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 especially Gabe Fish for the late night conversations that led to some of the more interesting choices made and questions raised throughout this document.

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 introduction of 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 the most powerful, subtle, but familiar disciplines: 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 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 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, 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 philosophical 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 Builder.

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 Iterations value. If the Iterations value 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 an associated production rule, then it is left alone.

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 $\rightarrow$ AB, B $\rightarrow$ 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 $\rightarrow$ BAB, B $\rightarrow$ 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 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 that might as well be code. It became clear that if I was to create something that actually met my goals that 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*?

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

The development of the program that I would eventually come to call the L-System Builder started with a series of drawings describing the different functionalities I was aiming to include. I wanted everything to be able to be accessed without touching the keyboard. There has 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.

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 Builder itself. I constructed the system with the goal that it would be as intuitive as possible, even to someone who isn't all that comfortably working with technology. While this goal might be impossible, keeping it in mind helped prevent me from making the program too dependent on any past knowledge. The structure of the Builder changed throughout its development, but I'll describe the form that it has taken at the time I'm writing this.

3.3 How it works

I talk about the gritty details of the L-System Builder 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 the Access and Use section.

The main window is initialized upon startup. It then creates 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 triangle that I fittingly named the start triangle. This is an instance of a class that can be moved up to the boundaries of its window. The triangle has a reference to the starting node, which is the root of a linked list structure of nodes. I've made a class to describe the nodes as well. When the start triangle is moved, the start node is moved similarly, and then recursive calls down the node tree move all the nodes to the correct positions. These start triangles and their closely-tied start nodes provide the foundation on top of which each of the production rule displays are constructed.

There are five divisions among the node class, each node belongs to only one of these divisions. The start node is the only element of the “start” division; I use this as a way to stop upward recursive calls 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 division is “forward” nodes. These are displayed as lines, unlike the other types of nodes, which are displayed as circles. The final division is “option” nodes. These are a special type of node with some unique characteristics. They have a pointer to a “parent” node, which may be a member of any division of node, including another option node. They will only appear if their parent node is being hovered over by the mouse or their parent has recently been hovered over by the mouse and the mouse has not left the vicinity of the parent or any of the parent’s option nodes. Each node of any division only knows about its neighboring nodes and the option nodes that refer to it as a parent. It does this by having a pointer to a single node going in the direction of the root 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 knows is the array of option nodes that consider this node to be their parent.

Beyond this vital reference information, each node contains some information about its state. Its state may be changed by upward or downward recursive calls. Each node is also able to send out some of these recursive calls in order to affect the status of the other nodes. One such communication is the status of a variable I named “activated”. This “activated” is a boolean (true or false) for whether or not a given node has been hovered over by the mouse or recently was hovered over by the mouse. If this value is true, then I’ve built some logic to ensure that no other node is activated or can become activated while this node is activated. The exception to this is option nodes whose parent is the node which is already activated. Each node also tracks its position and orientation, information that is referenced by the nodes in this node’s “after” array to ensure that every node is in the correct state.

The user may click on the option nodes of an activated node and if the clicked node has some associated action, something will happen. Usually this something 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). It can also be that the option node that was clicked will trigger the deletion of that option node’s parent node. For both of these possibilities, the current node’s “after” array is updated to match reality, and the display is updated accordingly as well.

Each sub window has a function to turn the tree of nodes into a string that uniquely describes it. This is done through a process similar to 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 “]”. Each node has a character that it is associated with. Rotation nodes have either a “+” or a “–”. Production nodes have the character of their production. Forward nodes have the character “F”. Start nodes are an exception to this, as they have no character, and the same is the case for option nodes, as they are never read by the function. These associated characters are added whenever a node of the appropriate type is read by the function. At the end of this process, the sub-window is left with a string that is the production rule for the production of that sub-window.

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 its 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. There are, however, pieces of information that must be communicated between these layers. One type of such information is the production rule just described. The main window is able to reference the production rules from each sub-window. It applies the L-System recursive replacement process (as described in Chapter 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.

To draw the L-System, the Builder reads the L-string character by character and performing actions based on that character. It acts very much like the Turtle described in Chapter 2, 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 Pez dispenser, or a vending machine. The Builder is only allowed to interact with the top item on the stack, and it can choose to read it or 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 Builder’s state. When the Builder 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 that’s on the top of the stack. Every time it reads an “F”, it goes some distance in the direction its 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 Builder 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 Access and Use

I've written this L-System Builder to be easily accessible. As I mentioned before, my main design goal was to make it as friendly to computer novices as possible. What remains is what I feel is necessary for someone to get a basic understanding of L-Systems. There were many judgement calls in the design of the work, though I hope that my choices will be able to stand on their own merit.

To access this program, go to "rjacobh.github.io/website" and click on the tab on the top of the page that says "Processing".

