Forest Growth Simulation Using L-Systems

Submitted in partial fulfilment
of the requirements of the degree
Bachelor of Science (Honours)
of Rhodes University

Kim Randleff-Rasmussen

8 November 2004

Abstract

This project creates a realistic model of forest development and growth. It models the trees in diagram form using the wide-spread technique of L-Systems. The simulation of the forest growth takes into account a number of important issues in ecosystem development including the addition of new trees to the population, the removal of old trees and the changes that occur in existing trees due to interactions with others.

The interactions aim to model the competition that occurs between individual trees within a forest for the limited resources such as light, nutrients, water and CO_2. The interactions are simulated by checking for intersections between the circles which represent the space each tree takes up. When an intersection is discovered, the stronger tree dominates the weaker and the latter begins to die. Plants also die when they reach a maximum age.

Two particular tests were conducted on the system, one which checked for the realistic depiction of the results of competition and another which compared the output to that of another forest simulator (JABOWA III). The results of both tests were positive, prompting the conclusion that, though not realistic in minute detail, this simulator produces a realistic and thus viable depiction of forest growth.

Acknowledgements

I would like to thank my parents and my dog, without whom my sanity would long ago have given up the ghost.

I would like to thank my supervisors for their unwavering support and patience in tracking down rogue pointers and those annoying segmentation faults which plagued my coding.

Thanks must also go to my digsmates for their patience and uncomplaining sympathy as I bewailed the existence of trees in general.

Finally, thanks must, traditionally, go to Google and the marvellous results of its searches.

Contents

List of Figures

Chapter 1

Introduction

In all areas of computer graphics - from CGI movies to role-playing games - fields, mountainsides, jungles and forests are needed to extend the realism and thus complete the experience. For this reason, computer scientists have conducted much research into the modelling and rendering of realistic plants and trees. However, these natural environments are more than just a collection of trees and plants; rather, they involve a number of related yet individual organic organisms interacting with each other and thus affecting one another's shape and growth.

When two plants grow too near each other, they compete for the the limited resources in that limited space. These resources include sunlight, CO_2, water and various soil nutrients. In such a situation, the stronger plant will triumph and claim the bulk of the resources. The relative strength of a tree depends on inherited characteristics and thus differs from species to species. Some tree types prefer an abundance of sunlight and direct rain, others prefer shade, still others flourish better in circumstances of increased amounts of soil nutrients. Temperature and precipitation are also environmental factors which affect tree growth.

The aim of this project is to create a diagrammatic representation to simulate the development of a complex organic environment (such as a forest). The program will show trees in diagram form joining the current forest population, old ones dying and being removed from the population and the existing trees interacting amongt themselves. These interactions will result in the domination of one individual by another and the subsequent slowing in growth rate and death of the former.

In order to test the quantitative and qualitative realism and accuracy of the final model, the produced simulations will be compared to existing forest growth models.

1.1 Document Structure

Previous work in the fields of plant modelling in general, L-Systems in particular and procedural modelling is looked at in Chapter 2. This is followed by a discussion of the specific design approaches chosen, reasons for these choices and discussion of the inherent issues in Chapter 3. Chapter 4 looks at the particular implementation decisions that were taken in this project. This section also documents the results that have been achieved and compares these to the expected results, as well as discussing reasons for the variations.

Chapter 2

Previous Work

Below is a summary of the information available on L-Systems and other plant modelling techniques, procedural modelling and computer graphics software, with focus on those which concentrate on translation of L-Systems.

The overview is structured as follows: first a description of L-Systems is given followed by an investigation into work that has been conducted on, with and concerning L-Systems. Next, this review looks at other ways of modelling plants and organic structures followed by an exploration into procedural modelling. This section will focus on instances in which procedural modelling has been used to model geometrically complex scenes. Lastly, a review is conducted of current graphics software which provides interpretation of L-Systems.

2.1 Plant Modelling

2.1.1 L-Systems - Definitions and Explanations

In 1968, Hungarian biologist Aristid Lindenmayer developed a formalism for multicellular development [Lindenmayer 1968]. This formalism, subsequently named L-Systems (Lindenmayer Systems), captured the imagination of computer graphics artists who extended and improved the basic string rewriting language to form a model. This model is now widely used in computer graphics for the modelling of plants and other organic structures.

An L-System is a string replacement languages which uses defined production rules to supplant symbols within a string which can then be interpreted graphically by a turtle graphics program. These rewritings can be applied sequentially or in parallel, to every module or according to contextual information. There has even been the definition of a type of L-System which allows for both - the rewriting of two or

more symbols with a single production [Prusinkiewicz 1986]. The specific application is defined by the production rules.

The rewritten strings are then parsed by a LOGO-style graphics turtle which interprets each symbol in the string as an action - an addition or change to the graphical representation of the organism the string describes. A common basic L-System alphabet consists primarily of:

`F(s)`	Move forward a step of length `s` and draw a line segment from the original to the new position of the turtle.
`+(`θ`)`	Turn left by angle θ around the z axis.
`-(`θ`)`	Turn right by angle θ around the z axis.
`&(`θ`)`	Turn (roll) left by angle θ around the y axis.
`^(`θ`)`	Turn (roll) right by angle θ around the y axis.
`/(`θ`)`	Turn left (pitch up) by angle θ around the x axis.
`\(`θ`)`	Turn right (pitch down) by angle θ around the x axis.
`|`	Turn 180 degrees around the z axis.
`[`	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.

