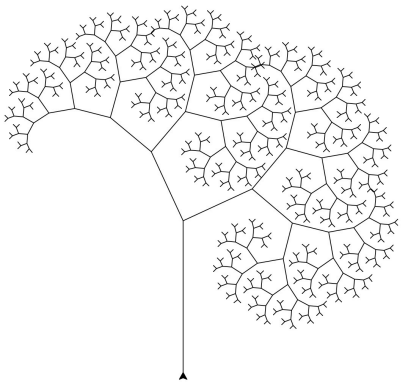
1.2.1 Turtle Graphics

Figure 1.7: The "Turtle" used in Turtle Graphics

My early attempts to make L-System drawing programs all utilized a concept called "Turtle Graphics".[5] Turtle Graphics is a simple program that is often used in introductory Computer Science classes to help get students familiar with the basics of coding. It's used in *The Algorithmic Beauty of Plants* to draw most of their L-Systems. I've also seen it used in other L-System implementations online. Turtle Graphics operates by reading a series of instructions and drawing according to those instructions. This is particularly applicable for L-Systems, where each symbol can be associated with a specific operation, such as drawing a straight line or turning to face in a different direction. Even while making my Explorer from the ground up, I've kept in mind the Turtle Graphics strategy.



(a) Simple Star



(b) Fern

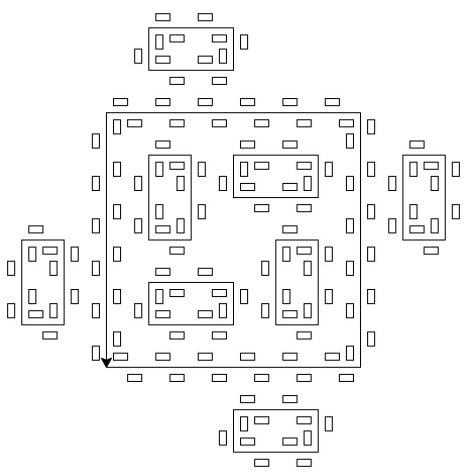
Figure 1.8: Images generated with Turtle Graphics

The central feature that Turtle Graphics uses to great effect is "state". "State" here means "condition" or "status". The state of the "Turtle" is its position, rotation, and whether its "pen" is up or down. This last part controls if it draws as it moves. With an x-position, a y-position, and an angle, the Turtle is uniquely placed in its window. By manipulating these values with the code, the Turtle turns abstract code into clear visualizations. This is also particularly powerful because it physicalizes the

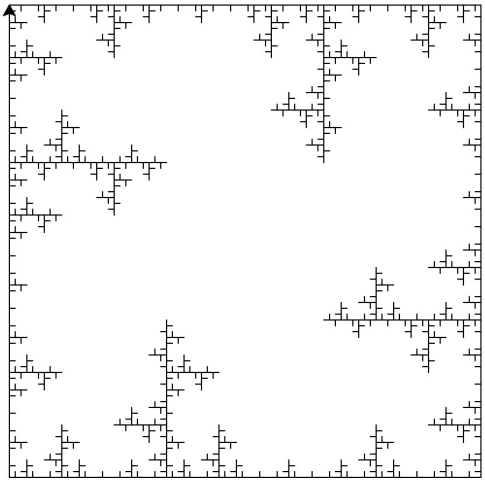
[5]Papert (1980)

process of coding that's often purely digital and accessible only through difficult-to-engage-with abstraction.

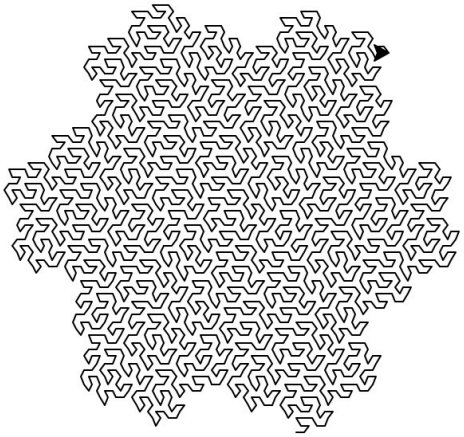
Figure 1.9 shows some fractals created with Turtle Graphics and L-Systems. Each of these could also be created with the L-System Explorer I introduce in the next chapter. L-Systems lend themselves particularly well to constructing fractals, due in part to their own recursive nature, but without requiring understanding of the concepts of functions or the often vexing concept of recursively defined functions.



(a) Islands & Lakes[6]



(b) Frame[7]



(c) Space-filling Hex[8]

Figure 1.9: L-Systems visualized with Turtle Graphics[9]

[6] F–F–F–F; F → F+f–FF+F+FF+Ff+FF–f+FF–F–FF–Ff–FFF, f → ffffff; $\theta = 90$; Iterations = 2

[7] F–F–F–F; F → FF–F– –F–F; $\theta = 90$; Iterations = 4

[8] F_L; $F_L \rightarrow F_L{+}F_R{+}{+}F_R{-}F_L{-}\ {-}F_L F_L{-}F_R{+}$, $F_R \rightarrow {-}F_L{+}F_R F_R{+}{+}F_R{+}F_L{-}\ {-}F_L{-}F_R$; $\theta = 60$; Iterations = 4

[9] The L-strings for these images were given in ABOP (Prusinkiewicz & Lindenmayer (1990))

Chapter 2

The L-Systems Explorer

In order to share my understanding about and my excitement for L-Systems, I wrote a program that allows users to interactively and graphically edit L-Systems and immediately see them visualized. My goal was to create a program that would reduce the barrier to entry of understanding these systems. My first explorations into making this space more accessible were done through turtle implementations in a basic Python program and later in a Blender program.[1] I'll talk about these in more detail later in the thesis. Both of these programs essentially involved *teaching the users to code*, and I wanted this system to be more accessible than that. In conversation with my advisor and through looking at resources online for projects like mine, I found *Processing*.[2]

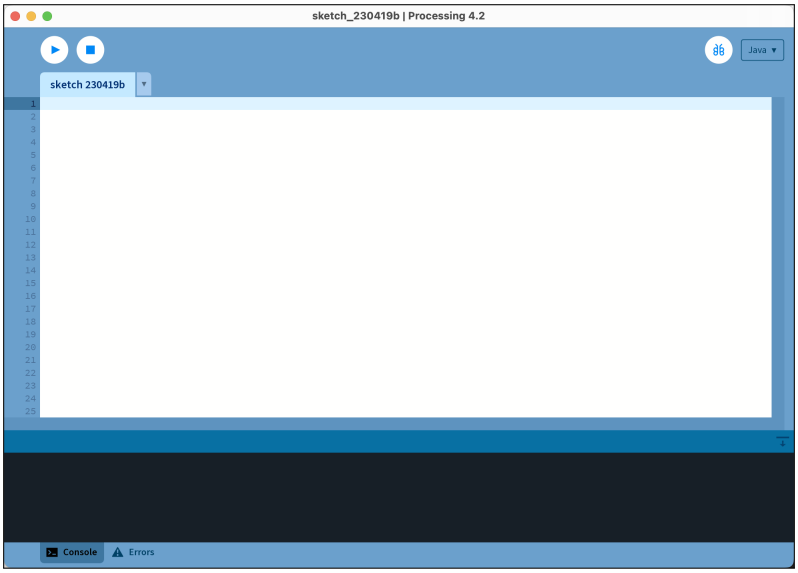


Figure 2.1: A blank Processing window

[1] Blender (2018)
[2] Reas & Fry (2014)

2.1 What is *Processing*?

Processing is a programming language and development environment written in Java and optimized for image generation. It was released by Casey Reas and Ben Fry in 2001 and has since found popular use as an art and educational tool. I chose to work with Processing because it was easy to use and appealed to my sensibilities as it is an open-source, free software. It also engages with the act of coding in the same way that artists have traditionally engaged with their mediums, which I also appreciated. In other programming languages, it often takes some time to go beyond just printing `hello world`. The first programs that people write in Processing produce concrete objects that can interact with the user, like a square that changes color when you hover over it. In Processing, coding *is* art.

To download and experiment with Processing yourself, go to Processing.org and click the download button. There is an expansive part of the Processing website dedicated to introducing newcomers to this art form, which you can access from the tab titled "Learn" near the top of the page.

2.2 Program Development

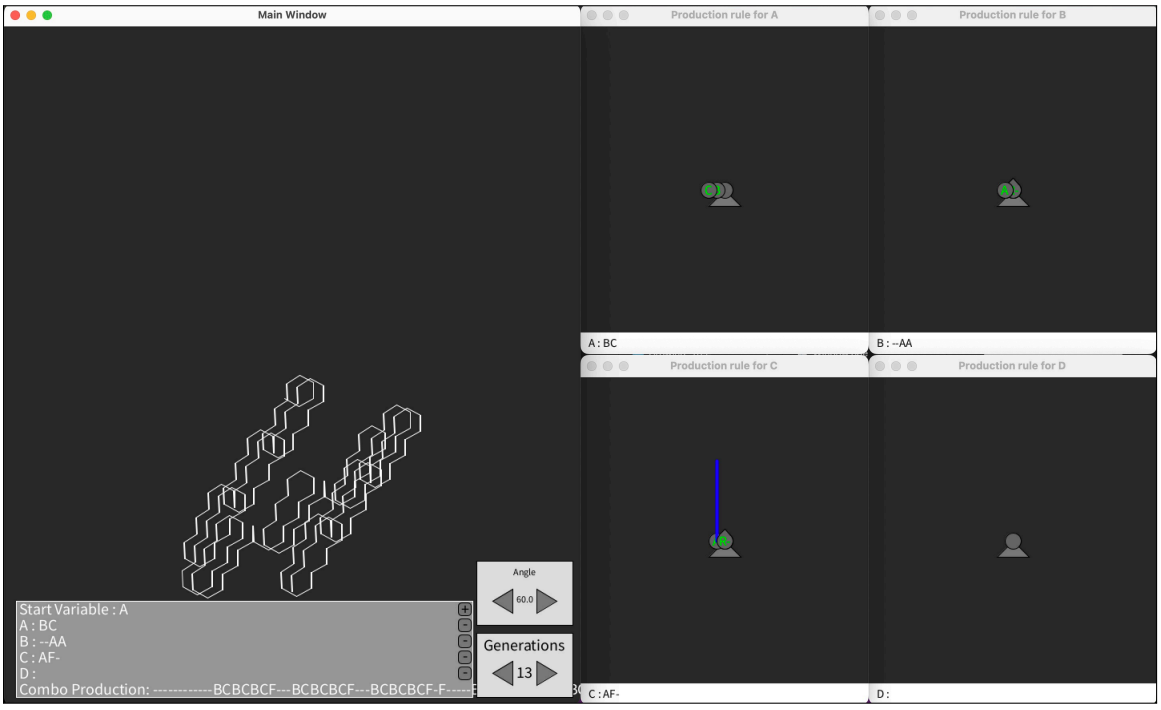


Figure 2.2: An early version of the Explorer

The development of the L-System Explorer started with a series of drawings describing the different features I was aiming to include. I wanted everything to be able to be accessed without touching the keyboard. There have been a few moments since I started development on this project where things could have been easier if

typing was allowed, but so far I only expect one of those uses to actually need the keyboard. (I was imagining that if someone wanted to input an L-System that they already had on hand, it would make sense for them to be able to type it in. [put this in a footnote]) Once I had a set of features in mind, I began to write the program.

I built the windows first. There is a main window that displays the current visualization of the L-System and one smaller window per active production. These sub-windows are created and removed to keep in line with the number of productions. In the basic processing version of the Explorer, the windows for the productions are able to be moved independently from the main display window. From there, I began to build up the framework of the Explorer itself. Keeping the goal of approachability in mind helped prevent me from making the program too dependent on any specific programming experience. Although this thesis adds depth and context to the program, I hoped to make the Explorer self-explanatory enough to operate without having to read this document. The structure of the Explorer changed throughout its development and continues to change, but I'll describe the form that it has taken at the time I'm writing this.

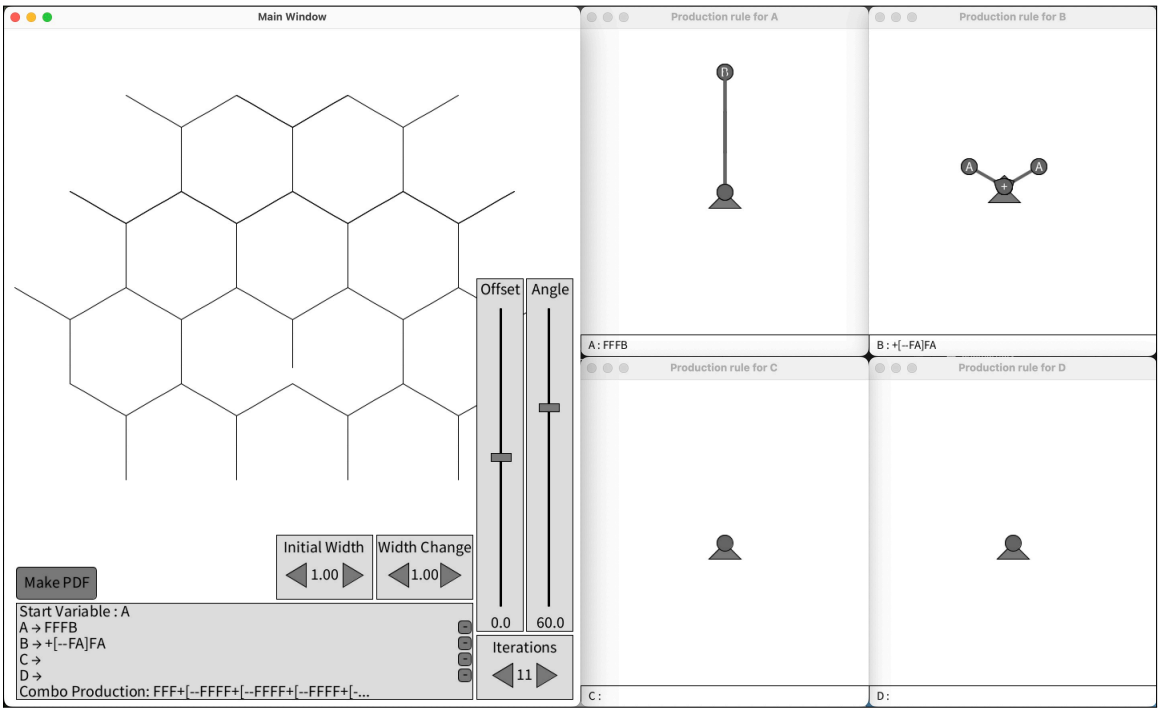


Figure 2.3: The current version of the Explorer

2.3 How it works

I talk about the details of the L-System Explorer in this section. I would encourage you to read on if you're interested in taking a look under the hood about how I brought it to life. However, there won't be any new concepts regarding L-Systems

introduced here. To move past this and get a summary of how to use the program, go ahead to Section 2.4.

