Computational Biology: A mathematical approach to visualize plant formation

Gabriel Martinez

Physics 409: Final project – L-Systems

Abstract

Plants support basic daily functions of human life by providing food and nutrition, supplementing medicines and cosmetic products and serving as important ingredients in a variety of medicines[1]. Yet, botanical modeling is among the many challenging systems to model due to its complexity of its intrinsic biochemical processes[2]. To eliminate such complexity, the idea in this paper is to explore the Lindenmayer (L-systems); a mathematical framework that can describe the fundamental formation of botanical structures as observed in nature. Namely, the goal here is to explore the parametric space of L-system[3] (also called D0L-system) and test its reliability as a framework for modeling the intrinsic characteristics of plant formation by attempting to recreate several images as depicted in **figure 5**.

I. Background

The mathematical model chosen to explore plant formation in this paper was introduced in 1968 by Aristid Lindenmayer a theoretical Biologist. Lindenmayer established a formalism for simulating the development of multicellular organisms (for example, algae) in terms of division, growth, and death of individual cells, subsequently named L-systems [3,4]. This formalism was closely related to abstract automata and formal languages, and attracted the immediate interest of theoretical computer scientists. The L-systems gained popularity around the 1980's primarily due to:

1) recognition of the fractal character of structures generated by L-systems, which related them to the dynamically developing science of fractals [6, 7, 8];

2) increased understanding of the modeling process, providing a methodology for constructing models according to biological observations and measurements [9, 10].

II. Methods

The essence of using L-systems is to describe plant development at the modular level. All modules belong to a finite alphabet of module types; thus, the behavior of a large configuration of modules will all be defined and derivable by a finite set of productions. In other terms, a string *w* is generated by a context-free grammar *G* from a finite set of productions *p* all from a finite set of alphabet Σ.

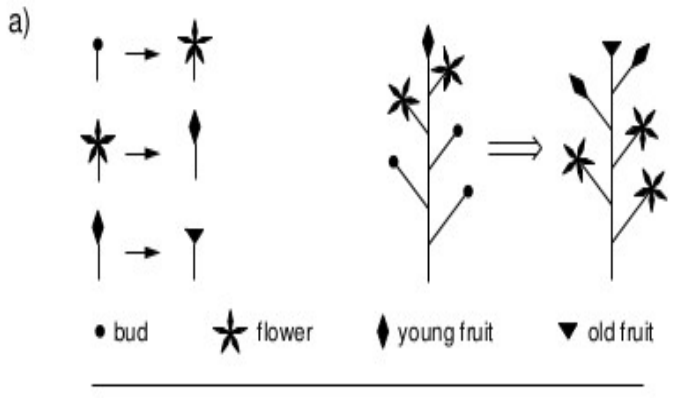


Figure 1: Example of productions: (a) a primitive idea of productions that demonstrate the formation of a plant.

Figure 1 is an example that demonstrates a sequence of finite productions:

p1: B => F

p2: F => YF

p3: YF => OF

where B=bud, F=flower, YF=young fruit, OF=old fruit.

Applying production(s) to an initial axiom will generate a growing sequence of alphabets *i.e.* a string *w.*

With the idea of productions in mind, we now move on to describe the importance of parametric D0L-system. If there is exactly one production for each symbol, then the L-system is said to be *deterministic* (a deterministic context-free L-system or *D0L-system*). If there are several, and each is chosen with a certain probability during each iteration, then it is a *stochastic* L-system. This paper will only focus on D0L-system. The D0L-system is defined as an ordered quadruple $G = \langle V, \Sigma, w, p \rangle$, where:

V - is the alphabet of the system,

Σ - is the set of formal parameters,

$w \in (V \times \text{Ŗ}^{*1})$ is a nonempty parametric word called the axiom.

$P \subset (V \times \Sigma^{*}) \times \varsigma(\Sigma)^{2} \times (V \times \varepsilon(\Sigma)^{*3})^{*}$ is a finite set of productions.

Strings (*w)* can be scanned from left to right sequentially and transformed geometrically and represented in three dimensions as a turtle [11]. It has state, which consists of the turtle position and orientation in the Cartesian coordinate system, as well as various attribute values, such as current color and line width. The position is defined by a vector $\vec{P}$, and the orientation is defined by three vectors $\vec{H}$, $\vec{L}$, $\vec{U}$, indicating the turtle's directions. These vectors have unit length, are perpendicular to each other, and satisfy the equation $\vec{H} \times \vec{L} = \vec{U}$. Rotations of the turtle are expressed

by the equation:

$$[\vec{H}' \vec{L}' \vec{U}'] = [\vec{H} \vec{L} \vec{U}] R$$

where **R** is a 3 x 3 rotation matrix[12]. The following list specifies the basic set of symbols interpreted by the turtle. The parameters affect the turtle's state. If no parameter is given then default values specified outside the L-system are used.

