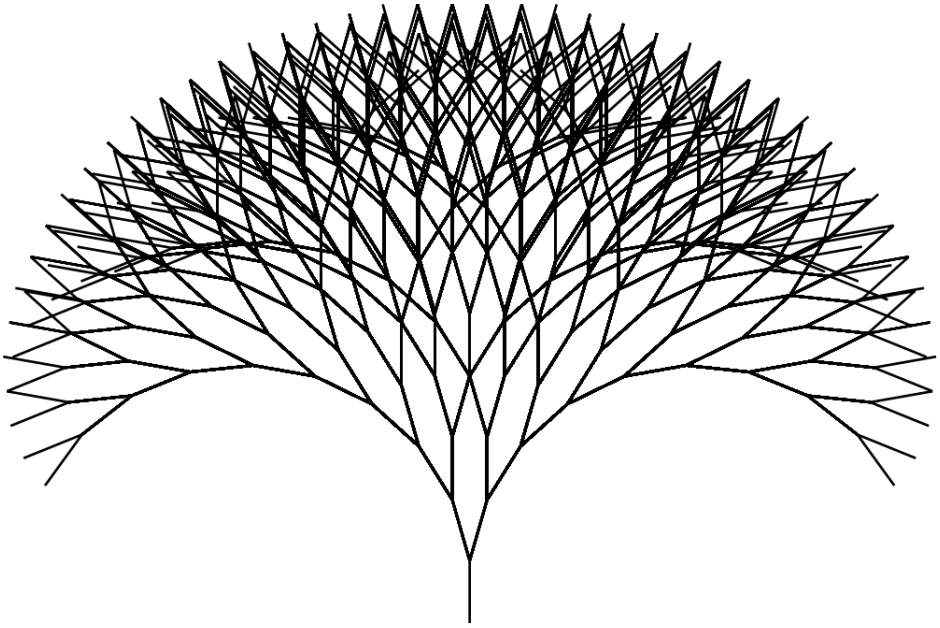
Procedural Plant Generation and Simulated Plant Growth

Massey University



Matthew Crankshaw

25 February 2019

Acknowledgements

Abstract

Contents

List of Figures

Chapter 1

Introduction

Here I will introduce the project and the thesis in general.

1.1 Context and Background

One of the most time consuming parts for digital artists and animators is creating differing variations of the same basic piece of artwork. In most games and other graphics applications environment assets such as trees, plants, grass, algae and other types of plant life make up the large majority of the assets within a game. Creating a tree asset can take a skilled digital artist more than an hour of work by hand, The artist will then have to create many variations of the same asset in order to obtain enough variation that a user of that graphics application would not notice that the asset has been duplicated. If you multiply this by the number of assets that a given artist will have to create and then modify, you are looking at an incredible number of hours that could potentially be put to use creating much more intricate and important assets.

The unique thing about plant life when compared with other types of graphics assets is that plant life is very random in the way it grows and it does not take an incredibly realistic model of a plant life to trick the human mind into believing that what it is seeing is some kind of plant. However what stands out like a sore thumb is when plants that are growing next to each other seemingly have no influence on the plant life around them.

Chapter 2

Outline

2.1 Outline

Chapter 3

Introducing Lindenmayer Systems

Aristid Lindenmayer is well known biologist who started work on what would become known as the Lindenmayer System or L-system for short. Originally they were intended to be used as a grammar for describing the development of simple organisms. However over the years they have been found to be useful in describing larger organisms and even non organic structures like music. [Worth and Stepney, 2005]

3.1 L-system grammars

According to Prusinkiewicz and Hanan a simple type of L-systems are those known as deterministic 0L systems, where the string refers to the sequence of cellular states and '0L system' abbreviating 'Lindenmayer system with zero-sided interactions'. With 0L systems there are only three major parts. There is a set of symbols (*alphabet*), the starting string (*Axiom*) and state transition rules (*rules*). The alphabet is a set of states. The starting string is a starting point containing one or more states. The transition rules are rules that dictate whether a state should remain the same or transition into a different state or even disappear. [Prusinkiewicz and Hanan, 2013].

An example of a deterministic 0L system is that a simplified model for the growth of algae:

We are given the *alphabet*: A, B
and the *axiom*: A
and the *rule* set:
$A \rightarrow AB$
$B \rightarrow A$

If we then apply the rules to the L-system we find it creates the following generation structure.
1.) A
2.) AB
3.) ABA
4.) ABAAB
5.) ABAABABA
6.) ABAABABAABAAB

This rewriting of initial string using a set of rules is ultimately the underlying concept behind L-systems. There are a number of improvements that can be made to this type of L-system in

order to accommodate for more complex and intricate structure. One of which is the inclusion of *constants*. Constants can be considered any state that does not have a rule associated with it or remains the same from generation to generation and therefore holds a consistent value or meaning. These constants are used when the L-system is interpreted and thus holds a constant value during string rewriting.