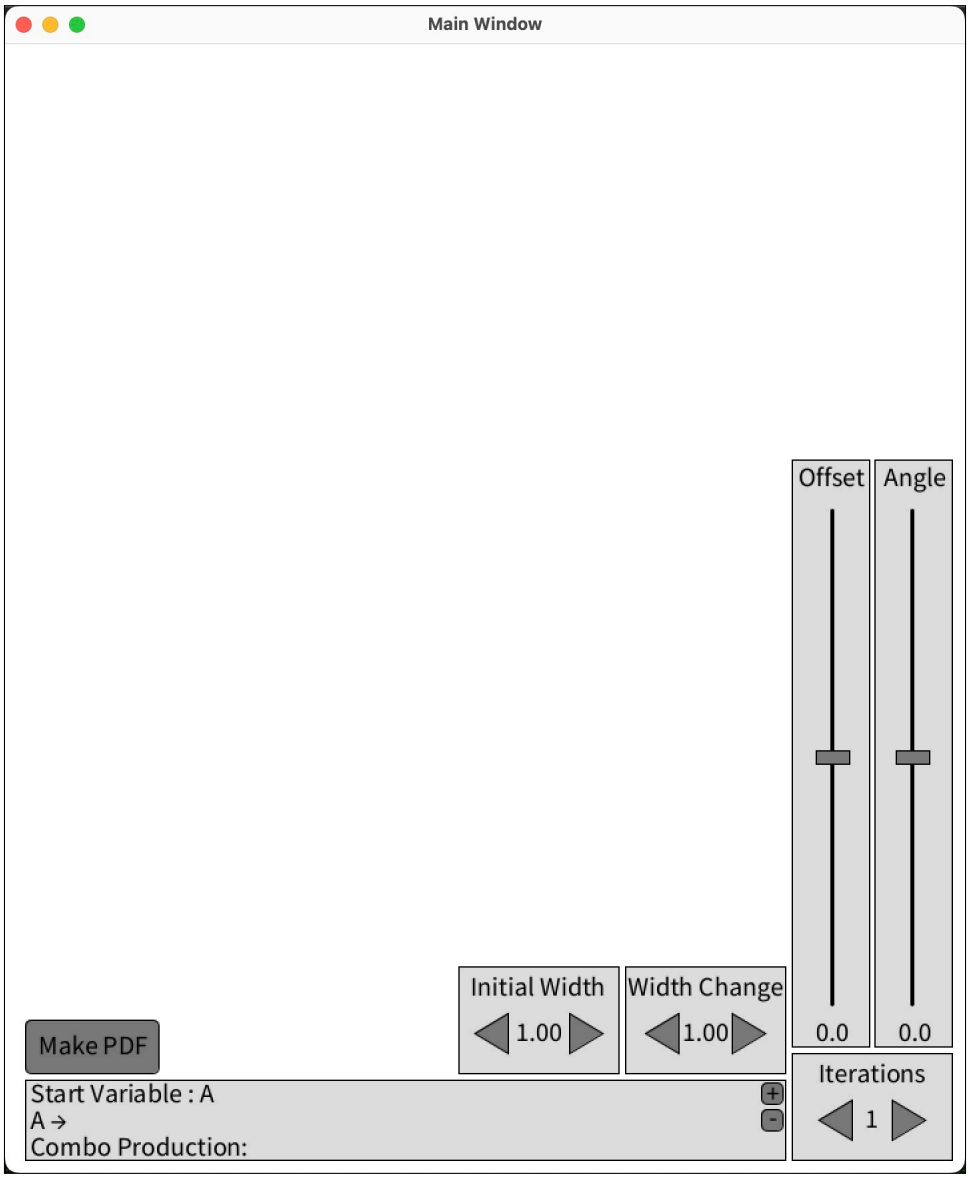


Figure 2.4: The Main Window

The main window is initialized upon startup along with some number of sub-windows, one for each production that the program starts with. More productions can be added, and more windows are created for productions when needed. Windows are similarly removed when their production is removed in the main window. Each sub-window contains a reference to a 'starting' triangle that I named the `startTriangle`. This is an instance of a class that can be moved up to the boundaries of its window. The `startTriangle` has a reference to the `startNode`, which is the root of a linked list structure of nodes. I've made a class to describe the nodes as well. When the `startTriangle` is moved, the `startNode` is moved similarly, and then recursive calls

down the node tree move all the nodes to the correct positions. Shown in Figure 2.6, `startTriangle` and its closely-tied `startNode` provide the foundation on top of which each of the production rule displays are constructed.

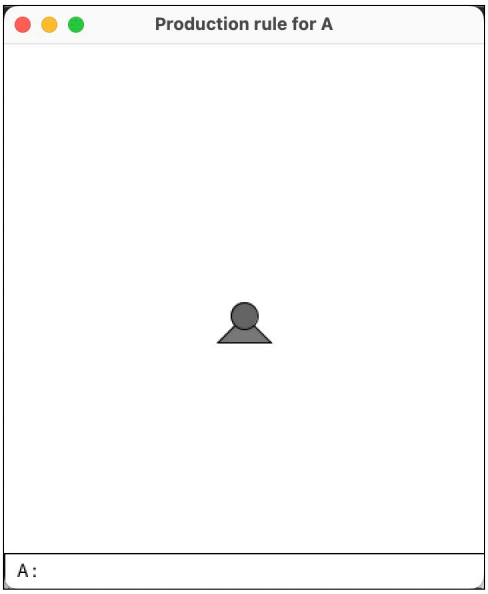


Figure 2.5: The sub-window for the production rule of A

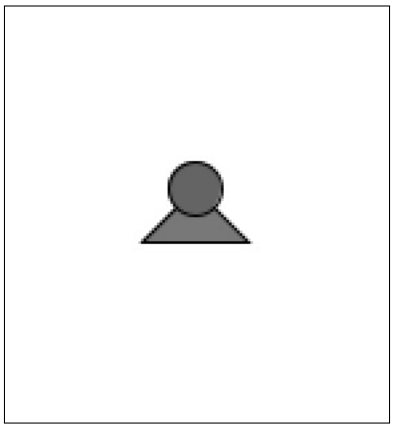
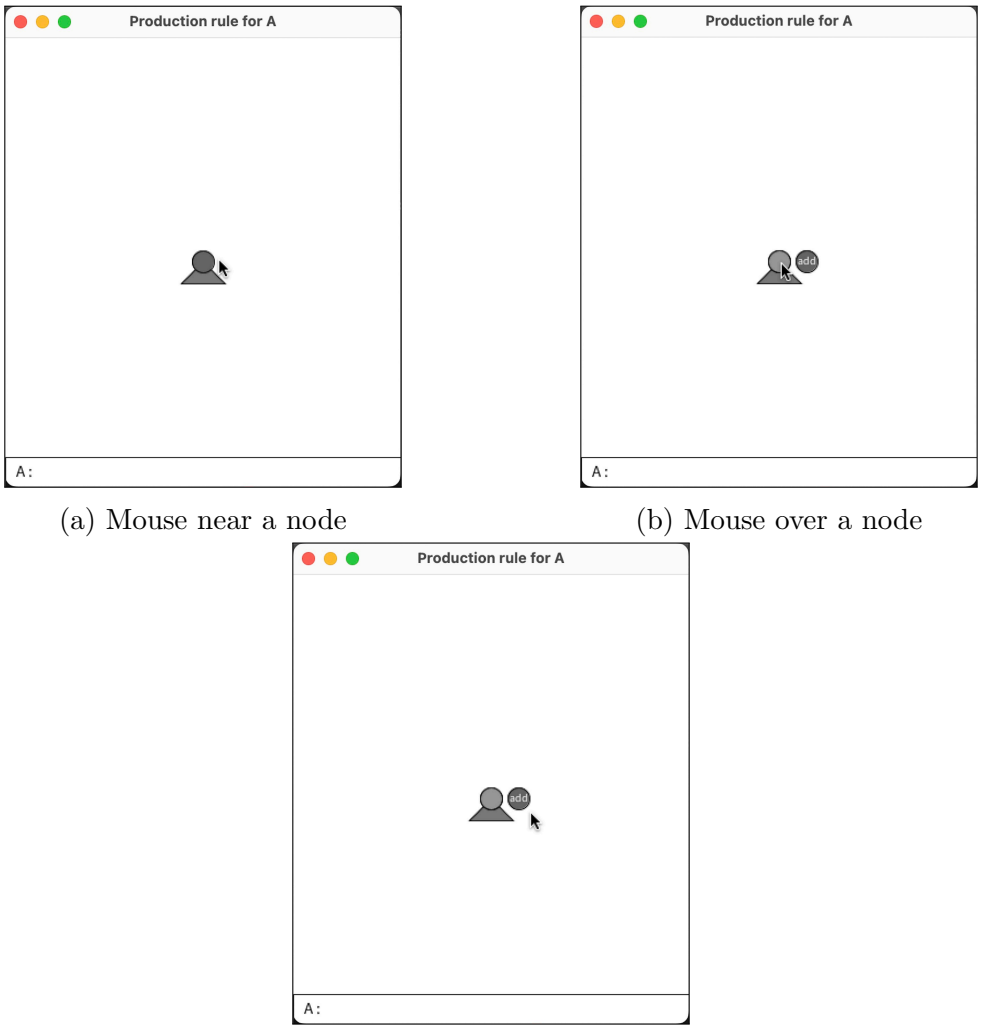


Figure 2.6: The `startTriangle` and the `startNode`

There are five subtypes among the node class, each node belongs to only one of these subtypes. These aren't subclasses, the different nodes have some different functionality, but they're only differentiated with a string. The `startNode` is the only element of the `start` subtype; I use this as a signpost to stop calls from nodes to their parents from going too far. There are `production` and `rotation` nodes, the main difference being that rotation nodes have a point that shows which direction they're oriented. The fourth subtype is `forward` nodes. These are displayed as lines, unlike the other types of nodes, which are displayed as circles. The final subtype is `option` nodes. These are a special type of node with some unique characteristics. They have

a pointer to an `optionParent` node, which may be a member of any subtype of node, including another option node. They will only appear if their `optionParent` node is being hovered over by the mouse or their `optionParent` has recently been hovered over by the mouse and the mouse has not left its vicinity or the vicinity of any of its `option` nodes. This is pictured in Figure 2.7.



(a) Mouse near a node

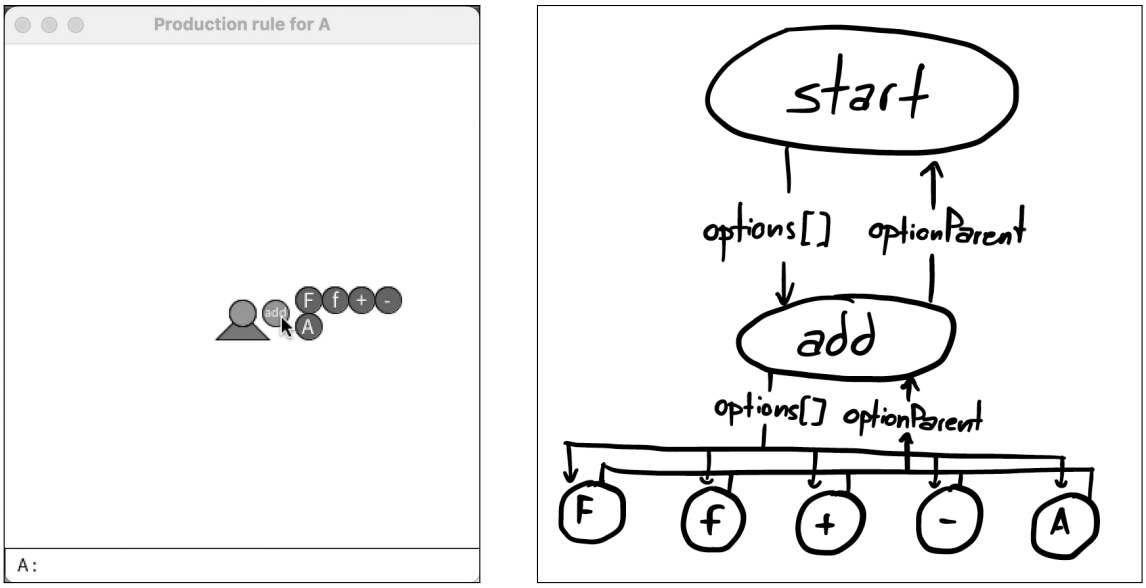
(b) Mouse over a node

(c) Mouse nearby after being over a node

Figure 2.7: Mouse interactions with nodes

Each node of any subtype can only interact with its neighboring nodes and the `option` nodes that refer to it as an `optionParent`. This intentionally limited functionality is created by giving each node a pointer to one other node in the direction of the `start` node. This is the node's `before` node. It also has an array of `after` nodes, which consists of all the nodes that consider this node to be their `before` node. The remaining reference in a node is the array of `option` nodes that consider this node to be their `optionParent`. Figure 2.8 depicts this parent and child structure.

Beyond the references, each node contains information that defines its state. One part of a code's state is the value `over`, which is a boolean for whether the mouse is



(a) Multiple nodes (b) The relationships between the nodes

Figure 2.8: The `option` nodes of the `option` node named "add"

over the node. Another important variable is `press` which is a boolean for whether the mouse is clicking the node. The variable that has the most control over the state of the program is called `activated`. `activated` is a boolean with a complicated condition. It corresponds to whether the mouse is currently over the node or was recently over the node and has not moved too far away.

If `activated` is true for some node, then some user interface logic ensures that no other node is activated or can become activated while this node is activated. This happens through a recursive method that is called in a node's `before` node. The message then propagates upwards until it reaches the `start` node, where it sets a boolean `somethingActivated`. In order for a node to be set to activated, it must first ask its `before` node if `somethingActivated` is true. This question is passed to the `start` node as well. This process is shown in Figure 2.9. `option` nodes are the exception to this. They will only ever be `activated` when their `optionParent` is already `activated`. Each node also tracks its position and orientation, information that is referenced by its `after` nodes to ensure that every node is in the correct state.

The user is able to click on the `option` nodes of an `activated` node, and if the node has an associated action and some conditions are fulfilled, the action happens. Usually this action is the creation of a new node whose `before` node is set to be the current node (the node whose `option` node was just clicked). The other case is that the `option` node that was clicked will trigger the deletion of that `option` node's `optionParent` node. Deletion is described in more detail in Figure 2.10. For both of these possibilities, the current node's `after` array and the display are both updated.