1 Ŗ* is the set of all finite sequences of parameters.

2 $\varsigma(\Sigma)$ denotes a logical expression with parameters from and

3 $\varepsilon(\Sigma)$ is an arithmetic expression with parameters from the same set.

3a)

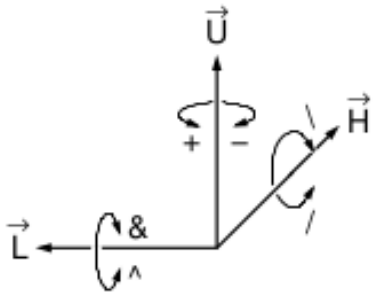


Figure 3a: Controlling the turtle in three dimensions.

3b)

Symbols that cause the turtle to move and draw

$F(s), G(s)$ Move forward a step of length s and draw a line segment from the original to the new position of the turtle.

$f(s), g(s)$ Move forward a step of length s without drawing a line.

$@O(r)$ Draw a sphere of radius r at the current position.

Symbols that control turtle orientation in space (Figure 6a)

$+(\theta)$ Turn left by angle θ around the $\vec{U}$ axis.

$-(\theta)$ Turn right by angle θ around the $\vec{U}$ axis.

$\&(\theta)$ Pitch down by angle θ around the $\vec{L}$ axis.

$\wedge(\theta)$ Pitch up by angle θ around the $\vec{L}$ axis.

$/(\theta)$ Roll left by angle θ around the $\vec{H}$ axis.

$\backslash(\theta)$ Roll right by angle θ around the $\vec{H}$ axis.

| Turn 180° around the $\vec{U}$ axis. This is equivalent to $+(180)$ or $-(180)$.

the

Symbols for modeling structures with branches

[Push the current state of the turtle (position, orientation and drawing attributes) onto a pushdown stack.

] Pop a state from the stack and make it the current state of the turtle. No line is drawn, although in general the position and orientation of the turtle are changed.

is given.

$!(w)$ Set line width to w, or decrease the value of the current line width by the default width decrement if no parameter is given.

$;(n)$ Set the index of the color map to n, or increase the value of the current index by the default colour increment if no parameter is given.

$,(n)$ Set the index of the color map to n, or decrease the value of the current index by the default colour decrement if no parameter is given.

Figure 3b): shows the symbols used to manipulate the turtle.

III. Results and Analysis

I am reluctant to admit I could not achieve my goal to produce a 3D representation set out in this paper. My goal was to use **figure 4** 's axiom and production rules to recreate the images depicted in **figure 5.**

I was only able to produce 2 out of the 9 pictures correctly and I know why. Initially, I chose to use the Turtle Python library[13] which natively works only in Two-Dimensional space. I tested the program only once and it appeared to work; I was invigorated thus blinded by this false hope of achievement too only realize I overlooked the rotation about the Z axis. Looking back on this blunder, one can only think how can I forget such a crucial part of the program? The lack of understanding of both python and the idea of this research played a role. I tried frantically to rewrite the program in Three-Dimensional space but due to the program's lengthiness, I was unsuccessful.

The reason I am able to produce the first two images successfully is due to the constants φ_1 and φ_2 both being zero. Since, both constants are zero then according to **figure 4** a rotation around the Z axis will NOT occur. My program does correctly generate the resulting string (*w*) from the productions defined by **any** D0L-system. However, my program will only work in two-dimensional space; I'm confident that this problem can be easily remedied, but for now I demonstrate for what I could produce.

Expected results:

$$\begin{aligned} \omega &: \quad A(100, w_0) \\ p_1 &: \quad A(s, w) \quad : s >= min \quad \rightarrow \quad !(w)F(s) \\ &\qquad\qquad [+(\alpha_1)/(\varphi_1)A(s * r_1, w * q \wedge e)] \\ &\qquad\qquad [+(\alpha_2)/(\varphi_2)A(s * r_2, w * (1 - q) \wedge e)] \end{aligned}$$

Figure 4: a parametric equation that can produce results shown in figure 5.

Figure 5: Sample structures generated by a parametric D0L-system with different values of constants using the productions rules and axiom as stated in figure 4.

