An 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

Thanks to my roommates, Henry Wilson, Brent Ellis, and Gabe Fish for many hours spent talking about the questions that have helped me to create this document in the way that I have.

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.

The cell has been the "tangible immaterial means of representation for everything. [...] [these things are] realities that emerge from handwork," - Rainer Maria Rilke, - Kelty and Landecker (58)

Table of Contents

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 be as interested in it as I have been. 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 I wrote in detail and demonstrate some of its capabilities, as well as the steps I took to get where I ended up. I will also discuss some methods that others have used to do similar things, and go on to explain possible extensions to this work. Wrapping things up, I take a step back from my work to ask some larger questions.

Dedication

To the unending search for understanding.

worlds in worlds and of worlds, by worlds and for worlds

Chapter 1

Introduction:

[Illustration of a simple L-System, perhaps just a fractal tree]

The world of Computer Science can seem like a wilderness, teeming with frightening phrases like Zero-knowledge proofs, NP-Completeness, Finite State Machines, and the Halting Problem, 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 will show and give to you will demonstrate the accessibility of one of this most powerful, subtle, yet familiar disciplines. 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 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 will be accessible on my github, *github.com/RJacobH/Thesis* as well as hopefully 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 on every page I could, and I'd like to imagine that they tell enough of a story by themselves that the words are ultimately unnecessary.

[An L-System that's mirrored across a horizontal line]

The fundamental 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[citation pointing to the section where I discuss the history in more depth], 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*, 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 understand and revel in 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.

L-Systems are fundamentally a collaborative 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.

[Illustration of an L-System, something that parallels the first one, and yet does something dramatically different (maybe the same image, just upside down? Inverted somehow?) Maybe just another cool L-System.]

1.1 Motivations

There are several motivations for this thesis, each which I hope will appeal to people from different disciplines with radically different interests. I hope that this thesis will be able to pull on threads in enough of a variety of disciplines that everyone will find something compelling in this short body of work.

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 of self-reference, universality, and beauty from simple rules arise in areas beyond just Computer Science. I hope to draw in Economists and Anthropologists in addition to Physicists and those from other areas with discussion of patterns that appear in the structures of society and social organization, and which can be illustrated to some extent with the methods that I outline here.

[Mathy L-System (Something Fractal)]

The second main motivation, one that I try to emphasize at every opportunity, is the stunning beauty of many of the patterns that I explore. This is the motivation that I imagine might be the most interesting to artists, be they visual artists or otherwise. This is 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 a huge variety of resources and tie-ins that connect the

Art aspects of this work to the Computer Science and Math aspects, some of which I will explore later in the thesis.

The third motivation is tied in with a philosophical understanding of these structures. I am not a trained philosopher, and I would not be surprised to learn if many of the questions that I ask as a part of my exploration are in well-tread ground. It's possible the questions I ask in this thesis will contribute to the conversations where I imagine they might find a home, but I don't expect them to. This motivation comes from that observation that I mentioned earlier that there are some ideas that seem to exist in disconnected disciplines. Some of the questions I ask address larger issues of the place of machines in the creation of art and the creative process. It feels like these are essential questions to be asking when the wider culture is experiencing what may be the first wave of publicly available and popular AI systems.

[Several more Fractal L-Systems]

While this is absolutely not a formal foray into the philosophy of these questions, I do hope that my readers will take my suggestions seriously. I present a thesis which asserts the importance of these systems that a person can hope to understand, in a time when so many of the systems that control our lives are invisible to us. This sort of question feels to me to be inseparable from the rest of the work.

I hope that there will be future interest in L-Systems, 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 the three outlined above, and I imagine that someone who sets new eyes on this work could see it in a way I never could have foreseen. I go into more detail near the end of the paper 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.

[A long, thin L-System which stretches from one side of the page to the other]

1.2 The Path We Will Follow

The first thing I will do in this thesis is formally introduce L-Systems. They will be explained and illustrated so that everyone who continues to read through my thesis will have at least some foundation for understanding. 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 do this. Different ways to interpret and draw L-Systems are discussed in Chapter 5. While they can look very compelling, they've been historically 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 at the end of Chapter 3.

[Some simple L-System (maybe one with step-like features?)]

I go on to talk about Generative Art and the broad crossover between Art and Computer Science that has fueled this thesis 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 structures, especially those of plants. *The Algorithmic Beauty of Plants*, 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. I talk about why I chose to include what I did and why I refrained from integrating some of the discussed ideas into this thesis before this section.