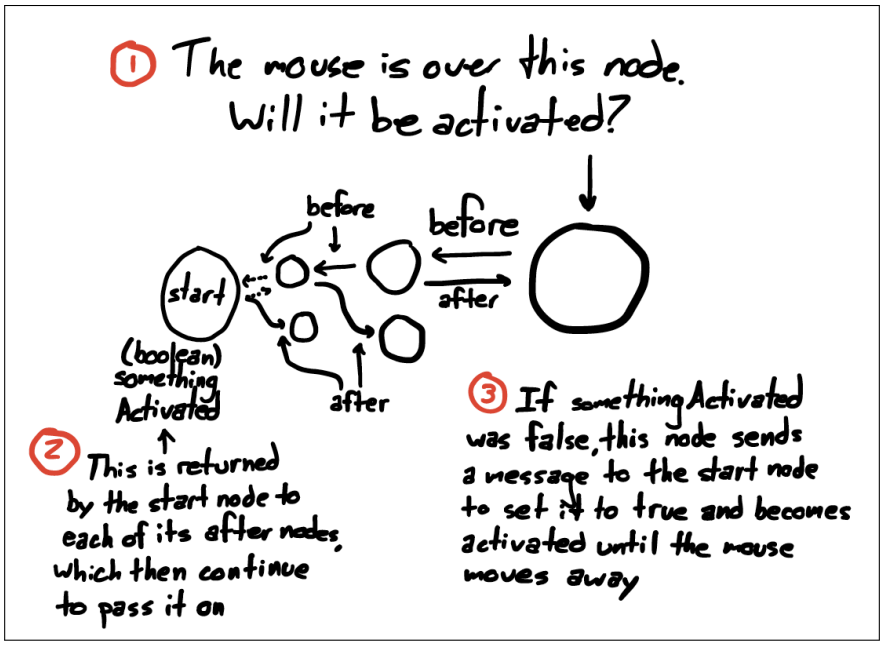
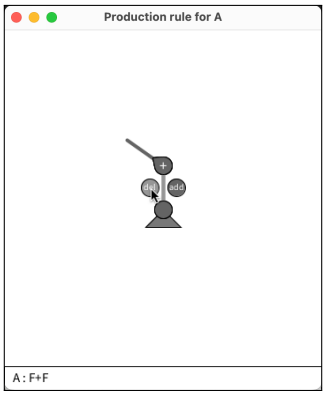
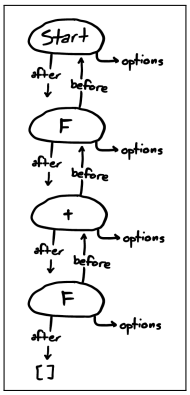


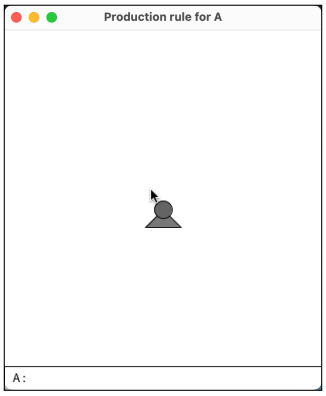
Figure 2.9: The recursive nature of `activated`



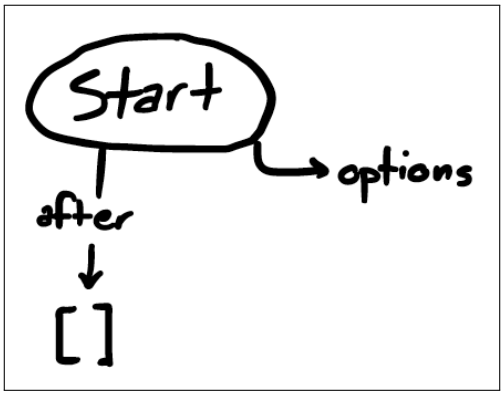
(a) Before deletion



(b) Before deletion (behind the scenes)



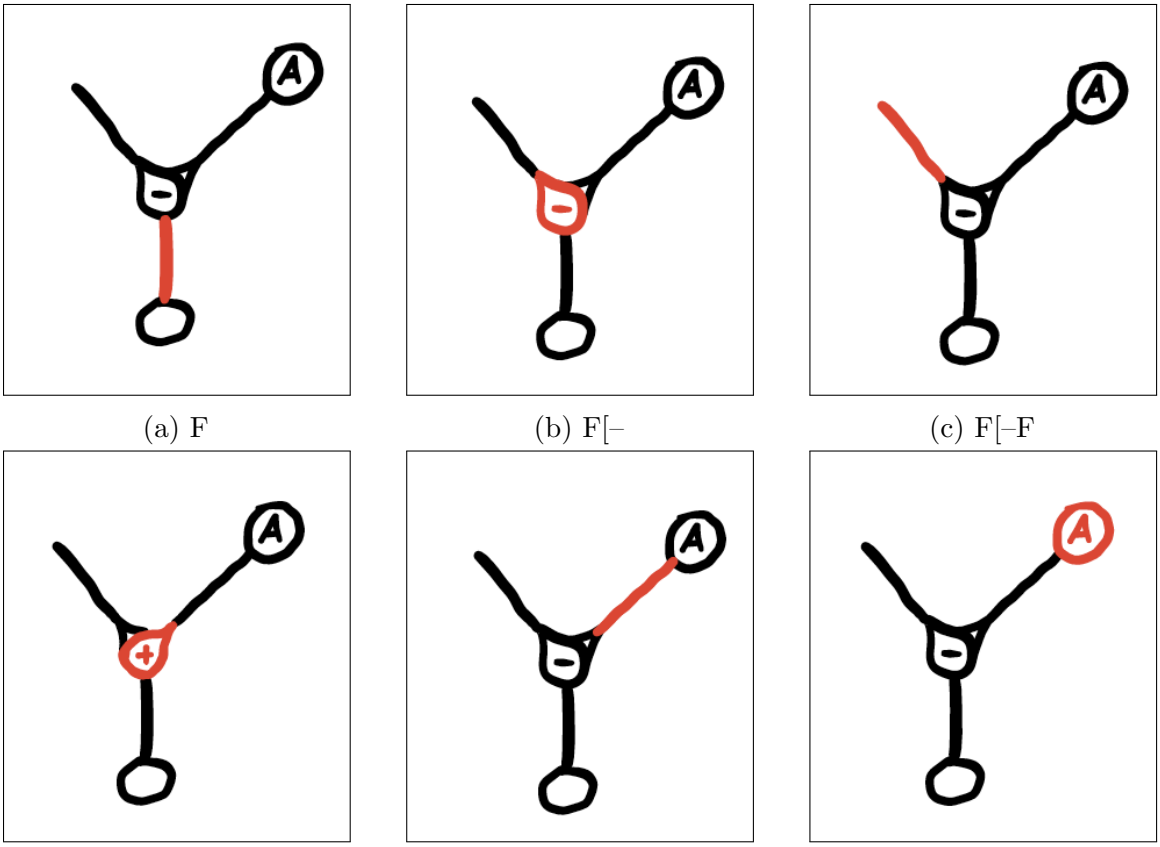
(c) After deletion



(d) After deletion (behind the scenes)

Figure 2.10: Effects of a node deletion

Each sub-window has a function to turn its tree of nodes into a string that uniquely describes it. Figure 2.11 shows this process of depth-first search, where each branch of the tree is explored down to its end, adding elements to the string as it does. The tree is explored through the `before` and `after` linkages. Any time the tree branches, the function adds an `[`, and any time the tree finishes reading a branch, the function adds a `]`. These brackets allows complex branching structures to be accurately described in only one string. They are used by the Explorer internally when interpreting the L-string to make an image, which is described more in Section 2.3.1. Most nodes have an associated character that is added to the string whenever the node is seen on the tree. `rotation` nodes have either a `+` or a `-`. `forward` nodes have either an `F` or an `f`. `production` nodes have the character of their associated production. (These might be `A`, `B`, etc.) The `start` node and `option` nodes are not read by the function and so do not have associated characters. At the end of this process, the sub-window has a string that is the production rule for the production of that sub-window.



(a) F (b) F[– (c) F[–F

(d) F[–F]+ (e) F[–F]+F (f) F[–F]+FA

Figure 2.11: The steps the Explorer takes to create a production rule

Similar to how each node's `after` nodes will have a way to access their `before` node, each sub-window is able to access some information from the main window. The main window is also able to change some information in the sub-windows. However, there is no way for the main window to directly affect the `startTriangle` or

nodes within any of the sub-windows. It calls a method called `getProdRule` in each sub-window to do that. Some information must still be transferred between layers, information such as the production rules. The main window accesses the production rules from each sub-window's `getProdRule` method. It then applies the L-System recursive replacement process described in section 1.2 a number of times equal to `Iterations`, a value which can be changed by the user with the buttons in the main window. The output of this process is the L-string, which is used to draw the L-System.

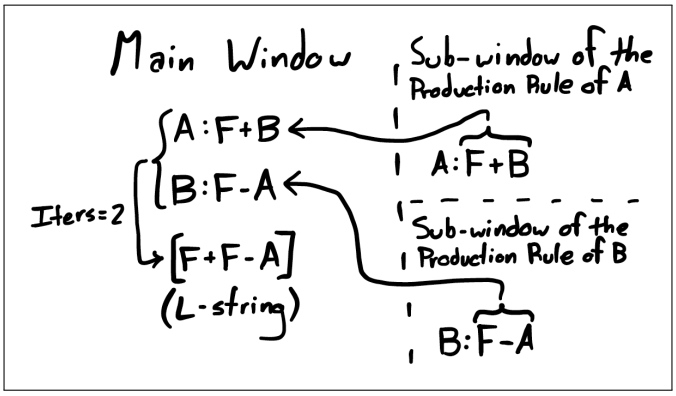


Figure 2.12: The movement of the production rules between windows

2.3.1 Rendering L-strings

In the process of drawing the L-System, the Explorer reads the L-string character-by-character and performs actions depending on that character. The Explorer interpreting the L-string acts very much like the turtle described in Section 1.2.1. It adds a data structure called a "stack", to the turtle's state. The structure of this stack is shown in Figure 2.13. The Explorer only interacts with the top item on the stack, and it can choose to read it or to remove it. It can also add a new item to the top of the stack. In this case, the kind of item that we're putting on the top of the stack is a snapshot of the current state.

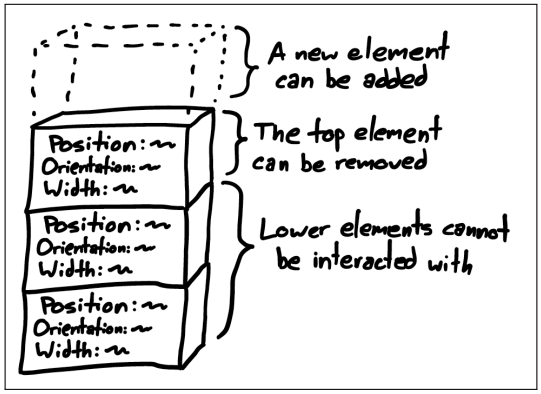


Figure 2.13: The structure of the stack

When the Explorer processes a `[`, it puts the current state onto the stack. Whenever it processes a `]`, it sets the current state to the state that is on top of the stack, then it removes the state on the top of the stack. If it processes an `F`, it goes a given distance in the direction it's facing and draws a line between its starting and ending point. When it processes an `f`, it goes that distance in that direction, but doesn't draw a line. A `+` adds the given angle value to its stored orientation. `{` does the same, but subtracts the angle from the orientation instead. The angle can be changed by the user to quickly see different possible views of the L-System. Reading a production like `A` or `B` doesn't change its state, nor does it draw anything. It won't change its state if the character that it reads is not among these ones.

While this could certainly be less complicated, I am please to have been able to make a system that feels to me to be relatively sleek and clear in its purpose, while hiding the more complicated mechanisms. It was educational to engage with the intricacy of building a user interface. This was an exercise in articulating exactly what I want from the program. It was satisfying being able to see where there were mistakes in my understanding, as the program would very explicitly tell me.

2.4 Using the L-Systems Explorer

The L-Systems Explorer has what I feel is necessary to get a basic understanding of L-Systems.

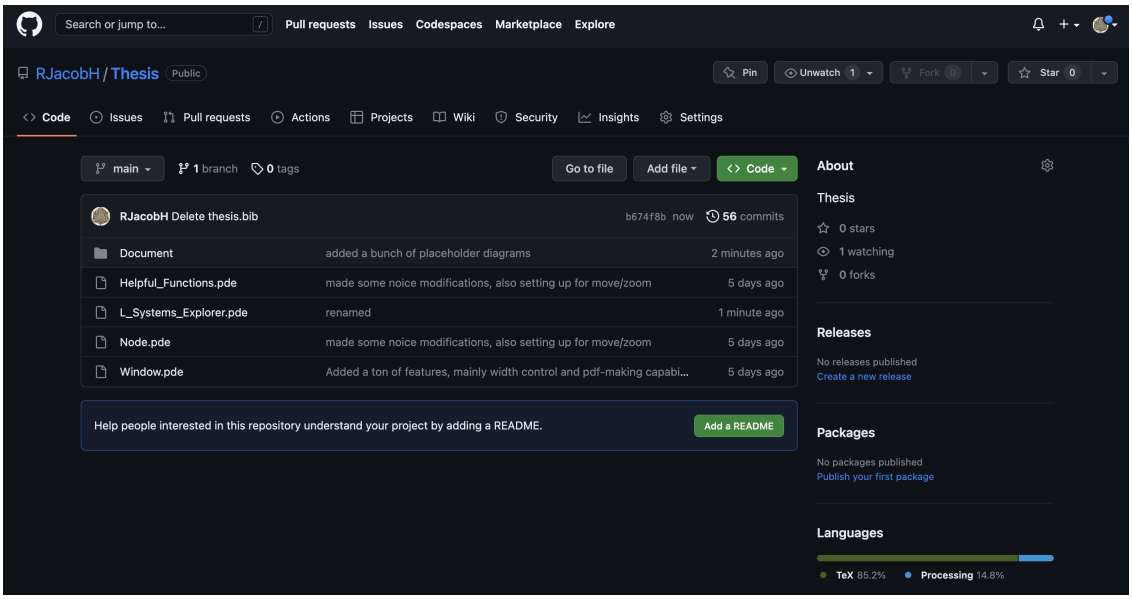


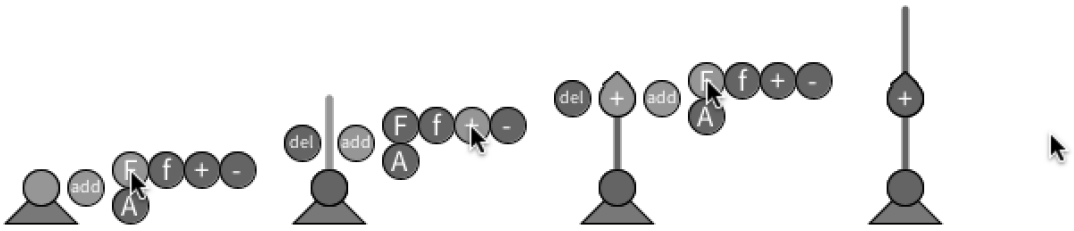
Figure 2.14: The Github page

To run this program, you need to download both the program and the framework to support it. The program can be downloaded at *github.com/RJacobH/Thesis*. The page should look something like Figure 2.14. You will need the four files, `L_Systems_Explorer.pde`, `Helpful_Functions.pde`, `Node.pde`, and `Window.pde`. Download these and put them in the same folder. Next, go to *processing.org* and download

Processing. At the time of writing, this program works in Processing 4.2. From here, you can open any of the four `.pde` files you just downloaded and you should something that looks like Figure 2.15. Click the play button in the top left corner and get started exploring!

Figure 2.15: The code for the program

2.4.1 How to Explore

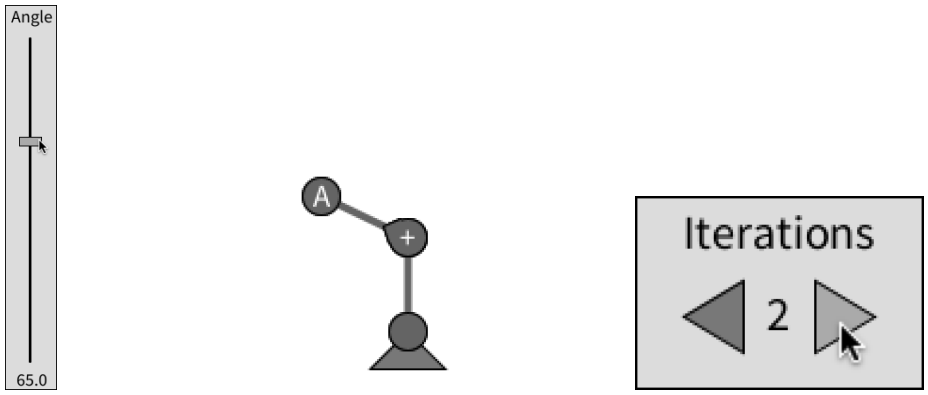


(a) Over F (b) Clicked F, Over + (c) Clicked +, Over F (d) Clicked F

Figure 2.16: Clicking Nodes

The first time you use the L-Systems Explorer, hover your mouse over any of the circles in one of the smaller windows. When you do this, you'll see several more circles appear, each with a different symbol inside of them. These are buttons. Click on the button that says `F` and a line will appear on the small screen as well as the larger screen. Hover over the new line in the small window and click the `+` button. This creates a new shape in the small window. Hover over this new shape and click the `F` button that appears. This will make another line appear in both the small

window and the big window. You can see the production rule of the small window running along its lower edge. This process is also illustrated in Figure 2.16. Also notice how the title of the window is "Production rule for A". Creating these elements has changed the production rule for `A`.