3.2 Interpreting L-systems

Once we have generated the set of rules that allow us to create the L-system we are left with a string of characters which represent that particular L-system. As with any grammar, there is a number of ways of interpreting the string that is generated by the L-systems rules. One method proposed by Przemyslaw Prusinkiewics is "to generate a string of symbols using an L-system, and to interpret this string as a sequence of commands which control a 'turtle'". [Prusinkiewicz, 1986]

A two dimensional L-system string may hold the following commands in the form of symbols.

- F: Move forward by a specified distance whilst drawing a line
- f: Move forward by a specified distance without drawing a line
- +: Rotate to the right specified angle.
- -: Rotate to the left by a specified angle.
- [: Save the current position and angle.
-]: Load a saved position and angle.

Similarly a three dimensional L-system string may hold the following commands in the form of symbols.

- F: Move forward by a specified distance whilst drawing a line
- f: Move forward by a specified distance without drawing a line
- +: Yaw to the right specified angle.
- -: Yaw to the left by a specified angle.
- /: Pitch up by specified angle.
- \: Pitch down by a specified angle.
- ^: Roll to the right specified angle.
- &: Roll to the left by a specified angle.
- [: Save the current position and angle.
-]: Load a saved position and angle.

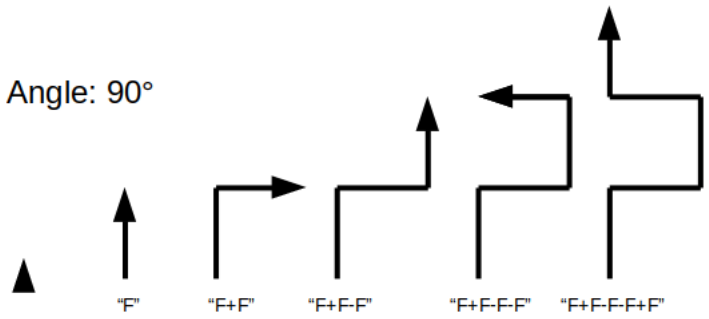


Figure 3.1: Diagram showing a turtle interpreting simple L-system string.

3.3 Branching Filaments

Explain how branching works and how it can be used for generating the L-systems

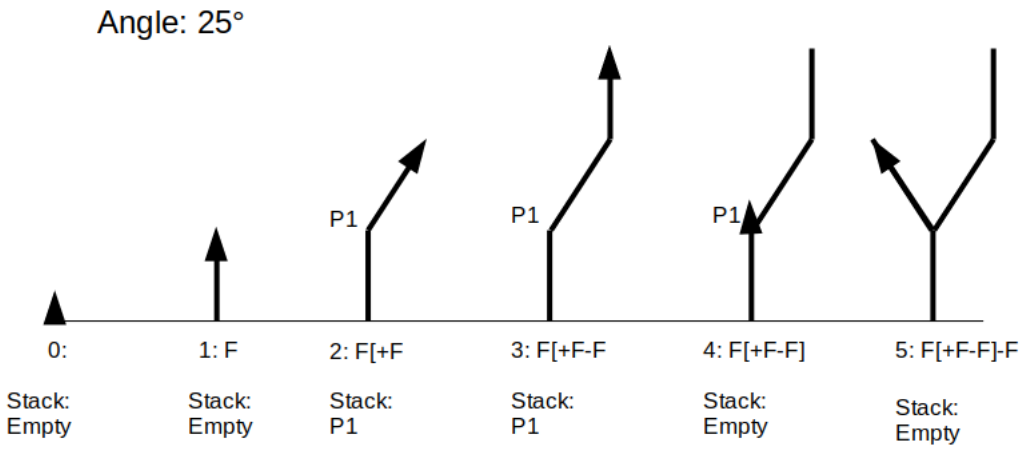


Figure 3.2: Diagram showing a turtle interpreting an L-system incorporating branching.

3.4 Basic 2D L-systems

There are a number of fractal geometry that have become well known particularly with regards to how they can seemingly imitate nature [Mandelbrot, 1982]. Particularly with the geometry such as the Koch snowflake which can be represented using the following L-system.