[Some of the images from ABOP]

Here I discuss possible extensions of the program. Some of these extensions are little more than quality of life changes while others might require a total rewrite in order to function. These extensions have one or both of two goals. The first goal is increasing usability and user-friendliness. The second goal is increasing functionality. 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.

This last section focusses on the limitations of L-Systems. First, I try to generalize L-Systems and clearly articulate their capabilities. I discuss dimensions and the way that L-Systems engage with different dimensions. I suggest possible alternative dimensions that could be explored and ways in which L-Systems displayed with those dimensions could look. The problem of depicting some of these theoretical dimensions will also be explored here, as well as why expanding understanding of L-Systems could be worthwhile.

I then conclude the thesis with a brief summary of the understanding I've gained through this process, the concrete objects that I have pulled into being, and the exploration of the larger ideas that lie under this whole process. I restate a few suggestions about 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, explore L-Systems and the larger world of Generative Art, and to be excited by work like this.

[Two L-Systems on the left and right, facing each other]

Chapter 2

What is an L-System?

This is an L-System:

[Picture of impressive looking L-system]

Well, it would be more right 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 just a few letters and symbols. Some Biologists may take offense, but it may be compared to DNA in how it compresses a potentially infinite complexity into a handful of characters. This chapter will cover a few main ideas. The first, and most central, is a discussion of the mechanics of how they work and how to build them. After that, I'll go on to describe some of their main features, some of their other traits and patterns, and their applications. As part of that last point, 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-System 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.

2.1 Understanding through Definitions

L-Systems have three main parts. The first part is the `start` variable. It is usually just a character, we'll call it `A` here. 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 changes. These are usually written like `A` $\rightarrow$ `AB` or `B` $\rightarrow$ `AA`, where each of these is a different rule. There are different names for what `A` and `B` are, but in this thesis I will call them "productions". In a similar fashion, each of these rules is called a "production rule". (Any character without a corresponding production rule is therefore not a production. This will be relevant later.) The third part of an L-System is a list of other information that is used when it is being constructed. For most of the L-Systems discussed in this thesis, there are two pieces of information in

this list, a value θ which is read as the `angle` and a value `Iterations` which is the number of times the rules are applied to the system. Alternative members of this list are discussed in Chapter 5.

Before I describe the process of making an L-System, I'd like to explain some vocabulary that I'll be using throughout this thesis. There are a few ideas that will hopefully make understanding these systems much easier to understand. The first term that I'd like to introduce to you is *string*. A *string* in Computer Science is a more expansive way of saying a *word*. A string does not need to be in any dictionary, however, 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./?;`. Strings do not need to be able to be spoken aloud to be valid. 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.

[(filler?) L-System that looks like a word]

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 however many times as 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 the production rule. (For example, if the L-string is `A`, and there is a production rule `A` $\rightarrow$ `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!

[Some extremely simple L-Systems]

2.2 Understanding through Images

L-Systems can be slippery 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.

The string can be transformed into the image and the image can be transformed into the string. No information is lost. I suggest alternative visualization strategies in Chapter 5.

This set of images is generated with the parameters, `start: A; rules: A → AB, B → AA;` θ `: 45°`. Different values for iterations are displayed in each image. `A` is drawn as a line going upward and `B` is drawn as a line going to the right. Each new production starts from where the last one left off.

[Image A0]: `Iterations = 0, current string: A`
[Image A1]: `Iterations = 1, current string: AB`
[Image A2]: `Iterations = 2, current string: ABAA`
[Image A3]: `Iterations = 3, current string: ABAAABAB`

Here is a different set of images with the same start variable and extra information, but alternative production rules. These rules are: `A → BAB, B → A`.

[Image B0]: `Iterations = 0, current string: A`
[Image B1]: `Iterations = 1, current string: BAB`
[Image B2]: `Iterations = 2, current string: ABABA`
[Image B3]: `Iterations = 3, current string: BABABABABAB`

These sets of images show the progression of an L-System between different iteration values. There is a sense that the L-System is "growing" here, fittingly tied to their origin as plant-modeling software. Look at the following L-System and see how the different sections can be seen in the output.