(a) Moving the slider (b) Adding a new `A` (c) Incrementing the Iterations

Figure 2.17: Adjusting the Angle

Look for a slider that says "Angle" on the right side of the page. Moving this slider, you'll see your lines move. Try going back over to the small window and hovering over the second line you drew. It doesn't need to be vertical. Click the button that pops up that says `A`. Now find the box that says "Iterations" in the bottom right corner of the large window. Click the triangle that points to the right. You'll see the number in the box go from 1 to 2, and the size of your line in the big box get much longer. Look at Figure 2.17 if you get lost. Try clicking the rightward-facing triangle a few more times. Also try moving the angle slider. Look at what's happening!

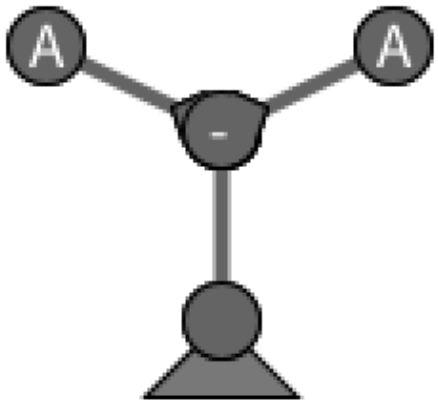


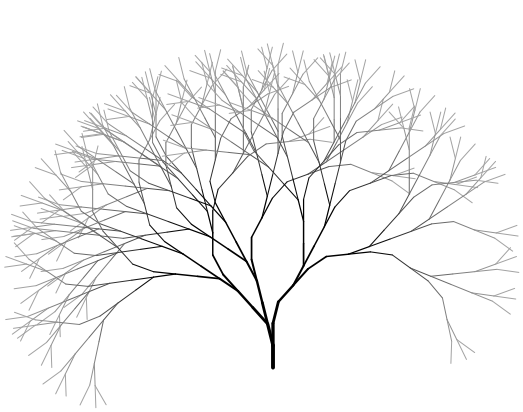
Figure 2.18: Adding a new branch

From here, you can go back to the small window and hover over the first line you drew. Click the `-` button that appears. Now hover over the new `-` node and click its `F` button. Make sure that you were hovering over the `-` button and not the `+` button that is in the same space. Hover over this new line and click its `A` button. At this point, the production rule window for `A` should look something like Figure 2.18. If your Iterations box says 1, increase it. Move the angle slider and see what's happening. Move the slider that says "offset" and see how that changes things. Try changing the production and seeing what interesting shapes you can produce.

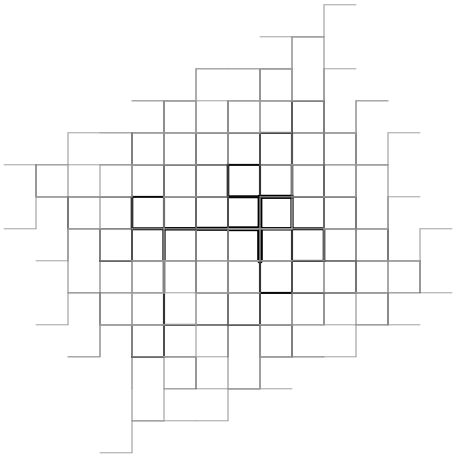
Figure 2.19: Adding a new production rule

The last thing to learn is how to add more productions and production rules. Look for a box that says "Productions" like the one in Figure 2.19. Click on the small `+` box. A new small window should appear. The original small window continues to correspond to the production rule for `A` and this new window will correspond with the production rule for the new production `B`. You can reference `B` within the `A` production rule window and vice-versa. One final point, you can stack rotation nodes by hovering over a rotation node (`+` or `-`) and clicking the button for the same type of rotation node. Choosing the other direction of rotation node will cancel its effect but choosing the same type will double the effect. This can be repeated any number of times. Some of the most interesting L-Systems I've found have used this strategy.

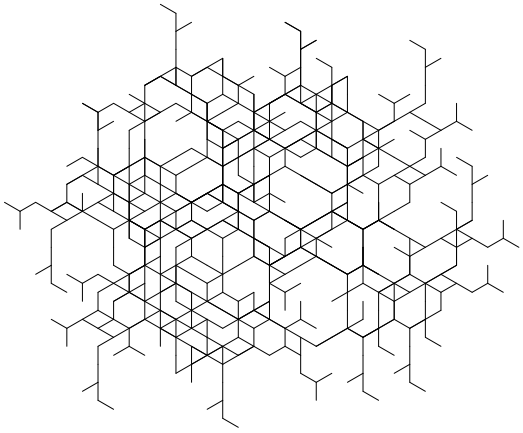
There are several other tools that you can use to extend your experience of the program. Try adjusting the Initial Width and Width Change panels, then varying the angle and offset so that you get something that looks more like a real tree. You could set the angle to 90° and see how the central branches vary in thickness so much from the branches nearer to the edge. Try creating an interesting L-System then clicking the 'Make PDF' button. You can print out your creations in vector graphics quality! Figure 2.20 shows some directions you can explore.



(a) Organic[3]



(b) Condensed Organic[4]



(c) Curves and Edges[5]

(d) Sharper Curves and Edges[6]

Figure 2.20: Some L-Systems to create and change

[3]A; A → F[+F[FA]++FA]F–F– –A; $\theta = 14°$; Iterations = 5; IW = 3.5, WC = 0.85

[4]A; A → F[+F[FA]++FA]F–F– –A; $\theta = 90°$; Iterations = 5; IW = 3.5, WC = 0.85

[5]A; A → F[–FB]F+FA, B → F[+F[–FA]+FB]–C, C → FFB; $\theta = 60°$; Iterations = 7

[6]A; A → F[–FB]F+FA, B → F[+F[–FA]+FB]–C, C → FFB; $\theta = 84°$, $\phi = 6$; Iterations = 7, IW = 10, WC = 0.85

Chapter 3

Contribution

A primary goal of this work is to make L-Systems, and by extension Generative Art and Computer Science broadly, more approachable. This was not my only goal. I want folks who haven't used computers as an artistic tool before to create things in this medium. I even want my peers, who have been immersed in the depths of CS for years, to be able to approach the discipline with a fresh awe and changed understanding. I want everyone to be amazed and inspired by the possibilities and to feel capable of creating their own new work. I feel that these are all very attainable without having to engage with the complexity of modern design software.

Keeping in mind that L-Systems were proposed back in 1968, I was worried that with 55 years to develop these ideas I wouldn't be able to tread on any new ground. This is true in many respects, and I re-emphasize that a main goal of this work is to raise awareness of L-Systems and to bring together some sources that discuss these ideas at the level of an undergraduate thesis. However, this work goes a little beyond that. There have been few implementations that make L-Systems easy to play with. Even those who have brought L-Systems to Blender still only make them available to those who know Blender.[1]

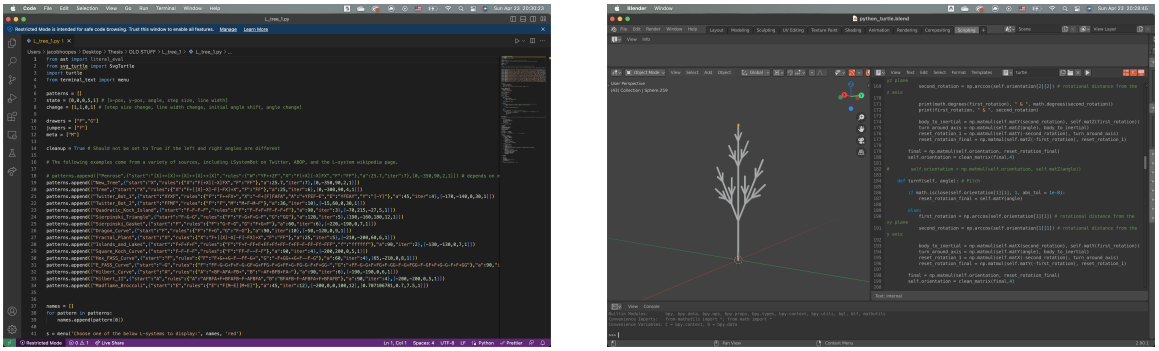
This Blender work I mention is incredible, and I found it a great source of inspiration, but our projects have fundamentally different goals. Leopold's work does reduce the amount of typing required, thanks in part to Blender's sturdy UI, but there is still a disconnect between the strings and the images. I haven't heard of any way to gain an intuition with these systems that isn't locked behind years of engagement with computer science, despite a central part of their appeal being simplicity.

My central contribution has been the development of the interactive L-System Explorer. I hope that it's clear that this project will be able to engage with people, no matter their experience with computer science. I hope that this ability to craft L-Systems will help those who are interested to overcome any wariness about engaging with digital machines as an artistic medium. Despite computer science being about much more than pretty pictures, I feel that a warm welcome will help those who were once strangers come into this place as their home. These pictures and this thesis are my attempt at creating a warm welcome.

[1] Leopold (2017)

3.1 Related Work

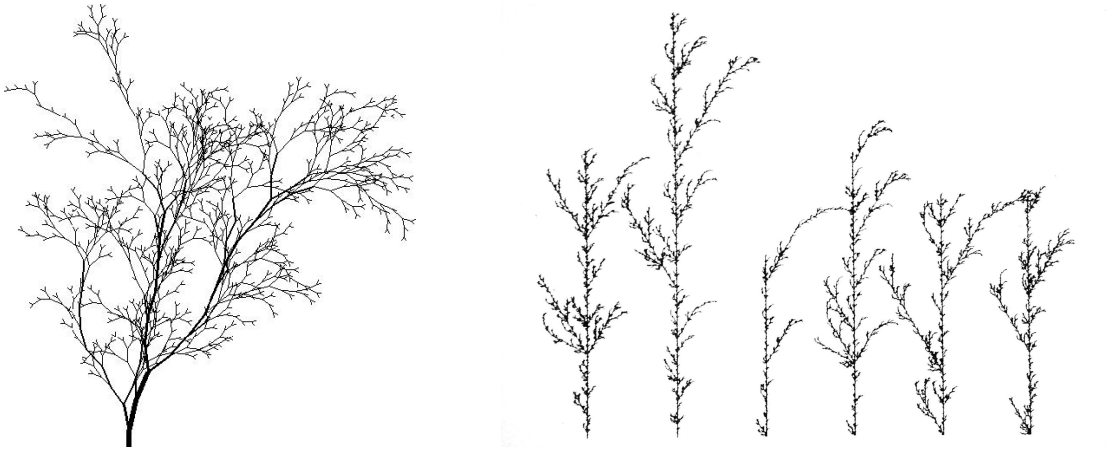
I wrote three main programs for the generation and display of L-Systems. These are a native Python program, an implementation of that program embedded in Blender, and the final L-System Explorer described in detail in this thesis. These projects were interspersed with a research process that involved investigating other implementations of L-Systems and related topics. I'll discuss a few of those related topics here before expanding the discussion to include other visual structures that I feel are worth exploration.



(a) The Python Program (b) The Blender Program

Figure 3.1: Earlier Programs

The 1990 book *The Algorithmic Beauty of Plants*[2] grounded my study from the beginning. This opened my eyes to the beauty of L-Systems. I recommend the free version that is available online, as the book is now sadly out of print. The book is a font of compelling diagrams and images of fairly convincing computer-generated plants. It would be worth a look purely to see the pictures of plants created with L-Systems.



(a) A sample tree (b) Some stochastic branching structures

Figure 3.2: Trees displayed in ABOP (Prusinkiewicz & Lindenmayer (1990))

[2]Prusinkiewicz & Lindenmayer (1990)

Another useful resource for me has been the YouTube channel *The Coding Train*.[3] I'm grateful for the wide and compelling variety of their videos. They were my introduction to some of the ideas that I talk about more in the following Generative Art section, and they helped me learn about Processing. If you have an interest in taking the next step in your computer science journey, I would highly recommend these videos, especially the tutorials on getting started in Processing. I've followed along many of the tutorials and discovered some fantastically interesting concepts that I hadn't come across in school, such as Perlin and Simplex noise, strange attractors, and space-filling curves.

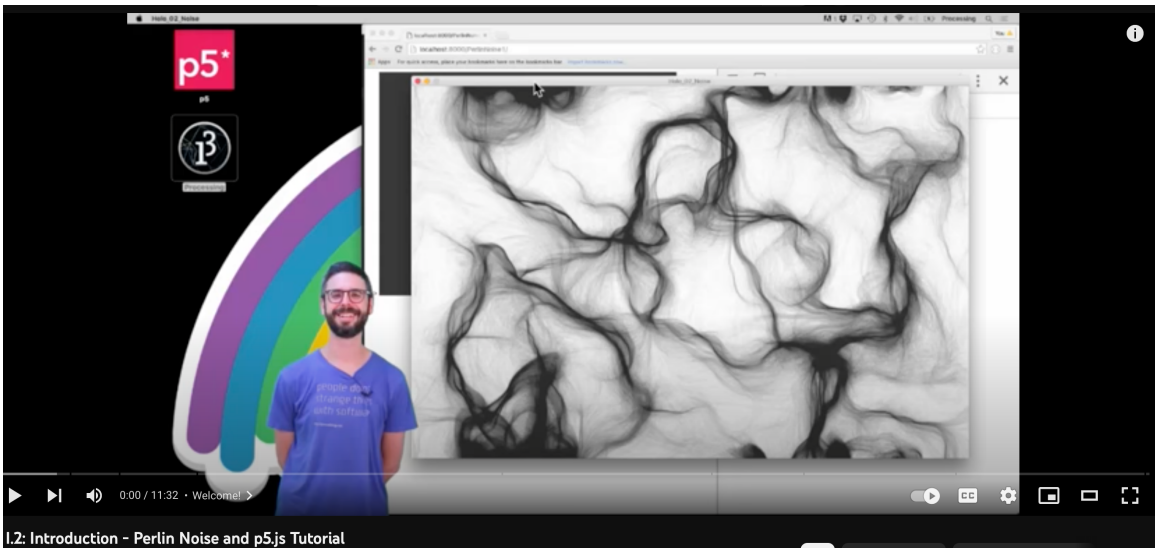


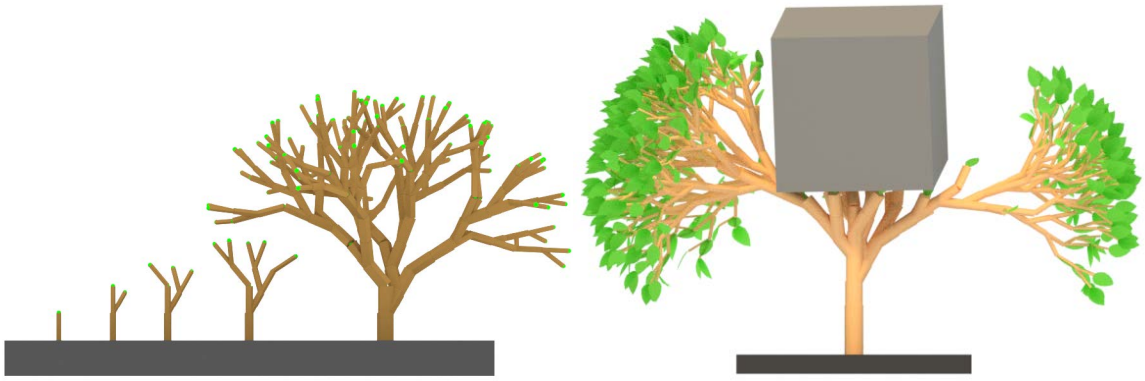
Figure 3.3: A Coding Train video (Shiffman (2023))

Nikole Leopold wrote a Bachelor's thesis about an implementation of L-System graphics in Blender in 2017 at the Technical University in Vienna.[4] This work informed some of the later discussions that I've had as part of this thesis. Leopold gives a comprehensive overview of the advantages of using a 3d modeling software like Blender to display L-Systems. They also introduce an add-on which enables the user to immediately see the results of their changes, a strategy I admire and have worked to make an essential component of my own project. One of the pieces of that research that I found especially interesting was the inclusion of methods by which the L-System can interact with its environment. The structure grows, finds an obstacle, and appears to go around it. (The program actually just cuts off the branches that intersect the object, but the effect is still compelling.) The discussion of Differential L-Systems was also fascinating. These can be animated continuously, by which I mean *between* production steps.[5] I'm grateful for the resources that Leopold included as well, as they pointed me in the direction of other compelling sources.