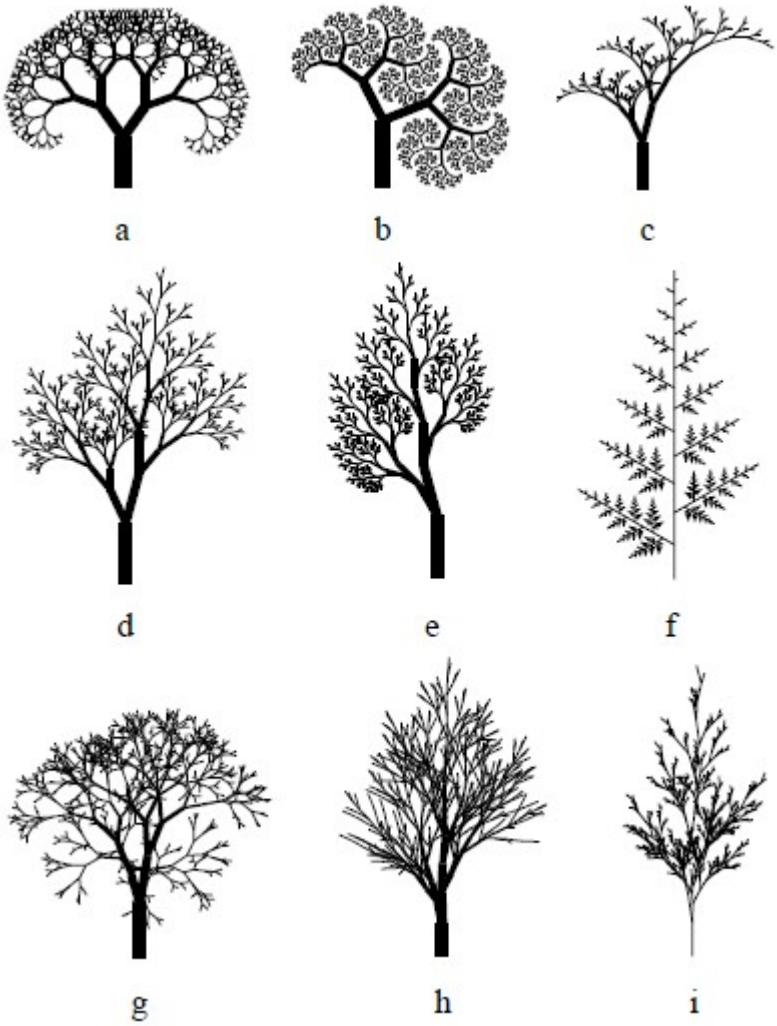
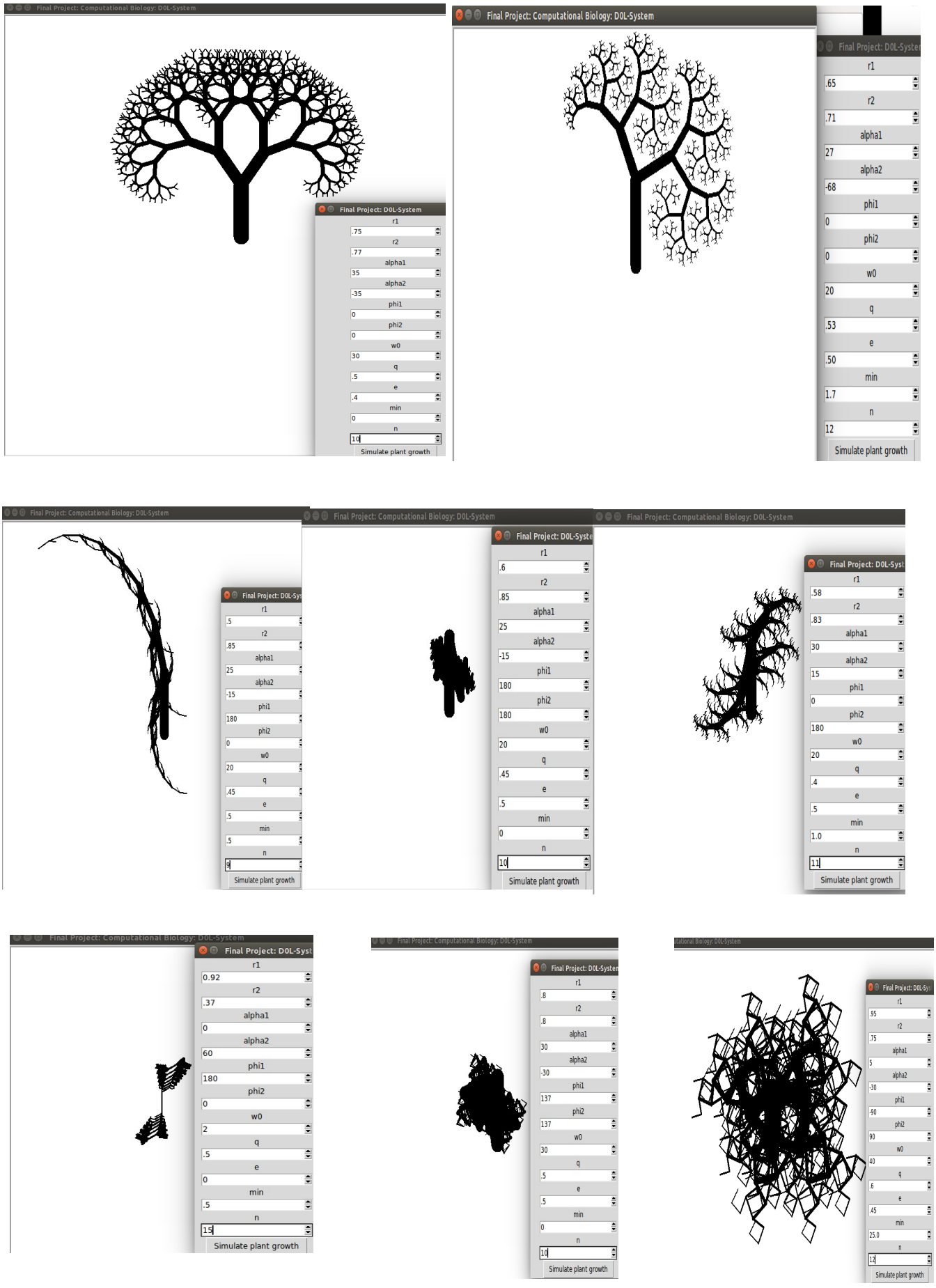