[L-System with the multiple very different production rules, and the effects of each production node can be clearly seen in the output]

2.2.1 Turtle Graphics

[An image of the turtle]

My early attempts to make L-System drawing programs all utilized a concept called "Turtle". Turtle 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. The Turtle operates by reading a series of instructions and drawing according to those instructions. This is particularly applicable for L-Systems, where each symbol is usually associated with a specific drawing operation. Even while making my Explorer from the ground up, I've kept in mind the Turtle strategy. As it's such a simple metaphor when figuring out how to draw structures like L-Systems, it's hard to draw in any way that doesn't fall in line with it in some way.

[A few simple turtle programs]

The central feature that Turtle uses to great effect is "state". "State" here means "condition" or "status". The Turtle keeps track of a few attributes as it goes through drawing the image described by the code. The attributes I focus on here are the Turtle's position and its rotation. 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 and choosing whether to draw or not draw between positions, the Turtle is able to draw a single path through every line that it draws. This is also particularly powerful because it physicalizes the process of coding that's often purely digital and accessible only through rough abstraction. This is a good abstraction for the purposes of this thesis.

[A variety of turtle-produced programs, some in color]

Chapter 3

Explaining the Program

In order to share my understanding about and my excitement for L-Systems, I wrote a program that allows users to directly change the parameters of the system without having to directly engage with any code. My central goal was to create a program that would reduce the barrier to entry of understanding these systems, and of understanding CS as a whole. My first explorations into making this space more accessible were done through turtle implementations in a basic Python program and later in a Blender[citation needed] program. I'll talk about these in more detail later in the thesis. Both of these strategies still heavily relied on the user to type input. It might as well have been code. It became clear that if I was to create something that was as accessible as I wanted that it to be, I would have to change my approach. In conversation with my advisor and through resources online for projects like mine, I came across *Processing*.

3.1 What is *Processing*?

[A blank Processing window]

Processing is a programming language built off of Java that has been made especially to work with image generation. It was released by Casey Reas and Ben Fry in 2001. 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 chose to engage with the act of coding in the same way that artists have traditionally engaged with their mediums. In other programming languages, it often takes some time to go beyond just printing simple messages to the screen; messages like `hello world`. The first programs that people write in Processing produce concrete images. 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 website dedicated to introducing newcomers to this art form, which you can access from the tab titled "Learn" near the top of the page.

3.2 Program Development

[A screenshot of one of the early versions]

The development of the program that I would eventually come to call 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. In the basic processing version of the Builder, the windows for the productions are able to be moved independently from the main display window. This doesn't add all that much to the program in terms of ability, but it allows users to move the windows in accordance with their whims. From there, I began to build up the framework of the Explorer itself. I constructed the system with the goal that it would be as intuitive as possible, even to someone with only a basic familiarity with computers. While this goal might be impossible, keeping it in mind helped prevent me from making the program too dependent on any past knowledge. 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.

[A screenshot of the most updated version I can include]

3.3 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 the Builder 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 3.4.

[Empty main window and empty sub-window]

The main window is initialized upon startup. 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 not needed. Each sub-window contains a reference to a 'starting' triangle that I fittingly named the `StartTriangle`. This is an instance of a class that can be moved up to the boundaries of its window. The start triangle 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. Each `StartTriangle` and its closely-tied `startNode` provide the foundation on top of which each of the production rule displays are constructed.

[Start Triangle and Start Node]

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 start node 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.

[Images that show how the mouse interacts with the nodes]

Each node of any subtype only knows about its neighboring nodes and the `option` nodes that refer to it as an `optionParent`. It does this by having a pointer to a single node going 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 final piece of reference information that a node contains is the array of `option` nodes that consider this node to be their `optionParent`.

[An image of the options, with the mouse hovering over an `optionParent` node]

[A diagram of the relationships between these nodes]

Beyond this vital reference information, each node contains some information about its state. Its state may be affected by calls sent by its `before` node or any of its `after` nodes. Each node is also able to send out some of these calls. One such communication is the status of a variable I named `activated`. This `activated` is either `true` or `false` and corresponds to if the mouse is over the node or the mouse was recently over the node. If `activated` is true, then some logic ensures that no other node is activated or can become activated while this node is activated. This happens through a method that calls itself in a node's `before` node and its `after` nodes. The exception to this is `option` nodes whose `optionParent` is the node which is already

`activated`. Each node also tracks its position and orientation, information that is referenced by this node's `after` nodes to ensure that every node is in the correct state.