[3]Shiffman (2023)

[4]Leopold (2017)

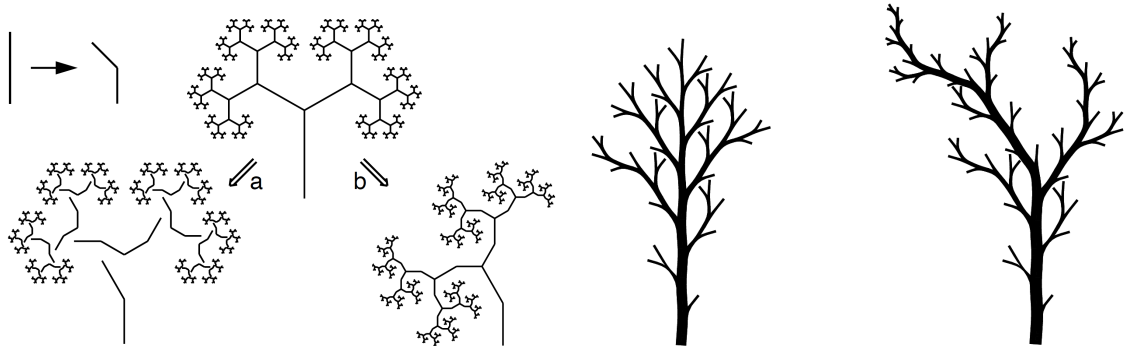
[5]This would mean animating discrete segments of production rules, enabling an even greater animation resolution



(a) Growth steps (b) Leopold's pruning algorithm in action

Figure 3.4: Images from Leopold (2017)

Przemyslaw Prusinkiewicz, who worked with Aristid Lindenmayer on *The Algorithmic Beauty of Plants*, produced a handbook in 2001, titled *L-Systems: from the Theory to Visual Models of Plants*[6] that I view as something of a sequel to the work presented in *ABOP*. It goes into depth on some of the ideas presented in *ABOP* but with new pictures that bring wonder in all the right ways. It discusses different forms of L-Systems, a few of which I discuss in the next chapter. It also explores hypothetical models that I'd never considered, such as simulating an attack on a plant by an insect, and displaying how the system reacts.



(a) The effects of changing angle (b) L-Systems with more growth rules

Figure 3.5: Images from Prusinkiewicz et al. (2001)

Finally, the paper "A Theory of Animation: Cells, L-Systems, and Film" goes well beyond the scope of this thesis in its discussion.[7] It provides some insight that's more helpful for the larger questions that have been raised during this process, and less about the structure of L-Systems. It suggests a view of L-Systems and similar structures as "animations of a *theory* of cellular life." (32) This is especially interesting

[6] Prusinkiewicz et al. (2001)
[7] Kelty & Landecker (2004)

to me in the context of the discussion of the alive-ness of machines, and the degree to which a machine that simulates life can be said to actually *be* life. I ask these questions more clearly a bit further on in section 3.3. This paper is also interesting to me for the degree to which it ties popular culture to these simulators. It is cross-discipline in the most appealing way. Here's a paragraph from the paper that I find to be particularly exciting in how it connects seemingly distant themes in a coherent way.

> To quote the poet Rainer Maria Rilke, who also clearly thought through cell theory, we have sought here to follow the cell through its presence in twentieth century media as the "tangible immaterial means of representation for everything." Scientific and artistic objects, sea urchin embryos and science fiction mountains, are "realities that emerge from handwork," that of animators tinkering with machines and media.[8]

Another class of life simulator that's often grouped with L-Systems is the varied field of Cellular Automata. These structures surpass L-Systems in terms of simplicity, often operating in a basic grid or even just a line of points. Popular examples are John Conway's *Game of Life*[9] and Stephen Wolfram's one-dimensional Automata.[10] These have been used alongside L-Systems to generate life-like effects for film and other media for a long time.[11] Their simplicity also makes them a great tool for creating emergent systems and acting as simple models for complex behavior. They also exhibit a remarkable capability for beauty, in a similar way to L-Systems.

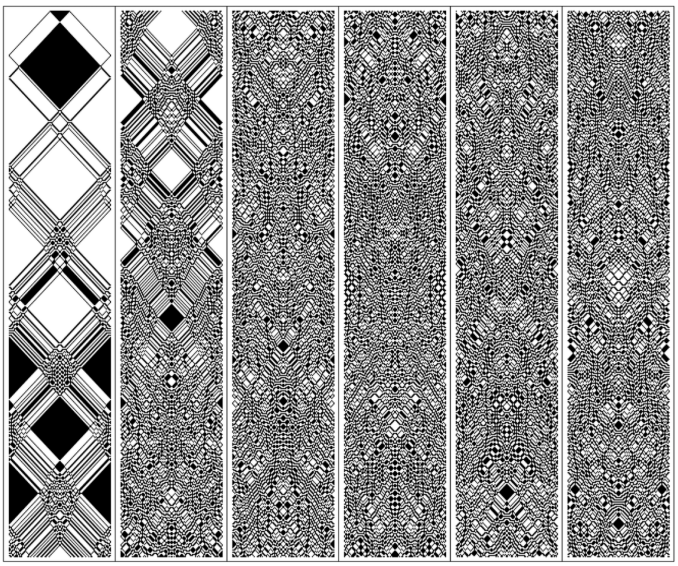


Figure 3.6: An illustration from Wolfram (2002)

[8]Kelty & Landecker (2004)
[9]Games (1970)
[10]Wolfram (2002)
[11]Kelty & Landecker (2004)

3.2 Generative Art

People have been fascinated by the cooperative creations of human and machines for as long as such things have been able to be made. Many of these creations are seen as *beautiful*. These structures don't necessarily have to conform to traditional standards of beauty, but they do look usually look like what we know as *art*. These works just happen to be created in part by machines. One section of the creations made by machine but under some human assistance fall under the label *Generative Art*. For the purposes of this thesis, I will define Generative Art as art that comes from a machine with the guidance of a human.

With only a little searching, I've found several online communities that create beautiful collections of this work.[12] I've also been fortunate to be near many people that are also interested in many of these same ideas and have supported me in this work more actively. I credit Trever Koch for helping me bring many of my early ideas to life. These resources have inspired me to create work that I feel excited about. There are resources available if you are at all interested in learning more about this compelling work. I've collected specific resources and put them in Appendix B.

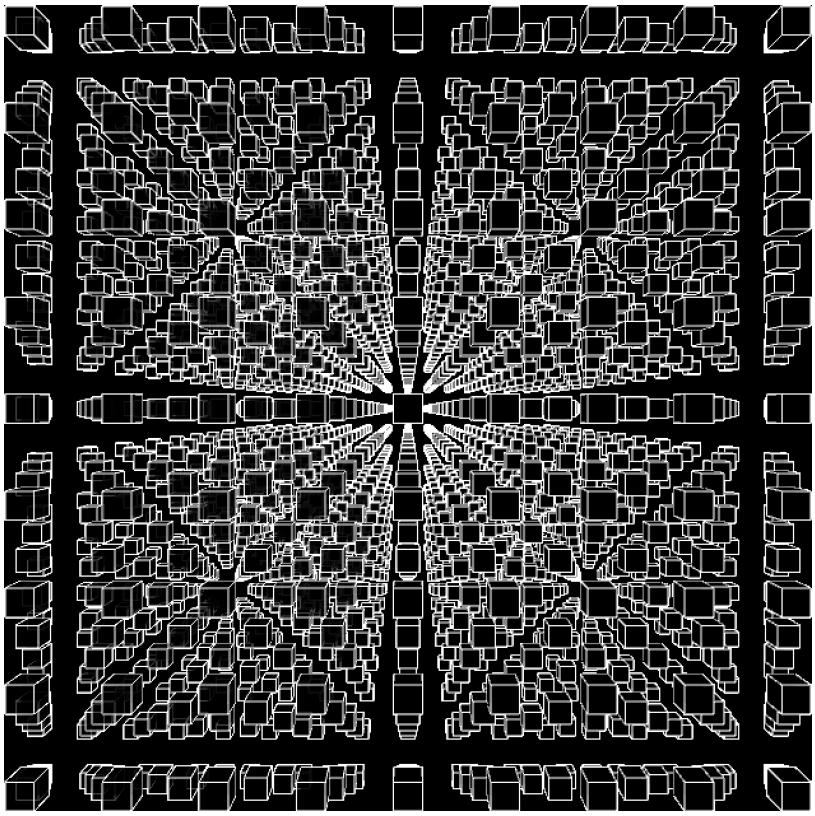


Figure 3.7: A Processing sketch I wrote, exhibiting Moiré patterns

There is a huge variety of work that could be considered Generative Art. The art made by systems like DALL-E, Midjourney, and other AI programs might fall under

[12]Reddit (2023)

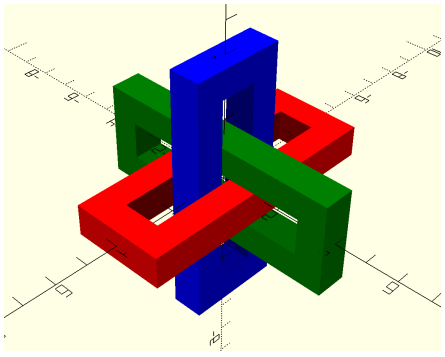
some definitions of Generative Art, but it feels distanced enough from the material that I'm interested in covering here that I'll isolate them to a discussion in section 3.3.1. Figure 3.8 is an example such work.

Figure 3.8: An image from the Midjourney Discord server (Discord & Bot (2023))

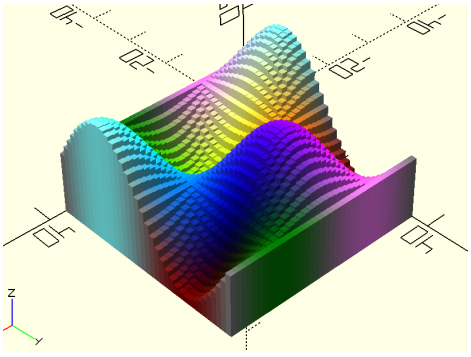
3.2.1 My Explorations

I explored a variety of systems for creating artwork with machine means during the first stages of my research process. I'll introduce some of those structures here and demonstrate how they tie into the theme of simulating life. Each of these projects are deserving of greater research in their own right, and so I won't attempt to make my explanations comprehensive. This section I hope you enjoy these objects as both the artworks and the mathematical structures that they are.

Here are some of the chronologically earliest projects that I explored. The objects shown in Figure 3.9 were made in a program called OpenSCAD (pronounced "open S cad") and were a first attempt to create artworks through software.[13]



(a) An early creation



(b) Playing with color and trigonometry

Figure 3.9: OpenSCAD creations

[13]Kintel (2023)

Figure 3.10 is a "mesh" that I was messing around with during these early stages. I was curious if I would be able to modify it so that I could fold it out of paper. This particular version was unfortunately never realized.

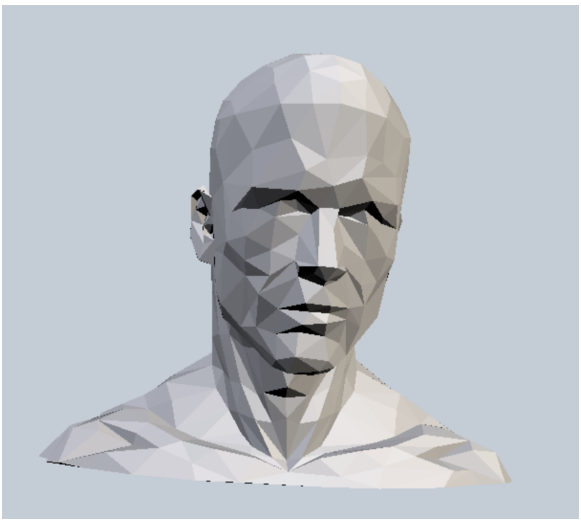
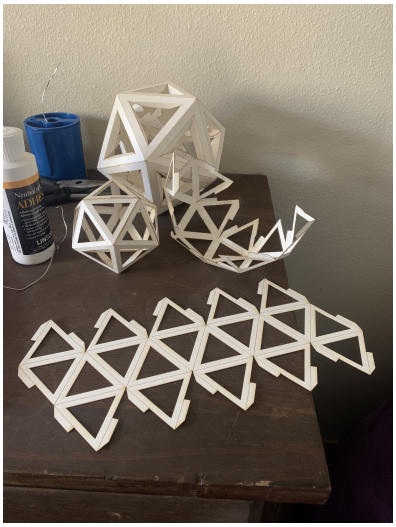


Figure 3.10: A head mesh I hoped to unfold, print onto paper, then refold

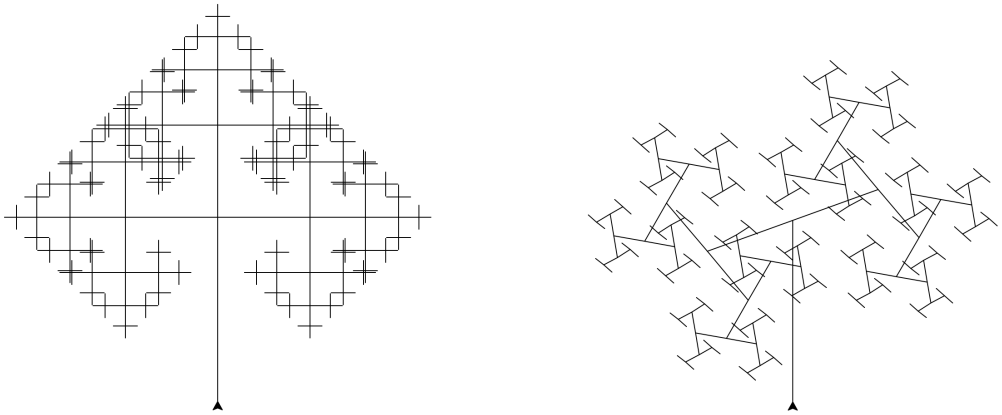
However, I did manage to create a few physical models of geometric solids, shown in Figure 3.11a. These fall into the category of "papercraft", a medium that I feel especially connected with. The L-Systems that I've been showing off take a line and brings it into the 2nd dimension. These papercraft objects bring 2d objects into the third dimension. These paper models are descendants of a series of creations that I've been making years before embarking on this thesis project. Figure 3.11b shows some examples of those.