The first time you're using this program, a way to get started is to 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. Try moving your mouse over the button that says "F" and click on it. You should see a line appear! If you look at the large window, a line will also have appeared! If you put the mouse over the line you just made in the small window, you'll see more buttons appear. Try clicking on the "+" button This will make a circle with a point appear in the small window. The big window won't have changed. Now, go over the new shape with a "+" on it and click on the button that appears that says "F". This will make another line appear in both the small window and the big window.

Look around the page now for a slider that says "Angle". The slider looks like a small gray rectangle over a long black line over a much larger gray rectangle. If you put your mouse over the small gray rectangle, press down, and drag the mouse, the lines that you drew will move! When you're done moving that slider, you can stop pressing the mouse. 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 look around the page for a box that says "Iterations". This one is fairly small with two black triangles pointing in different directions. Click the triangle on the right. You should see the number in the Iterations box go from 1 to 2, and the size of your line in the big box double in size! Try clicking the right triangle a few more times. Try doing this a few times then moving the angle slider. Look at what's happening!

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 "–" circle and click the "F" button that appears. 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. The last 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 with a "+" inside of it. 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 type 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.

Chapter 4

Contribution

A primary goal of this work is to make L-Systems, and by extension, generative art and even Computer Science as a whole, less frightening. I want folks who have never used computers become able to create things on their own initiative to be amazed and inspired by the possibilities. 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 feel that these are all intensely possible things without having to engage with the complexity of modern design software like Blender, Photoshop, Fusion 360, or others, though those absolutely have their place.

I am fully aware that L-Systems have been around for 55 years, since they were proposed back in 1968, and for a time 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 Nikole Leopold, a grad student at the Technical University in Austria who made a brilliant program to bring L-Systems to Blender, still locked away engaging with these systems behind a fairly complex system. Not to mention that Blender needs to be downloaded to be used.

Leopolds work is incredible, and it should be said that I found it a great source of inspiration while I was bringing my thesis together, but our projects have fundamentally different goals. As far as I'm aware, no one has made L-Systems this easy to play with. Despite the central part of their appeal being their simplicity, I haven't seen any implementations that allow people to play around with them freely. I feel comfortable asserting that there is no easy way to gain an intuition with these systems that isn't locked behind years of engagement with computer science. I want to change that as much as I can.

My central contribution has been the development of the interactive L-System Builder. I hope that it's clear that this project will be able to engage with people, no matter their dedication to CS. I hope to have used these structures to lower the bound for entry into this discipline, or at least increase my readers' excitement to engage with CS. The field is clearly much more than pretty pictures, but 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 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 into Blender, and the final L-System Illustrator that I've polished and for which I've added instructions. 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 briefly here before expanding my view to include other visual structures that I feel are worth exploration.

A document that grounded my study from the beginning was the 1990 book *The Algorithmic Beauty of Plants*. As I've said before, this opened my eyes to the beauty of L-Systems and helped me begin to see the possibilities of exploring this space. I recommend taking a look at 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 plant pictures.

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. If you're interested, the tutorials on getting started in Processing are a good place to begin.

A graduate student at the Technical University in Vienna by the name of Nikole Leopold wrote a Bachelor's thesis about an implementation of L-System graphics in Blender in 2017. I would have liked to come across this paper earlier in my process, but it still has been able to inform 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 directly 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 more compelling sources.

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. This is another text that I wished I could have encountered earlier in the process. I don't know how much it would have shaped the course of my research, but it remains fascinating and a helpful source for illuminating pictures.

There was one other paper that I'd like to bring your attention to, this paper being titled, "A theory of Animation: Cells, L-Systems, and Film". This paper goes off the rails in the best way, and it provides some insight that is helpful more so for the larger questions about this project, 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 the section titled, "Philosophical Quandaries". This thesis attempts to make L-Systems able to be understood by a broad audience. I believe that it would be impossible to do L-Systems justice by approaching them as a purely mathematical concept. In order to appreciate their flavor and unique character as a life simulator, I want to bring in ideas from sources like this one. 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.

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 Automata. These have been used alongside L-Systems to generate life-like effects for film and other media, as they offer a different balance of capabilities. Their simplicity also makes them a great tool for creating emergent systems and acting as models for complex behavior. They can also demonstrate a remarkable capability for beauty, a topic that I will delve into more now.

4.2 Generative Art

There exists a major subset of the creations made by the cooperation of machines and people which we have come to see 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*. I'll refrain from a hard definition, but broadly stated for the purposes of this thesis, Generative Art is 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.

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 will hold off on discussing them until a little later, in the Quandaries, Puzzlements, and Musings section.

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

Here is where I feel it's appropriate to ask some of the big questions. I know it's infeasible to try to answer any of these questions, much less all, but I think there's still something valuable about asking them. I've wondered about why this work

I'm going to break this section into two main parts.

Start with an intro explaining and defending the existence of this section,

4.3.1 Is Math Art?

Is this art? Where do we draw the line between art and math?
Where does the simulation of life fall into this?