[A diagram of the recursive-ness of the code that ensures that all the right nodes are activated]

There are specific ways for users to engage with this process. 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. For both of these possibilities, the current node's `after` array and the display are both updated.

[Two images showing how the deletion of a node affects the arrays and display (maybe actually four images? Two that show what is on the screen and two that show the changing relationship of the nodes behind the scenes?)]

Each sub-window has a function to turn its tree of nodes into a string that uniquely describes it. This is done through a 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. 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.

[Images showing the creation of a string based on some L-System]

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 start triangle or nodes within any of the sub-windows. It must have each sub-window do that. Some information must still be transferred between layers, information such as the production rules. The main window is able to reference the production rules from each sub-window. It applies the L-System recursive replacement process described in section 2.2 some number of times equal to the value of `Iterations`. This value can be changed by the user. The output of this process is the L-string, which is used to draw the L-System.

[A labelled drawing of the different strings and how they are related between the different windows]

In the process of drawing the L-System, the Explorer reads the L-string character-by-character and performs actions depending on that character. It acts very much like the Turtle described in section 2.2.1, even so far as keeping track of its state. It also has another data structure called a "stack". This stack can be thought of as a stack of books, or more accurately a spring-loaded stack of coins. The Explorer is only allowed to interact 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. (It cannot add a new item in the middle.) In this case, the kind of item that we're putting on the top of the stack is a snapshot of the current state.

[An illustration of the Explorer's stack and how the system engages with it]

When the Explorer reads a `[`, it puts the current state onto the stack. Whenever it reads 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. Every time it reads an `F` or `f`, it goes some distance in the direction it's facing. Each time it reads a `+` or a `{`, it adjusts its orientation by the current angle, changing where it's facing. 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.

This is how the L-System Explorer works! While this could certainly be less complicated, I feel proud to have been able to make a system that feels to me to be relatively sleek and clear in its purpose. It does succeed in hiding much of its complexity under the hood to give users as simple an experience as possible.

3.4 Using the L-System Explorer

I've written this L-System Explorer to be easily accessible. As I mentioned before, my central design goal was enabling people to engage with L-Systems without needing to know every detail. This project has what I feel is necessary to get a basic understanding of L-Systems. There were many judgement calls in the design of the work, though I hope that this program has achieved my goals.

[An image of the github page with the files and the processing website with all the relevant areas emphasized]

To access 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*. You will need the four files, `L_System_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 that this was written, this program

works in Processing version 4.2. From here, you can open any of the four files you just downloaded and you should something that looks like the image below. Click the play button in the top left corner and get started exploring!

[An image of the screen when the program is opened]

3.4.1 How to Explore

[Give images for each of the following steps]

The first time you use this program, hover over any of the circles in one of the smaller windows. When you hover over one of these circles, you'll see several more circles appear, each with a different symbol inside of them. These are buttons. Clicking on the button that says `F` will cause a line to 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. Also notice how the title of the window is "Production rule for A". Creating these elements has changed the production rule for `A`.

[Give images for each of the following steps]

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. Try clicking the rightward-facing triangle a few more times. Also try moving the angle slider. Look at what's happening!

[Give images for each of the following steps]

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. Hover over this new line and click its `A` button. 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.

[Give images for each of the following steps]

The last essential thing to learn about this program is how to add more productions and production rules. Look for a box that says "Productions". 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 effect.

[Provide as many interesting initial inputs as I can justify, perhaps 6? Show some techniques used to great effect.]

Chapter 4

Contribution

A primary goal of this work is to make L-Systems, and by extension Generative Art and Computer Science broadly, more approachable. This is not the 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 are few implementations that make L-Systems easy to play with. Even those who brought L-Systems to Blender still only make them available to those who know Blender.

The Blender work I mention is incredible, and I found it a great source of inspiration, but our projects have fundamentally different goals. As far as I'm aware, no other program has made L-Systems this easy to play with, despite the central part of their appeal being simplicity. As far as I can tell, there has been no easy way to gain an intuition with these systems that isn't locked behind years of engagement with computer science.

[Some sliders, the parts of the program that are especially simple]

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 these structures will help lower the barrier to entry. 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.

4.1 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.

[Three images that show the three programs in action]

The 1990 book *The Algorithmic Beauty of Plants* 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 plant created with L-Systems.