(a) Iscosohedra laser-cut from cardstock, then folded and glued

(b) Modular Origami geometric solids (these don't use glue!) Also surrounded by other projects.

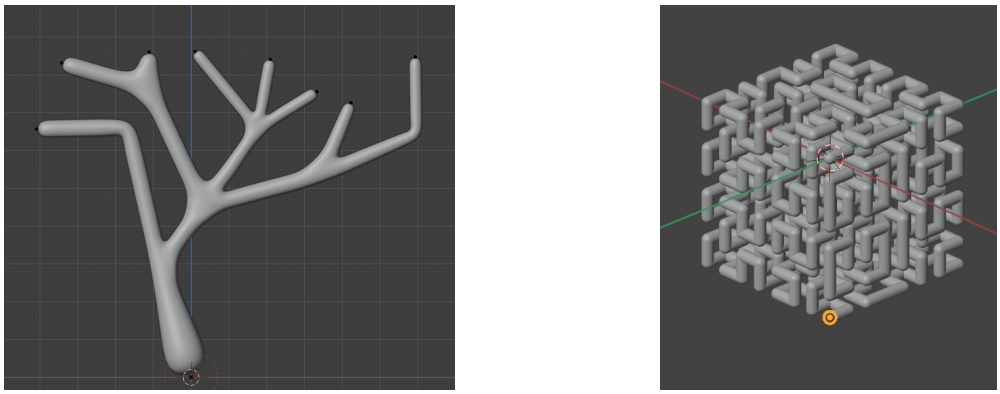
I've already mentioned my early explorations into the Python implementation of Turtle Graphics. Figure 3.12 shows some of the structures constructed during that time.



(a) An early creation (b) Playing with color and trigonometry

Figure 3.12: Early creations with Turtle Graphics

At this point, I began to explore the possibility of bringing L-Systems into the third dimension. I used Blender and an implementation of Turtle Graphics that I constructed to build complex 3d shapes. My favorite of these is probably the 3d Hilbert Curve, in part because it made me consider how the turtle works. In a sense, the turtle doesn't move through its world, instead the world moves around it.[14] This is a part of this process that would be very fun to explore further, especially with the knowledge from a paper that I came across later in the process. Figure 3.13 depicts an early exploration into Blender's capabilities and a 3D Hilbert Curve made using an L-System.

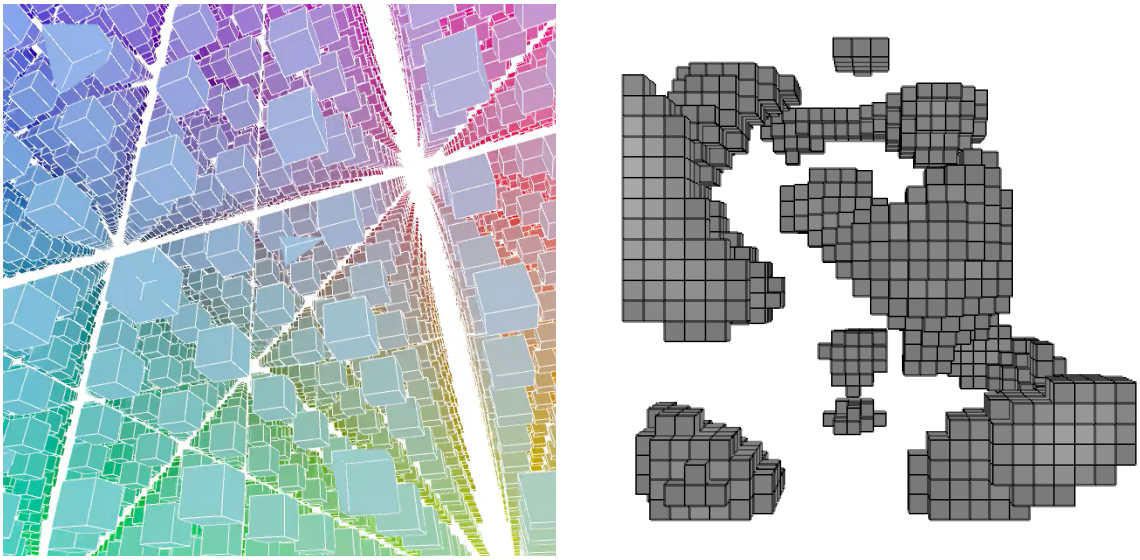


(a) Blender can smoothly attach joints (b) 3D Hilbert Curve

Figure 3.13: Blender Experiments

[14] DeRose (1989)

It was at this point in the thesis that I encountered Processing. Figure 3.14 shows two of the projects that I made when learning how to code in this language. The second image, Figure 3.14b is a snapshot of a program in motion, and much of the detail is lost when the time dimension is removed. I'd like to put some of these on my website at some point to let folks experiment with them.



(a) Processing handling a large array (b) Dynamic cube using Simplex Noise

Figure 3.14: Two of my early Processing programs

These projects were instrumental to the development of the thesis as you see it now. Without acknowledging the steps that led to the final product, the story of this work would be incomplete.

3.3 Quandaries, Puzzlements, and Musings

While working on this thesis and the various projects that came with it, I began to begin to see things through a peculiar lens. I was researching this type of structure that had been used for half a century to describe living things with an almost startling degree of accuracy, considering the simplicity of their construction. I started to question the line between living things and the code we write to imitate those living things. Is there actually a difference? I'm not so sure. I'll share where my thinking has been going these last few months over the course of this section.

This is the sort of question that people have been asking for hundreds, if not thousands of years. Machine-makers have attempted to imitate life by different means for a similarly long period.[15] Science Fiction authors and computing pioneers like Alan Turing have suggested ways in which we might learn if there is a difference between life and machines.[16] Some of those questions have borne fruit or guided our

[15] Riskin (2016)
[16] Turing (1936)

development, others have marked moments when we moved the goalposts to imagine ever-grander goals.[17]

In the last decades, there have been tremendous advances in the simulation of life. Robots have beaten skilled human Chess, Go, and Starcraft II players, and have become indistinguishable from humans in many respects.[18] We've become used to robot assistants in our homes and in our daily lives. The rise of AI-powered chatbots seems poised to throw a wrench into the gears of several industries. There seems to be no end in sight.

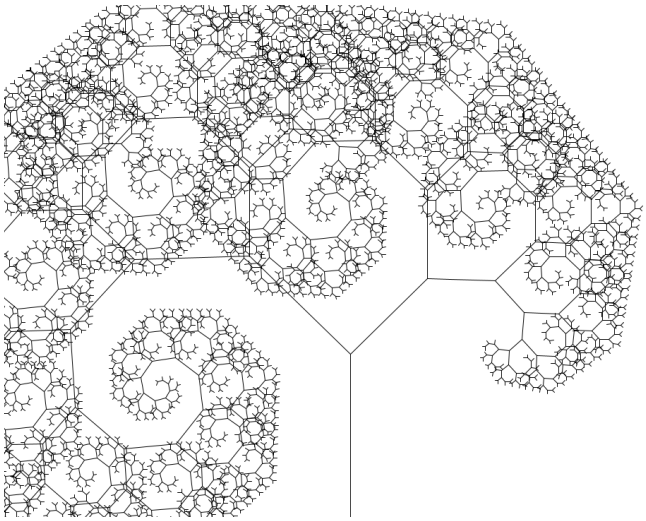


Figure 3.15: Finding Chaos in Order

In contrast to these massively complex systems, L-Systems are extremely simple, and I will not make the argument that they are themselves alive,[19] but I suggest that they support a specific understanding of the world. This understanding suggests that everything consists of purely physical operations and no invisible, imperceptible, separate element, like a spirit or a soul. L-Systems demonstrate that an intense variety of complexity can easily arise from almost nothing. I don't see any reason why people shouldn't be subject to the same rules.

3.3.1 The Times We Live In

At the time of writing, ChatGPT has taken the world by storm.[20] While it doesn't appear to be destroying society yet, there's no saying what its successors will do. I believe that if we are going to be engaging with AI in any sort of meaningful way in the near future, we need to understand what that really means. How do we engage with entities that can do almost anything we can do, but often faster and better?

[17]Kelty & Landecker (2004)
[18]Garisto (2019)
[19]Kelty & Landecker (2004)
[20]OpenAI (2023)

We can try to hold onto the reins of these new creations, but so many pieces of sci-fi media emphasize why doing such a thing will only lead to cruelty, confusion, and won't even actually work. It seems right to me to suggest that people must be willing to accept that at least *some* of that which an AI makes must be called art. This art might come from a human prompt and a huge human-generated dataset, but things like collages might be comparable.[21]

How does this relate to this thesis? Our definition of art has changed drastically over the years, and it might be a useless endeavor to even try to hold onto the conceit of the word in the face of a dramatically changing social dynamic with visual (and otherwise) media. The little simple shapes that can be built in the Explorer are just that, little simple shapes, but they carry with them a wealth of implication and meaning that goes far beyond their appearance. Who knows if our definitions of art will really change with the coming of AI? I'm willing to bet they will.

3.3.2 Is Math Art?

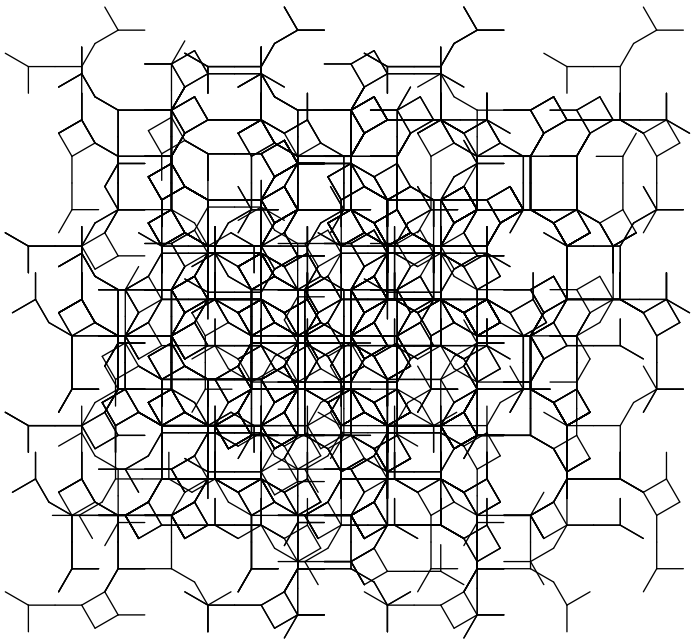


Figure 3.16: Tile Patterns[22]

It feels right to ask this question here as well. Where is the line between work that is created by a person as art, and similar work that is simply drawn from structures inherent in math? Would a person be able to fairly call a math structure art? What if they were responsible for choosing this particular iteration? Maybe they picked the color that it's displayed with? Would it still be art if they found the barest way to

[21] Harris (2023)

[22] A; A $\rightarrow$ +F[+F+B]–FA, B $\rightarrow$ F[+FA]–F–B; θ = -90°, ϕ = 60°; Iterations = 11

display the simplest of equations? When can we say it isn't art? Can all math be art?

Where does the simulation of life fall into this? Physicists might have you believe that everything can be governed by some set of equations and everything that exists is as it is because of the results of those equations. Would it then be believable to call some of the fundamental physical structures artistic? Does art need an artist? Does this then preclude non-living things from producing art? Perhaps this suggests that AI would never be able to create art so long as it remains non-living.

I would be so bold to suggest that the products of the L-System Explorer *are* art. I couldn't say for sure why I feel the way I do, although I suspect part of it relates to the user's involvement in their creation, and that they wouldn't have existed without the intervention of a person.

However, these structures are clearly just numbers and letters, all of which are directly displayed on the screen for anyone to read. A random number generator could have easily created another L-System which looks similar and probably looks pretty cool. Would that new L-System be art? Would it even be artistic? They are all just math, so what would actually make one L-System art and another not, beyond the method of their creation?

I don't know!

Chapter 4

Past, Present, Future

L-Systems were introduced in 1968 in a paper by Aristid Lindenmayer, where he suggested how one could construct a mathematical model of the structure of simple cells.[1] Even in this introductory paper, L-Systems were deeply tied to the modeling of life. As could be expected, the early models generated by the first L-Systems were extremely simple. They laid the foundation for what was to come, but did not investigate the larger world of possibilities that these structures could make real. In the last 55 years, there have been many developments in this area, most of which have been deeply attached to the development of fast computer graphics. By bringing these systems onto a screen where users could easily engage with them, there was increased interest in making them more accurately reflect the growth of actual plants. He discussed the inseparability of theories from languages, which others have interpreted to mean "Theories become like the biology they purport to describe: natural, evolving forms with dynamics and feature all their own." (46)[2] In other words, Lindenmayer was in part concerned with the creation of axioms of biological knowledge.

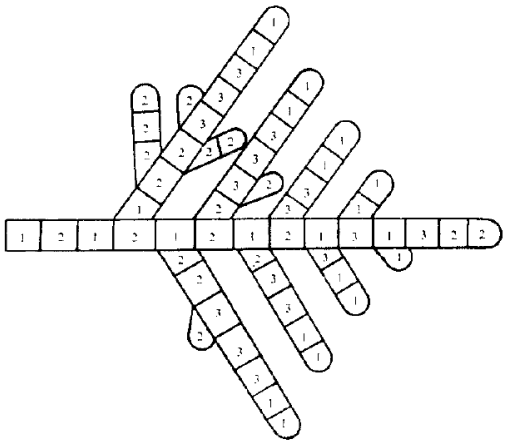


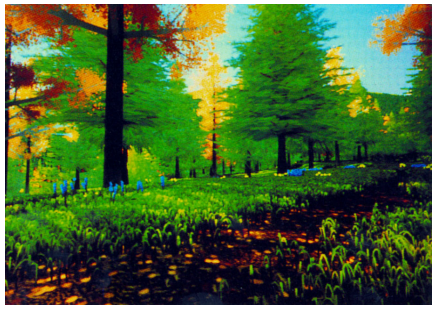
Figure 4.1: One of the first diagrams of an L-System, from Lindenmayer (1967)

Since their invention in 1968, L-Systems have found places in many notable cul-

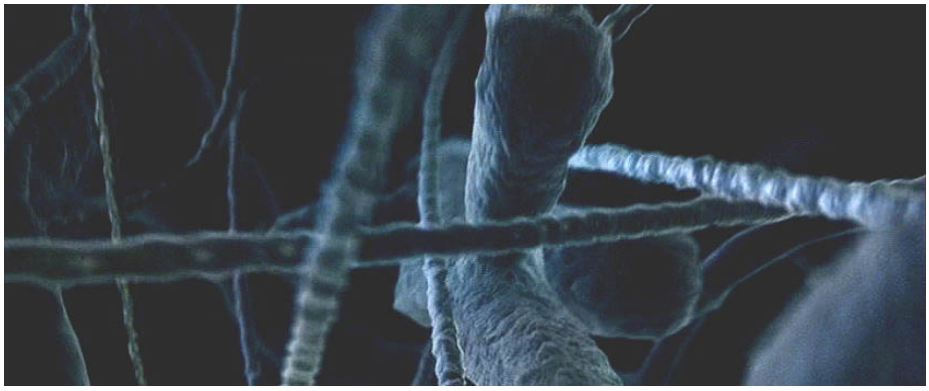
[1] Lindenmayer (1967)
[2] Kelty & Landecker (2004)

tural touchstones. They were used in the creation of the forests in Shrek,[3] the opening sequence in Fight Club,[4] and trees and other structures in many Pixar films.[5] They were the focus of around 5000 articles published from 1968 - 1996,[6] and have been a part of many more since then.[7] L-Systems are extremely useful for creating complex structures with minimal overhead, a feature which lent itself particularly well to simulations used in films. In an extension of Lindenmayer's original goal of creating a formal structure for describing biological systems, L-Systems found use in many new and varied applications. The "productivity of the mathematical formalism would eventually overwhelm its usefulness as a biological theory".[8]