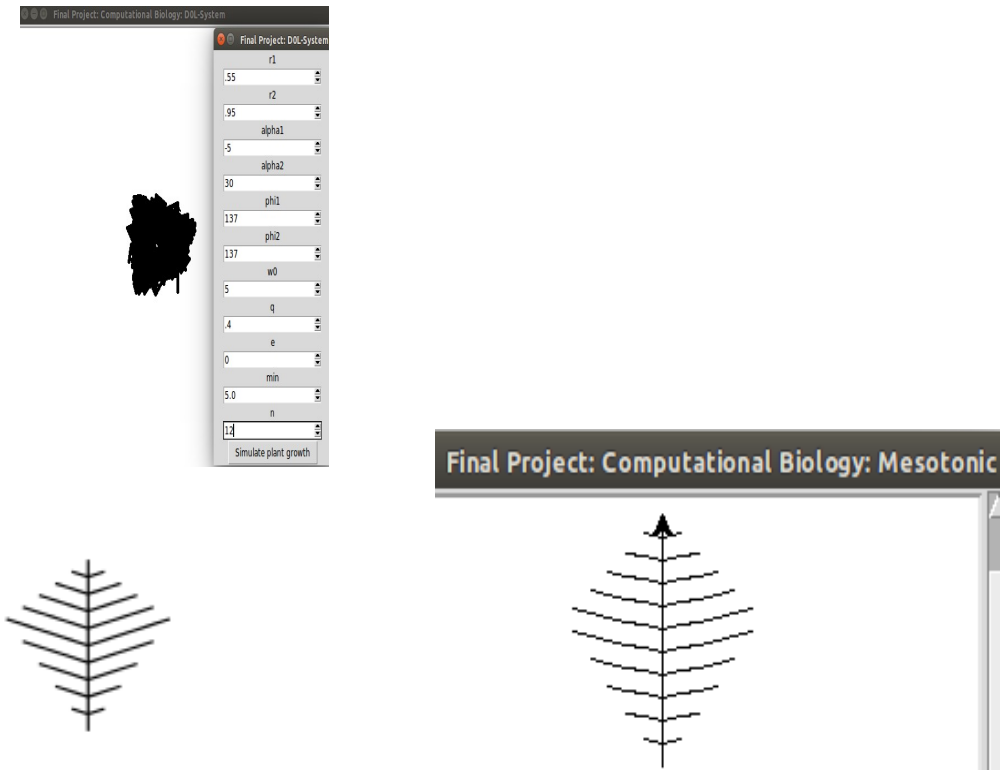