[Some images from ABOP]

Another useful resource for me has been the YouTube channel *The Coding Train*, and I'm grateful for the wide variety and expansive content 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're responsible for much of my foundational learning about Processing. If you have an interest in taking the first step in your computer science journey, I would be hard-pressed to find a better resource than this channel. 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. If you're interested, the tutorials on getting started in Processing are a good place to begin.

[Some images from the Coding Train]

Nikole Leopold wrote a Bachelor's thesis about an implementation of L-System graphics in Blender in 2017 at the Technical University in Vienna. 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 are L-Systems which can be animated continuously, by which I mean *between* production steps. [Include some footnote about how cool this is - it's animation to a crazy degree!] I'm grateful for the resources that Leopold included as well, as they pointed me in the direction of additional compelling sources.

[Some images from Leopold's paper]

Przemyslaw Prusinkiewicz, who worked with Aristid Lindenmayer on *The Algorithmic Beauty of Plants*, produced a handbook 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.

[Some images from this text]

Finally, the paper "A Theory of Animation: Cells, L-Systems, and Film" goes well beyond the scope of this thesis in its discussion. 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 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 4.3. 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.

[Some related image? (perhaps not, this'd be filler)]

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

4.2 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. 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. The point here is that there are resources available if you are at all interested in learning more about this. I have collected specific resources and put them in Appendix B for your perusal.

[Some weird & different Generative Art things]

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 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 4.3.1.

[One or two AI art things (maybe don't include this? Ethical concerns?)]

4.2.1 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. These were made in a program called OpenSCAD (pronounced "open S cad") and were a first attempt to create artworks through software.

[OpenSCAD projects]

This is a "mesh" that I was messing around with during these early stages as well. I was curious if I would be able to modify it so that I might be able to fold paper

into a physical model. This particular version never came about.

[Head Mesh]

However, I did manage to create a few physical models of geometric solids.

[Paper icosohedra]

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. Here are some examples of those.

[Modular Origami] (If you're interested, these specific types of objects are part of a discipline called "Modular Origami".)

I've already mentioned my early explorations into the native python implementation of turtle. Here are some of the systems constructed during that time.

[Early turtle SVGs]

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 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. 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. Here is the Hilbert Curve and a few other 3d constructions from L-Systems.

[3d Hilbert Curve and others]

It was at this point in the thesis that I encountered Processing. Here are images of some of the projects that I made as part of this particular exploration. Note that most of these are snapshots of a program in motion, and a lot of detail will be lost when we remove the time dimension. I would like to put some of these on my website at some point to let those who are interested to experiment.

[Stills from Processing animations, definitely the Simplex Cloud]

These projects were instrumental to the development of the thesis as you see it now. Without considering them, an understanding of my thesis is incomplete.

4.3 Quandaries, Puzzlements, and Musings

While developing this thesis and the various projects that came with it, I began to begin to see things through a specific 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. Machines makers have attempted to imitate life by different means for a similarly long period. Science Fiction authors and pioneers like Alan Turing have suggested ways in which we might learn if there is a difference between life and machines. Some of those questions have borne fruit or guided our development, [mention ideas which have guided our development, like the Turing test] others have just marked moments when we moved the goalposts. [include citations for this]

In the last decades, there have been tremendous advances in the simulation of life. Robots have beaten professional Chess and Go players, and have become indistinguishable from humans in many respects. We've become used to robot assistants in our homes and in our daily lives. The rise of AI-powered chatbots has thrown a wrench in the gears of several industries. Are we so sure that robots will never be able to do what we as humans can do?

[An especially beautiful, lifelike fern-like L-System]

L-Systems are not complicated, and I will not make the argument that they are themselves alive (though some have), [include citation to that one guy] but I suggest that they support a physicalist interpretation of the world. By that I mean the whole world, including humans and their apparently unique consciousnesses, are only the result of physical operations and no invisible, imperceptible, separate element. L-Systems demonstrate that an intense variety of complexity can easily arise from almost nothing. Why shouldn't people? [footnote some defense?]

4.3.1 The Times We Live In

At the time of writing, ChatGPT has taken the world by storm. While it doesn't appear to be destroying society yet, there's no saying what its successors will do. Regardless, it feels apparent to me that things are on the verge of changing dramatically. 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 everything we can do, but often faster and better?

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.

[An AI-generated image of "L-Systems". Let's see what it comes up with!]