(a) The Shrek forest, Adamson & Jenson (2001)



(b) A Pixar forest, Reeves & Blau (1985) and Prusinkiewicz & Lindenmayer (1990)



(c) Opening of Fight Club, Fincher (1999)

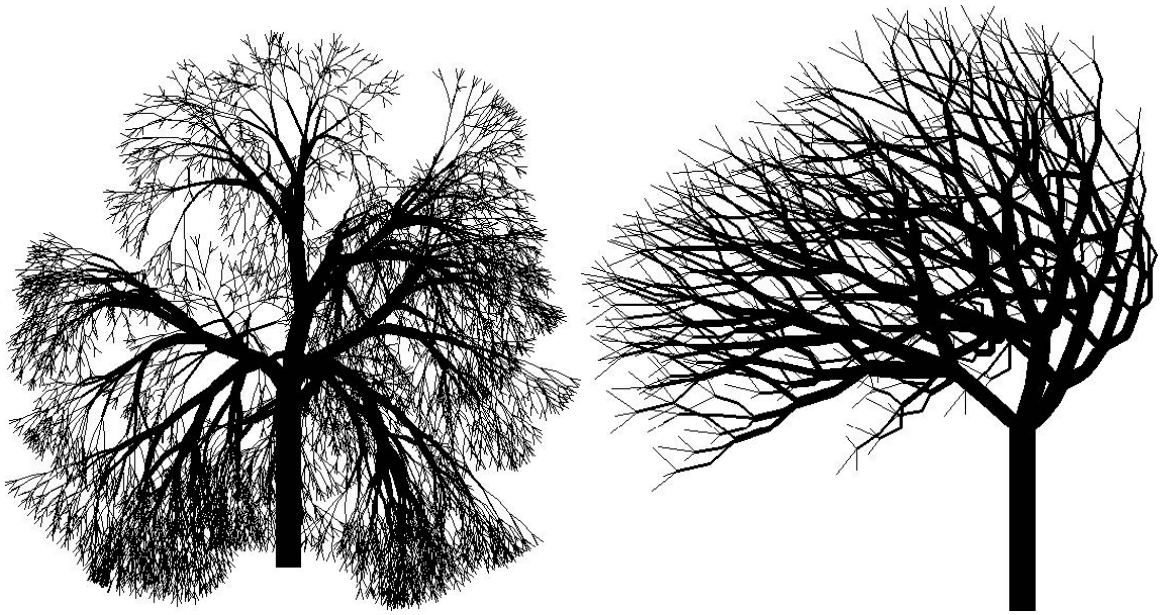
Figure 4.2: Pop Culture appearances of L-Systems

[3] Adamson & Jenson (2001)
[4] Fincher (1999)
[5] Kelty & Landecker (2004)
[6] Kelty & Landecker (2004)
[7] Academia.edu (2023)
[8] Kelty & Landecker (2004)

4.1 The work of Przemyslaw Prusinkiewicz and Aristid Lindenmayer

The Algorithmic Beauty of Plants was an important source in the development of this thesis. It has guided my thinking and helped point me in directions that I've since explored. It develops a narrative around increasingly complex descriptions and implementations of L-Systems. It starts with much of the material I cover in this thesis and dives deeper into the foundations of this work.

There are some compelling parts of the book that I mention in section 4.3, specifically regarding extensions to L-Systems that dramatically increase their capabilities and functionality. It contains an abundance of thoughtfully demonstrated resources and examples, as well as, of course, many compelling pictures. If this work has been interesting to you so far, I strongly suggest taking a look at *ABOP* and deepening your understanding.



(a) Heavy branches (b) Blowing in the Wind

Figure 4.3: Tree-like structures from Prusinkiewicz & Lindenmayer (1990)

4.2 Extending the Program

The L-System Explorer could be extended in many ways. An early addition would be to finish porting the processing version of the code to p5.js, so that it can be hosted on a website and accessed easily without needing to download and install anything. If I can successfully port the Explorer, I'd like to optimize the site for mobile use.

Beyond the porting considerations, I'd like to make a number of improvements to the program itself. I'd like to the ability to change the color, length, and waviness of

the lines it draws. It would be good to add some extra precision in the angles to make the transition between different angles smoother. It would also be helpful to allow the user to move around the main screen and zoom in and out. Another addition would be safeguards to make sure that the program doesn't go out of control if someone tries to simulate an L-System that the program can't handle.[9] These safeguards could come in the form of bounds on the number of generations or possibly total number of line segments drawn.

I'd also like to make it possible for users to type in the production rules for an L-System directly and bypass the interactive editing process. While I see the drawing through node selection as the main attraction of my program, I understand that it would be extremely useful to be able to copy and paste productions directly into the Explorer. It would also be nice to include a library of cool L-Systems that every user could access. On that note, it would be good to make some sort of tutorial about how to work the program within the program itself. This last point is especially important. My goal is for the Explorer to be intuitive enough to understand purely from individual exploration, however it doesn't seem to be there yet. I also don't want to force users to read this thesis before engaging with the Explorer.

One user interface problem I would like to solve is the awkward overlap of control nodes. When one node is selected immediately after another node, it's often impossible to select the earlier of those nodes. This is a problem for mostly rotation and production nodes. It happens because each of the nodes are the same size and my code will only select the most recently created node if the mouse is over multiple nodes. I might resolve this by making the earlier node larger than the more recent node, but then it might make it difficult to select the most recently placed node. This solution would spiral into a dramatically worse situation the more production nodes are put directly on top of one another. With 5 such nodes, it might be nearly impossible to access the middle one. Another solution to this problem could come from making a list of nodes pop up whenever the mouse is hovering over their stack. This would look very much like the menu for adding a node, but perhaps vertically so the two systems could co-exist.

A set of further extensions fall into the category of significantly extending the functionality of the Explorer. These extensions are fascinating and powerful, but generally beyond the scope of this thesis. I'll explore some of these possibilities in the next section.

4.3 Generalizing L-Systems

So far, we've only been looking at L-Systems that fall into a very narrow category. Prusinkiewicz and Lindenmayer go into detail about a number of variations on L-Systems that increase not only their complexity but also their capabilities. I've

[9] When rendering a certain amount of lines, the Explorer slows dramatically and I'm unable to change anything. I'm not sure how many lines it takes to do this, but a good way to avoid having to restart is to bring the number of iterations down to a low value before adding new parts to any of the production rules.

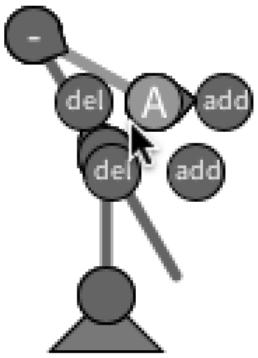


Figure 4.4: Unintended behavior (the `A` node is under the rotation node. It should not be able to be selected right now. Multiple nodes are also activated at the same time, which should be impossible.)

avoided these extensions for two reasons, one reason being time and the other being a desire to keep the complexity of the L-Systems Explorer low in order to keep it approachable. I'll focus on three main variations that were discussed in length in *ABOP*, and I invite the reader to explore them if you feel so led.

The first of these is *context-sensitive L-Systems*. In the L-Systems that I've shown up until this point, each production variable does not depend on the surrounding values when determining what its production will be. If we choose to explore what L-Systems can look like when they are context-sensitive, we can see straightaway that there will be more possibilities than with the context-free L-Systems we've seen up to this point.[10] This type of L-System can build structures that look more like something you might expect from a cellular automata, like John Conway's Game of Life.

Stochastic L-Systems are a type of L-System that introduce randomness as part of the construction process, resulting in different structures each time a particular L-System is drawn. The randomness might be used to vary the angle of rotation by some amount or vary the length of the drawn lines. It might even play a role in the placement of productions in the L-string. These L-Systems contradict a fundamental aspect that are part of what makes L-Systems so compelling to me. That aspect being the rise of apparent complexity out of a set of simple rules, and specifically complexity that is deterministic. However I admit that there is something interesting about how this L-System variation might demonstrate the effects of mutation in an organism. These are often the L-Systems that look most natural, as a random factor can sometimes imitate the near infinite complex factors that affect plant structures out in the world. Some examples are shown in Figure 4.5.

[10] Any basic context-free L-System can be made by a context-sensitive L-System where none of the productions actually depend on context.

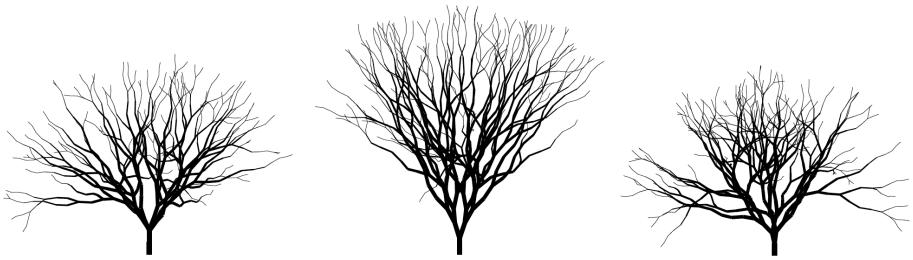
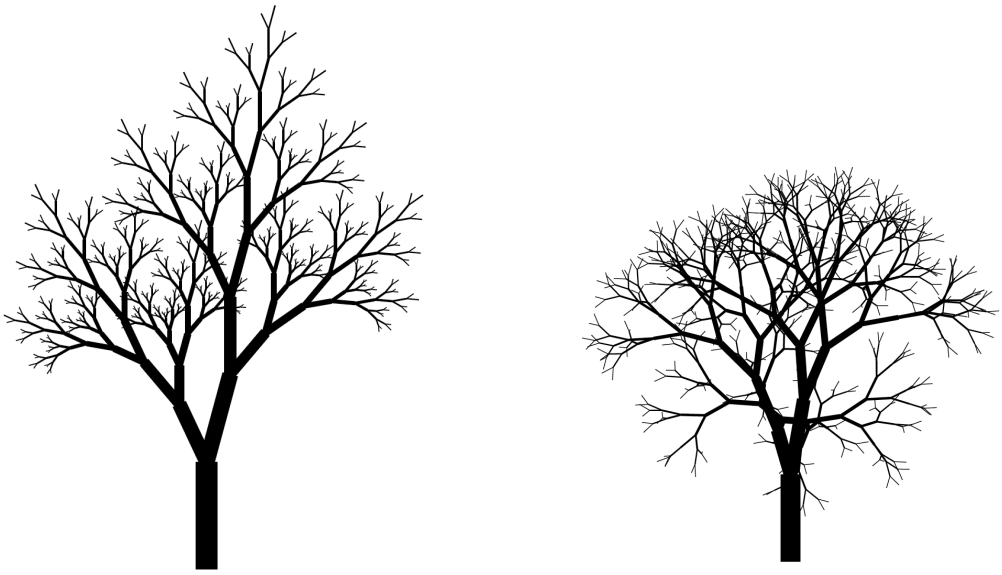


Figure 4.5: Stochastic L-Systems from Prusinkiewicz et al. (2001)

Parametric L-Systems are the most appealing and powerful L-System variation to me. You can see two examples in Figure 4.6. They permit the different operations that the L-System do to no longer necessarily be discrete. It does this by allowing the parameters to be defined as continuous values defined by mathematical equations. They also can include logical operators like "and" and "or" as parts of conditions that must be satisfied for some part of the L-System to be created. This freedom of expression allows the width and length of the lines drawn to be dynamically changed *partway through* the drawing of the L-System. A parametric L-System allows for some fantastically organic-looking images to be generated.



(a) Upwards Parametric (b) Sideways Parametric

Figure 4.6: Parametric L-Systems from Prusinkiewicz et al. (2001)

Limited-Propagation L-Systems can be classified as those systems which encode the removal of parts of productions into the productions themselves. These are actually implicitly implemented in the L-System Explorer, as when a production is referenced and its production rule doesn't contain any other symbols, then that production is just erased. There are a few types of these L-Systems. Non-propagating L-Systems use productions which *erase* themselves. The production rules for these

productions look like `A` $\rightarrow \varepsilon$. This character, ε, symbolizes *nothing*. When this production is processed, it produces nothing. One variation of this type of L-System is shown in Figure 4.7.

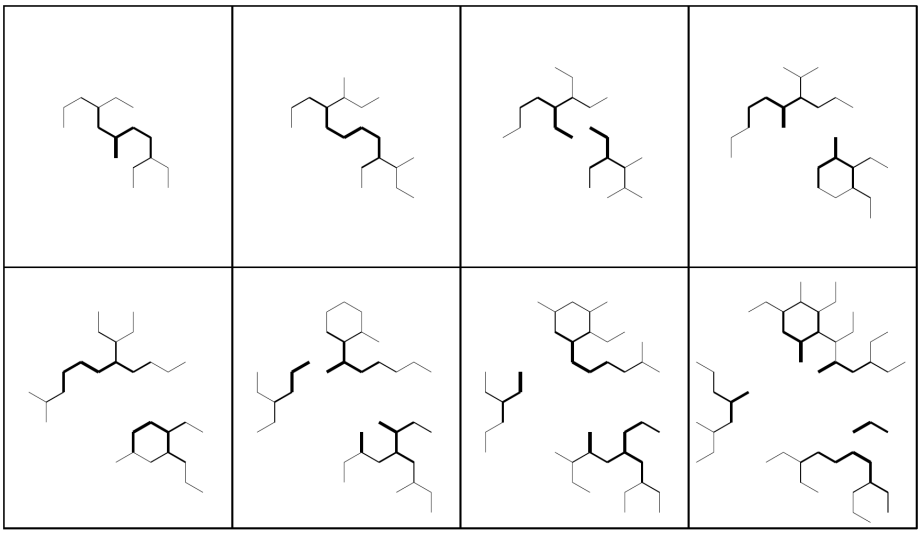


Figure 4.7: Limited Propagation L-System from Prusinkiewicz et al. (2001). Segments are removed 7 iteration steps after their creation.

These extensions of the basic L-System idea are covered more extensively in Prusinkiewicz's 2001 paper.[11] Some variations, such as environmentally-sensitive L-Systems, deserve a thesis to themselves.[12] While a larger discussion of the capabilities of all forms of L-Systems would be exhilarating, it definitely falls outside of the bounds of this undergraduate thesis. I can imagine a program like the Explorer that has implementations of all of these variations and which can produce structures with orders of magnitude more complexity. Given the time and the resources, it would be a joy to explore this avenue.

4.3.1 Interpretations

Up to this point, I've described the content of an L-System as only the combination of some productions and their rules to form an L-string. We can go beyond the overlapping categories of trees, plants, and black and white branching structures.

The most easily removed limitation is that our structures must be black and white. I've provided some examples of work by others that use color as a mechanism to display information imbued in the L-System, such as Figure 3.4b, where the leaves are the end nodes of the tree and have been given a different color. This is still operating under the tree metaphor. Remember the opening sequence of Fight Club.

[11] Prusinkiewicz et al. (2001)

[12] This type of L-System engages with the world *outside of the L-string*, in contrast to context-sensitive L-Systems. If a tree in a video game were generated with an environmentally-sensitive L-System, it could be able to grow around its virtual world and only grow where the sun shines, for example.