Koch Curve:
Alphabet: F
Constants: +, -
Axiom: F
Angle: 90°
Rules:
F → F+F–F+F

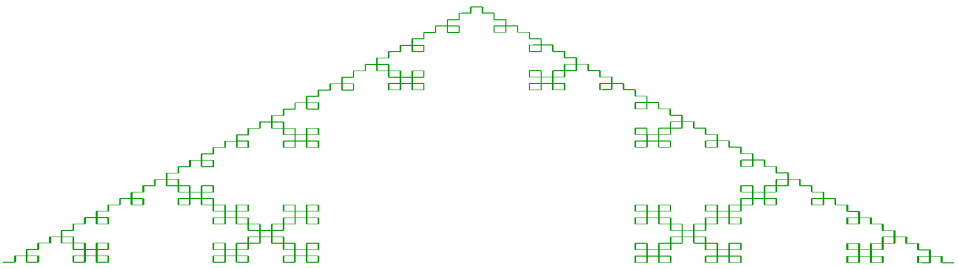


Figure 3.3: Koch Curve.

Sierpinski Triangle:
Alphabet: A, B
Constants: +, -
Axiom: A
Angle: 60°
Rules:
A → B-A-B
B → A+B+A

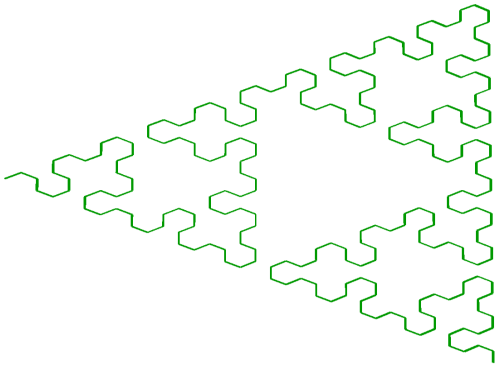


Figure 3.4: Sierpinski Triangle.

Dragon Curve:
Alphabet: F, X, Y
Constants: +, -
Axiom: FX
Angle: 90°
Rules:
X → X+YF+
Y → -FX-Y

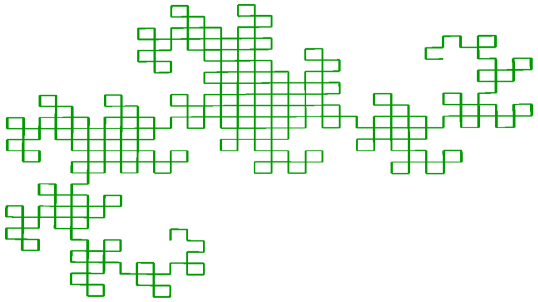


Figure 3.5: Dragon Curve.

Fractal Plant:
Alphabet: X, F
Constants: +, -, [,]
Axiom: X
Angle: 25°
Rules:
X → F-[[X]+X]+F[+FX]-X
F → FF

Figure 3.6: Fractal Plant.

Fractal Bush:
Alphabet: F
Constants: +, -, [,]
Axiom: F
Angle: 25°
Rules:
F → FF+[+F-F-F]-[-F+F+F]

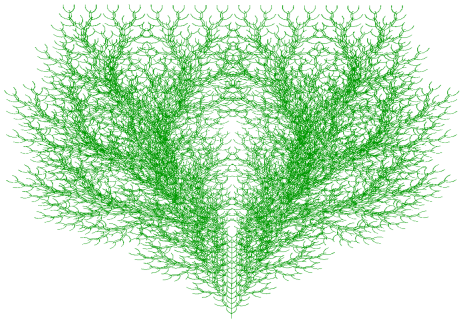


Figure 3.7: Fractal Bush.

3.5 The Use of L-systems in 3D applications

L-systems have been talked about and researched since its inception in 1968 by Aristid Lindenmayer. Over the years it's usefulness in modelling different types of plant life has been very clear, however its presence has been quite absent from any mainstream game engines for the most part, these engines relying either on digital artists skill to develop individual plants or on 3rd party software such as SpeedTree. These types of software use a multitude of different techniques however their methods are heavily rooted in Lindenmayer Systems.

Chapter 4

Implementation

Introduction to the implementation section

4.1 Language and environment

4.2 Open Graphics Library

Appendix A

Appendix

A.1 Appendix 1

A.2 Bibliography

Bibliography

[Mandelbrot, 1982] Mandelbrot, B. B. (1982). *The fractal geometry of nature*, volume 2. WH freeman New York.

[Prusinkiewicz, 1986] Prusinkiewicz, P. (1986). Graphical applications of l-systems. In *Proceedings of graphics interface*, volume 86, pages 247–253.

[Prusinkiewicz and Hanan, 2013] Prusinkiewicz, P. and Hanan, J. (2013). *Lindenmayer systems, fractals, and plants*, volume 79. Springer Science & Business Media.

[Worth and Stepney, 2005] Worth, P. and Stepney, S. (2005). Growing music: musical interpretations of l-systems. In *Workshops on Applications of Evolutionary Computation*, pages 545–550. Springer.