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.

4.3.2 Is Math Art?

[Beautiful fractal and otherwise L-Systems]

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 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.

[Some more beautiful, artistic L-Systems]

I would be so bold to suggest that the products of the L-System Explorer *are* art. They qualify themselves enough for me to feel satisfied including them under that umbrella. 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 as they do without the involvement 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? This is despite them being clearly just math.

Chapter 5

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. Even in their 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 (Kelty and Landecker 46). "Theories become like the biology they purport to describe: natural, evolving forms with dynamics and feature all their own." (46) In other words, Lindenmayer was in part concerned with the creation of axioms of biological knowledge.

[Images of the earliest L-Systems]

Since their invention in 1968, L-Systems have found places in many notable cultural touchstones. They are responsible for the forests in Shrek, the opening sequence in Fight Club, and trees and other structures in many Pixar films. [Of course, include citations] They were the focus of some 5000 articles that were published by 1996. [include citation for this] It became clear that L-Systems were extremely useful for creating complex structures with minimal overhead, a feature which lent itself particularly well to simulations used in films. In a contrast to Lindenmayer's original goal of creating a formal structure for describing biological systems, L-Systems found themselves more helpful in other applications. The "productivity of the mathematical formalism would eventually overwhelm its usefulness as a biological theory".

[Show these references, ie the Shrek forests, the Fight Club opening, and the Pixar trees]

5.1 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 5.3, specifically regarding extensions to L-Systems that dramatically increase their capabilities and functionality. There It contains an abundance of thoughtfully demonstrated resources and examples, as well, 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.

[Share some more images from the book]

5.2 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 line color, length, and waviness. If I could, it would be good to add some extra precision in the angles to make the transition between different angles a bit smoother. It'd also be helpful to allow the user to move around the main screen and zoom in and out to some extent. 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. These safeguards could come in the form of bounds on the number of generations or possibly total number of line segments drawn. It would also be nice to have the ability to print out the final system to an SVG file or a PDF.

I'd also like to make it possible for users to type in the production rules for an L-System directly and bypass the drawing 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 someone else's productions directly into your instance of the Explorer. It would also be nice to have a small 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, I don't expect it to be. I also don't want to force users to read my thesis before engaging with the Explorer itself.

One problem I would like to solve is the messy overlap of 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.

[Show an image of the problem in action]

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.

5.3 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 avoided these extensions for two reasons, one reason being time and the other being a desire to keep the complexity of this thesis low so as to invite in people who might otherwise feel intimidated. 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 systems 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. Any basic context-free L-System can be made by a context-sensitive L-System where none of the productions actually depend on context. 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.

[A context sensitive L-System or two]

Stochastic L-Systems are a type of L-System that use randomness to construct different structures each time a particular L-System is drawn. 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.

[A stochastic L-System or two]

Parametric L-Systems are the most appealing and powerful variations on the L-System formula for me. They allow the different operations that the L-System does to no longer have to 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 parametric L-System or two (probably the set of images from Prusinkieweicz 1997 paper)]

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 $\texttt{A} \rightarrow \varepsilon$. This character, ε, symbolizes *nothing*. This production produces nothing in the generation after it is read.

[A parametric L-System or two (definitely the cool hex one from that one paper)]

These extensions of the basic L-System idea are covered more extensively in [papers that cover them more extensively]. Some variations, such as Environmentally-sensitive L-Systems, deserve a thesis to themselves. 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.

5.3.1 Interpretations

Up until this point we have limited our expression of L-Systems. Even in the expanded world that I've shown to you in this last section, I've refrained from pulling back the curtain completely from reality. I've explained that the content of an L-System is 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 dismissed assumption 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. These have still been trees. You might remember I mentioned the opening sequence of Fight Club as an example of L-Systems as work. 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. [citation to the video] 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*.

[More neurons or similar alternative interpretations of L-Systems]

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, if the right conditions are met.

[Some structures that could be made with L-Systems, possibly paired with attempts to display them with L-Systems]

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 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.

Chapter 6

Conclusion

L-Systems have the ability to 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

Extra Images

[Put some of the reaction diffusion tests here]
[Also the Perlin Cloud]
[Lorenz Attractor as well]

Appendix B

Generative Art Resources

Citation to the generative art subreddit
Citations to some of the cool youtube channels
Instagram, Twitter, etc tags of some relevant artists