It was an example of L-Systems at work.[13] This sequence is a computer-animated fly-through of something that seems remarkably like *neurons*. I recommend watching that video so you have a clear example of what I'm referring to.[14] These structures don't seem to be much like trees at all, and yet they were created by the same recursive replacement structures that made all the trees in this thesis. At the level of an L-string, I suspect that those neurons would be indistinguishable from all the trees I've shown you. That's because they are the *same structure*.

Artists like Michael Hansmeyer have taken L-Systems away from their biological roots and have used them as a tool for expression. His show, *L-Systems in Architecture*,[15] transformed the traditionally organic shapes of L-Systems into a mass of metal and glass that seem definitionally opposed to their origin. Figure 4.8 shows a snapshot of his show.

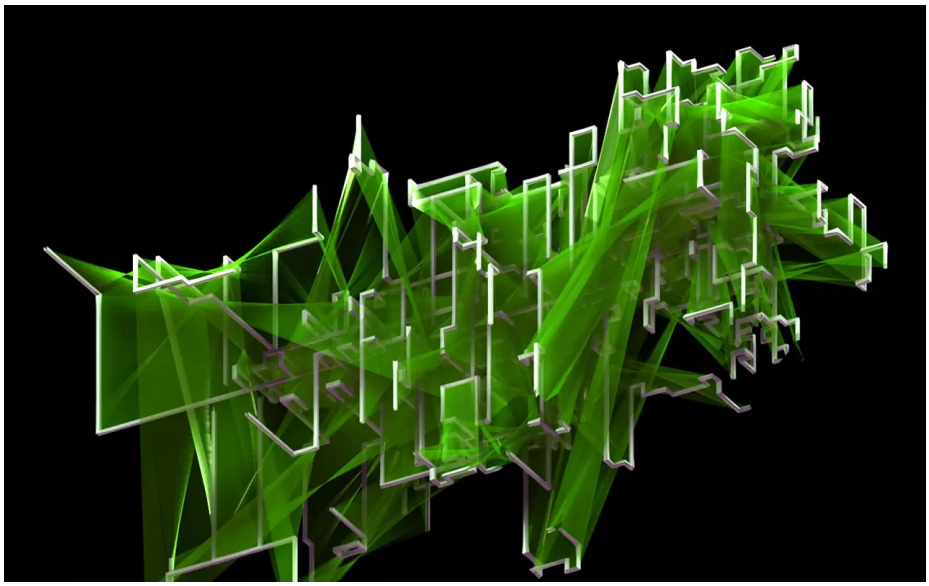


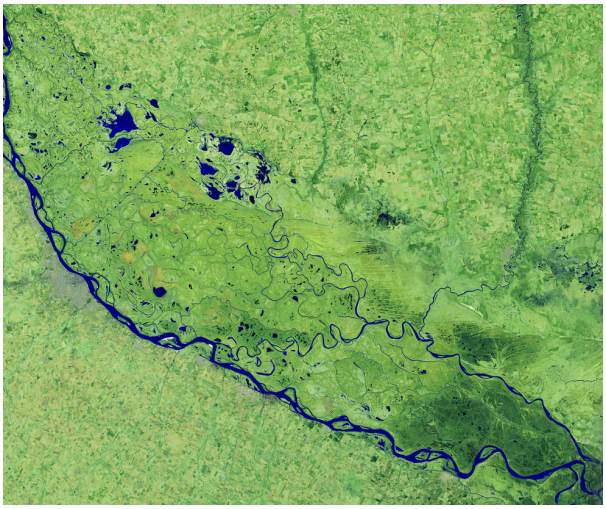
Figure 4.8: An alternative interpretation of L-Systems, from Hansmeyer (2003)

The neurons of the Fight Club opening are one example of an alternative *interpretation* of the L-string. L-Systems can be used to simulate mountains, algae, fungi, bacteria, lightning, blood vessels, and even galaxies if given the right parameters and a sufficiently convincing interpretation. I can even imagine the possibility of L-Systems being able to generate music, with the right rules and rendering system.

My intention with this last section is to encourage you to see that L-Systems are capable of more than what they've been used for up until now. While they've been extensively studied mathematically, that doesn't mean that they're exhausted of compelling material. I feel that we are in an especially strong position regarding L-Systems right now, seeing as so much of their power has already been documented. We can build off of the solid foundations that were first established in 1968 by Aristid

[13] Fincher (1999)
[14] CHmjoo (2012)
[15] Hansmeyer (2003)

(a) A Snowflake, Bentley (1865-1931)



(b) The Paraná River, Dauphin (2020)

(c) A mangrove forest in Loxahatchee, Florida, US Department of Commerce (2023)

Figure 4.9: Structures that might be imitated with L-Systems

Lindenmayer in his search to describe natural structures though mathematical means. We have discovered the potential of Artificial Intelligence to fly past so much of the grunt work that has been holding us back. We can create art hand-in-hand with these machines that we've built, while not forgetting the simple systems that helped get us to this point.

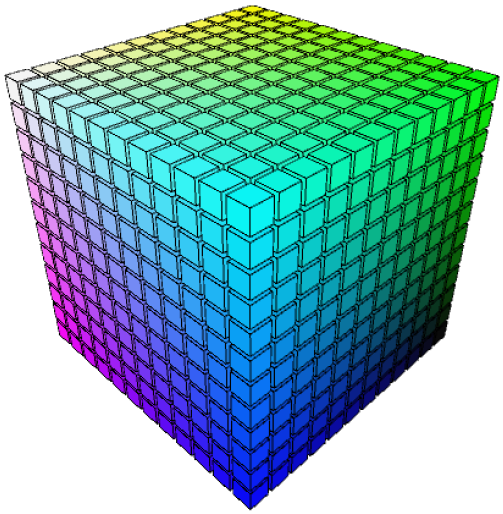
Conclusion

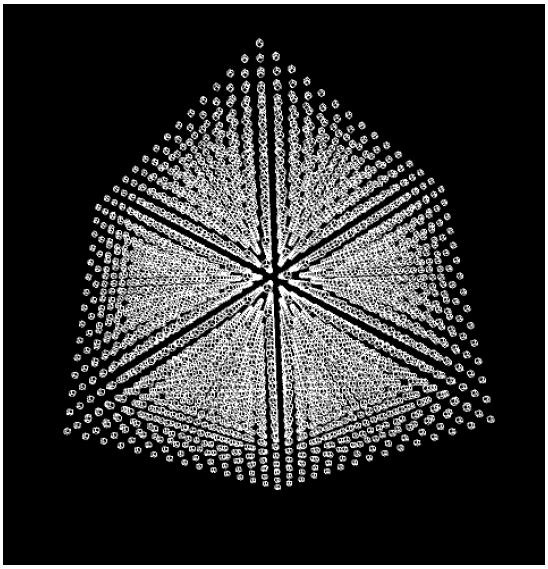
L-Systems can set free the imagination. They simulate some element of what it means to be alive while still remaining an extremely simple structure. They have been a powerful object to demonstrate the complexity of life but also to show that much of what we might imagine to be beyond description is in fact capable of being described, and described succinctly. I have introduced the L-System Explorer, a tool that brings L-Systems into the hands of anyone who is remotely interested. This tool brings full circle the promise of L-Systems as simple complexity; their possibilities can finally be taught with this Explorer in simple terms.

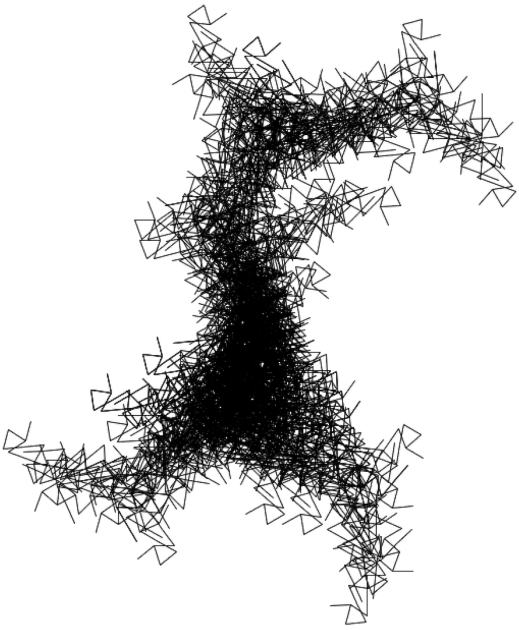
I'm deeply thankful for both the research and the opportunity to do the research that have brought me to this point. I hope that I've been able to convince you of the value of this subject as well as the importance of the methods with which I've discussed it. As every resource that I've come across has acknowledged, it wouldn't be right to try to understand L-Systems in any way other than an interdisciplinary way. I encourage my readers to see that this awareness goes beyond these systems and into more everyday realms. I hope you've enjoyed this work. Take care.

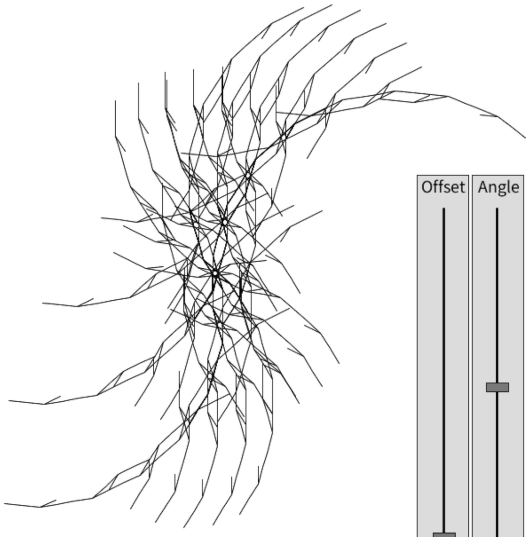
Appendix A

Early Processing Experiments









Appendix B

Generative Art Resources

The subreddit, r/generative has been a treasure trove of intriguing projects. There is a constant stream of new content that continues to push the envelope of what I think could be possible. I especially appreciate the genre of generative art where it feels like the media is coming out of the page. You can sort by "Top All Time" to find some of the best.

Beyond the Coding Train, which I've already mentioned, there are a few YouTube channels that have some especially solid pearls of wisdom to share in areas related to this thesis. Of course, I'd like to mention `3Blue1Brown` and `Numberphile` first, as their videos on nearly every topic are engaging, thoughtfully constructed, and often some combination of witty and beautiful. `Stuff Made Here` and `Mark Rober` have been inspirations in regards to the coding aspects of this work especially, though I would be remiss to not mention `Vihart` as one of my first major introductions to the beautiful relationship between art and math. There are many others who have helped this happen, and the communities that this thesis owes some credit to are constantly evolving.

There are some other digital artists that I recommend you engage with. Many of these are Instagram artists that often post interesting visual graphics, sometimes even L-Systems. Another artist that works heavily in this area between art and math is `Luke Shannon`. His work can be found online at `https://www.lukeshannon.xyz/`. He works with the line between intentional design and random chance in the creation of many of his works. Much of the art he makes blurs the boundary between the maker and the machine, asking us to consider how much control we actually have over the process of creation when machines are involved. It's wonderfully invigorating stuff.

References

Academia.edu (2023). (99+) Academia.edu | Search | L-Systems. https://www.academia.edu/search?q=L-Systems

Adamson, A., & Jenson, V. (2001). Shrek.

Ben-Ari, M., & Mondada, F. (2018). Finite State Machines. *Elements of Robotics*, (pp. 55–61).

Bentley, W. (1865-1931). Snowflake. https://snowflakebentley.com/images

Blender (2018). *Blender - a 3D modelling and rendering package*. Stichting Blender Foundation, Amsterdam: Blender Foundation. http://www.blender.org

CHmjoo (2012). Fight Club Opening Sequence. https://www.youtube.com/watch?v=Ze9-wg9k7AU

Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, (p. 151–158). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/800157.805047

Dauphin, L. (2020). Paraná River. https://www.nasa.gov/feature/amazing-earth-satellite-images-from-2020/

DeRose, T. D. (1989). A Coordinate-Free Approach to Geometric Programming. In W. Straßer, & H.-P. Seidel (Eds.), *Theory and Practice of Geometric Modeling*, (pp. 291–305). Berlin, Heidelberg: Springer Berlin Heidelberg.

Discord, S., & Bot, M. (2023). Midjourney Math Tree.

Fincher, D. (1999). Fight Club.

Games, M. (1970). The fantastic combinations of John Conway's new solitaire game "life" by Martin Gardner. *Scientific American*, *223*, 120–123.

Garisto, D. (2019). Google AI beats top human players at strategy game StarCraft II. *Nature*. Bandiera_abtest: a Cg_type: News Publisher: Nature Publishing Group. https://www.nature.com/articles/d41586-019-03298-6

Goldwasser, S., Micali, S., & Rackoff, C. (1989). The Knowledge Complexity of Interactive Proof Systems. *Society for Industrial and Applied Mathematics*.

Hansmeyer, M. (2003). L-Systems in Architecture. `http://www.michael-hansmeyer.com/l-systems`

Harris, E. A. (2023). Peering Into the Future of Novels, With Trained Machines Ready. *The New York Times*. `https://www.nytimes.com/2023/04/20/books/ai-novels-stephen-marche.html`

Kelty, C., & Landecker, H. (2004). A Theory of Animation: Cells, L-Systems, and Film. *Grey Room*.

Kintel, M. (2023). OpenSCAD. `https://openscad.org`

Leopold, N. (2017). *Algorithmische Botanik durch Lindenmayer Systeme in Blender*. Graduate thesis, Technischen Universität Wien.

Lindenmayer, A. (1967). Mathematical Models for Cellular Interactions in Development II. Simple and Branching Filaments with Two-sided Inputs. *Journal of Theoretical Biology*.

OpenAI (2023). ChatGPT. `https://chat.openai.com`

Papert, S. (1980). *Mindstorms*. Basic Books Inc.

Prusinkiewicz, P., Hanan, J., Hammel, M., & Mech, R. (2001). *L-systems: from the Theory to Visual Models of Plants*. Springer-Verlag.

Prusinkiewicz, P., & Lindenmayer, A. (1990). *The Algorithmic Beauty of Plants*. The Virtual laboratory. New York: Springer-Verlag.

Reas, C., & Fry, B. (2014). *Processing: a programming handbook for visual designers and artists*. Cambridge, Massachusetts: The MIT Press, second edition ed.

Reddit (2023). r/generative. `https://www.reddit.com/r/generative/wiki/index/`

Reeves, W. T., & Blau, R. (1985). Approximate and Probabilistic Algorithms for Shading and Rendering Structured Particle Systems. *SIGGRAPH 1985*, (pp. 313–322).

Riskin, J. (2016). *The Restless Clock: A history of the centuries-long argument over what makes living things tick*. Chicago: The University of Chicago Press.

Shiffman, D. (2023). The Coding Train - YouTube. `https://www.youtube.com/channel/UCvjgXvBlbQiydffZU7m1_aw`

Turing, A. M. (1936). On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, *2*(42), 230–265. `http://www.cs.helsinki.fi/u/gionis/cc05/OnComputableNumbers.pdf`

Turing, A. M. (1950). Computing Machinery and Intelligence. *Mind*, *59*(236), 433–460. Publisher: [Oxford University Press, Mind Association]. http://www.jstor.org/stable/2251299

US Department of Commerce, N. O. a. A. A. (2023). What is a mangrove forest? https://oceanservice.noaa.gov/facts/mangroves.html

Wolfram, S. (2002). *A New Kind of Science*. Wolfram Media. https://www.wolframscience.com