4.3.2 The Times We Live In / Flights of Whimsy

What does it mean for a machine to make art?

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.

Over the last decades, L-Systems have found places for themselves 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. They were the focus of some 5000 articles that were published by 1996. 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".

5.1 The work of Przemyslaw Prusinkiewicz and Aristid Lindenmayer

As I've mentioned before, the book that brought me into awareness of these systems was *The Algorithmic Beauty of Plants*. This is 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, diving more deeply into the foundations of this work and explaining things that I've left unsaid.

There are some compelling parts of the book that go well beyond what I discuss in this thesis, 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.

5.2 Extending the Program

There are so many extensions that I envision for the L-System Builder. An early change 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 anything. Another general improvement along the same lines would be to integrate the system into its host website as much as possible. One possible improvement along those lines is optimizing the site for mobile use and not lose functionality. I'd also like to shift my attention from the processing version to the new javascript version, and develop the program in that language instead. This would shorten the pipeline between writing the code for the program and bringing the results of that code into the hands of interested people.

Beyond just those considerations about porting, I'd like to make a number of improvements to the program itself. There are a few attributes that I haven't given the user control over which I believe could be done without too much heavy lifting. These would be things like color, line width and length, and line waviness. It would also be nice to add some extra precision in the angles to make the transition between different angles a bit smoother. It would be good to have a secondary slider for the angle that would control the offset of the angle from the vertical, allowing for a wider range of constructions. I'd also like to enable the user to move around the final production screen and zoom in and out to some extent. Another early addition would be a handful of 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. It would also be nice to have the ability to print out the final system to either 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 process 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 own instance of the Builder. It would also be nice to have a small library of cool L-Systems that others have made that could be hosted in such a way that every user of the Builder would be able to access them. On that note, it would be good to make some sort of tutorial about how to work the program. This last point is especially important. While I would love if the system was intuitive enough to understand purely from looking at it, I don't expect it to be. I also don't want to have this thesis document be a roadblock to engaging with the Builder itself.

Of course, I hope to also resolve whatever bugs I come across. There are a few features that I've discovered problems with that also go beyond just being bugs. One such problem is the overlap of nodes. When one production node is selected immediately after another production node, it's impossible to select the earlier of those production nodes. The problem also exists when the more-recently placed node is a rotation node. 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 Builder. 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 people in 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 the possibilities will be more expansive than with the context-free L-Systems we've been seeing 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 are interesting in a couple ways, but I find it most interesting that they seem to contradict the fundamental argument for why L-Systems are compelling. That reason being the rise of apparent complexity out of a set of simple rules, and specifically complexity that is deterministic. There is something interesting about how these might demonstrate the effects of mutation in an organism. These are the L-Systems that often 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 one of the most personally appealing and powerful variations on the L-System formula. 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 in terms of 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. 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*. If 0 is nothing among numbers, ε is nothing among strings. This production simply erases itself whenever it is called. These are actually implicitly included in the L-System Builder, as if a production is referenced and its production rule doesn't contain any other symbols, then that production is just erased whenever 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 Environmentallysensitive L-Systems, seem to 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 leave it to those who have made it a larger project of their life to illuminate this audience about the details of these beautiful systems.

5.3.1 Interpretations

Up until this point we have been limited in 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 suggested before that the actual content of an L-System is fundamentally just the combination of the different production rules that can form an L-string when put together. Throughout this thesis, I have only shown you a specific group of interpretations of L-Systems. By this I mean that all of the examples that I've given up to this point have fallen into the overlapping categories of trees, plants, and black and white branching structures. We can go beyond this.

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 you look for that video so you have a clear example of what I'm referring to. These structures don't seem to be much like trees at all, and yet they were created by the same recursive replacement structures that are responsible for the variety of trees scattered through the work you're reading right now. At the level of an L-string, I suspect that the neurons would be essentially indistinguishable from all the L-strings I've shown you. That's because they are the *same structure*.

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. (This list was partially generated by Chat-GPT 3.5.) I can even imagine the possibility of L-Systems being able to generate music, if the right conditions are met.

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 to this point. While they have been extensively studied mathematically, that doesn't mean that they're exhausted of compelling material. I feel that we are in an especially compelling position regarding L-Systems, seeing as so much of their power has already been documented. We can build off of the strong foundations that were first established 55 years ago 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 AIs that we've built, and be all the better for it.

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 a fundamentally simple structure. They have been a powerful item 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 Builder, a tool that brings L-Systems to anyone who is remotely interested. This tool brings the promise of L-Systems as a way to describe complexity in simple terms full circle; they themselves can finally be taught with this Builder 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 my subject as well as the importance of the methods by 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 also goes beyond these systems and into more everyday realms. I hope you've enjoyed this work. Take care.

Appendix A

Extra Images

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

Appendix B

Generative Art Resources