Figure	r_1	r_2	α_1	α_2	φ_1	φ_2	w_0	q	e	*min*	n
a	.75	.77	35	-35	0	0	30	.50	.40	0.0	10
b	.65	.71	27	-68	0	0	20	.53	.50	1.7	12
c	.50	.85	25	-15	180	0	20	.45	.50	0.5	9
d	.60	.85	25	-15	180	180	20	.45	.50	0.0	10
e	.58	.83	30	15	0	180	20	.40	.50	1.0	11
f	.92	.37	0	60	180	0	2	.50	.00	0.5	15
g	.80	.80	30	-30	137	137	30	.50	.50	0.0	10
h	.95	.75	5	-30	-90	90	40	.60	.45	25.0	12
i	.55	.95	-5	30	137	137	5	.40	.00	5.0	12

Figure 6: Experimental results.





7a) 7b)

Figure 7: (a) Mesotonic structure. (b) experimental mesotonic structure.

Despite, my failure to produce a three-dimensional representation of the D0L-system, I was able to produce a mesotonic structure given by the D0L-system:

$$\begin{array}{lll} \omega & : & FA(0) \\ p_1 & : & A(v) \quad \rightarrow [-FB(v)][+FB(v)]FA(v+1) \\ p_2 & : & B(v) : v > 0 \quad \rightarrow FB(v-1) \end{array}$$

. As observed by Frijters and Lindenmayer [14], and formalized by Prusinkiewicz and Kari [15], arbitrarily large mesotonic structures cannot be generated by non-parametric deterministic 0L-systems with subapical branching.

IV. Conclusion

The Linden-mayer system is great mathematical framework to eliminate such complexity in modeling botanical systems. Despite, my attempt to generate code to produce three-dimensional D0L-systems, I learned key ideas behind the fundamental modeling of plant formation. I believe this research leads to a further exploration of the mathematics because many of the papers I read did not mention **how** they came about the production rules or **why** they chose to start with the initial axiom that they did. Further exploration is needed to find an approach that is less heuristic.

References

[1] "Botany." - *Complete University Guide*. Web. 08 Mar. 2016.

[2] "Untangling the Quantum Entanglement Behind Photosynthesis: Berkeley Scientists Shine New Light on Green Plant Secrets | Berkeley Lab." *News Center*. 2010. Web. 08 Mar. 2016.

[3] Przemyslaw Prusinkiewicz, Mark Hammel, Jim Hanan, and Radomir Mech. L-systems: from the theory to visual models of plants. In M. T. Michalewicz (ed.), *Plants to ecosystems: Advances in computational life sciences I.*, pp. 1-27.

[4] A. Lindenmayer. Mathematical models for cellular interaction in development, Parts I and II. Journal of Theoretical Biology, 18:280–315, 1968.

[5] A. Lindenmayer. Developmental systems without cellular interaction, their languages and grammars. Journal of Theoretical Biology, 30:455–484, 1971.

[6] P. Prusinkiewicz. Graphical applications of L-systems. In Proceedings of Graphics Interface ’86 — Vision Interface ’86, pages 247–253, 1986.

[7] P. Prusinkiewicz and J. Hanan. Lindenmayer systems, fractals, and plants, volume 79 of Lecture Notes in Biomathematics. SpringerVerlag, Berlin, 1989 (second printing 1992).

[8] P. Prusinkiewicz and A. Lindenmayer. The algorithmic beauty of plants. Springer-Verlag, New York, 1990. With J. S. Hanan, F. D. Fracchia, D. R. Fowler, M. J. M. de Boer, and L. Mercer.

[9] P. Prusinkiewicz, W. Remphrey, C. Davidson, and M. Hammel. Modeling the architecture of expanding Fraxinus pennsylvanica shoots using L-systems. Canadian Journal of Botany, 72:701–714, 1994.

[10] P. M. Room, J. S. Hanan, and P. Prusinkiewicz. Virtual plants: new perspectives for ecologists, pathologists, and agricultural scientists. Trends in Plant Science, 1(1):33–38, 1996.

[11] H. Abelson and A. A. diSessa. Turtle geometry. M.I.T. Press, Cambridge, 1982.

[12] J. D. Foley and A. Van Dam. Fundamentals of interactive computer graphics. Addison-Wesley, Reading, 1982.

[13]"24.5. Turtle — Turtle Graphics for Tk¶." *24.5. Turtle*. Web. 18 Mar. 2016.

[14] D. Frijters and A. Lindenmayer. Developmental descriptions of branching patterns with paracladial relationships. In A. Lindenmayer and G. Rozenberg, editors, Automata, languages, development, pages 57–73. North-Holland, Amsterdam, 1976.

[15] P. Prusinkiewicz and L. Kari. Subapical bracketed L-systems. In J. Cuny, H. Ehrig, G.

Engels, and G. Rozenberg, editors, Graph grammars and their application to computer science; Fifth International Workshop, Lecture Notes in Computer Science 1073, pages 550–564. Springer-Verlag, Berlin, 1996