List taken from [Prusinkiewicz et al. 1995]

Other letters and symbols may also be used to delineate areas for string replacement.

Another important element of L-Systems is the production rules which specify what symbols are to be replaced by which strings at each iteration step. Due to this recursive process, L-Systems are a succinct language for describing the growth of organic material such as plant. The production rules have the following structure:

$$\texttt{lc < pred > rc : cond} \rightarrow \texttt{succ : prob}$$

where `lc` is the left-hand context, `rc` the right-hand context, `pred` the predecessor (the symbol to be replaced), `cond` a condition and `succ` the successor (the string to replace the predecessor). The rule is only run (i.e. the symbol replaced by the successor) if the condition evaluates to *true*. `prob` is a number which defines the probability of the rule being chosen. This is only found in stochastic L-Systems where more than one rule can exist for each symbol. Only the predecessor and the successor are compulsory.

For example, an L-System may comprise an axiom `F` and a replacement rule `F` → `F[<F][>F]F`. A first pass of the L-System will yield a new string `F[<F][>F]F` where the original `F` has been replaced. A second pass will yield `FF[<F][>F]F[<FF[<F][>F]F][>FF[<F][>F]F]F[<F][>F]F` where each `F` in the second string is replaced by the successor of the rule.

Since the original definition of L-Systems, many formal explanations and definitions have been published. Among the more prominent writers is Professor Przmyslaw Prusinkiewicz from the University of Calgary in Canada who, along with Lindenmayer, wrote what is considered to be the authoritative work on the subject [Prusinkiewicz and Lindenmayer 1990]. He has also written papers describing more specific types of L-Systems.

2.1.2 L-Systems - Extensions

Much has been published regarding different types of L-Systems which allow for certain changes in order to more accurately model reality. Rozenberg, for example, defined table L-Systems [Rozenberg 1973] which allow for the choosing of a set of developmental rules according to certain environmental factors. Environmentally sensitive L-Systems [Prusinkiewicz et al. 1994] were the next step, providing query symbols (`?E(parameters)`) which allow L-Systems to change as a result of local environmental conditions. Open L-Systems [Prusinkiewicz and Měch 1996] extend this functionality by including bilateral communication allowing the plant to return information to the environment. This extension allows the environment to be influenced by changes in the plant model, enabling the realistic modelling of such aspects as nutrient uptake and supply.

Other extensions provide functionality for variation of a single model to be applied to a number of individuals. Parametric L-Systems [Prusinkiewicz et al. 1995] include the facility to involve parameters in a standard L-System which allows certain aspects of the model (such as angle size and segment length) to be changed dynamically rather than statically defined. Differential L-Systems (or dL-Systems) [Prusinkiewicz et al. 1993] extend these parametric L-Systems by incorporating continuous time flow. This facility allows smooth animation of plant development rather than the rendering of discrete development steps, as is the standard practice.

Perhaps the most widely used L-Systems in terms of individual variation are stochastic L-Systems [Eichhorst and Savitch 1980]. These use multiple productions for each symbol, the specific rule being chosen according to assigned probabilities and a random number generator. This functionality allows a single L-System to yield a number individual plants with the same basic shape but with noticeable differences.

Many works exist which serve as extensions to Prusinkiewicz and Lindenmayer's original explanatory work. These compile all of the extensions made to L-Systems since 1990 into formal definitions and explanations including examples. A representative example of this synoptic work can be found in [Prusinkiewicz et al. 1996]. The most recent paper published discussing the current state of L-System modelling is [Prusinkiewicz 2004a].

Some L-System extensions which are unlikely to be found in such formal definitions are ones which are intended to model a specific element of plant development for a specific project. Some of these have been the includion sof a cut symbol (%) [Hanan 1992] to model self-pruning (see also [Prusinkiewicz et al. 1994]) and the development of erasing productions [Herman and Rozenberg 1975] to enable individual variation in a standard model.

Prusinkiewicz and Lane define a particular type of L-System which allows a set of productions to be applied to a set of strings (individual plants) instead of to a single string. These L-Systems, called multi-set L-Systems [Prusinkiewicz and Lane 2002], mean that a dynamic plant population can be maintained and operated on with a single model. This technique was developed in order to manage spatial distributions of a plant population.

Much work has been done on the inclusion of genetic algorithms [McCormack 1993], mutation [Mock 1998] and evolution [Rodkaew et al. 2002] into L-Systems. This amalgamation of techniques allows for such applications as virtual landscaping and adaptation simulations. These simulations can be further aided by a tefchnique known as extrusion in space-time [Hammel and Prusinkiewicz 1996] which extends a two-dimensional structure's development into a three-dimensional line or curve which represents the progress over time.

2.1.3 L-Systems - Graphical Interpretation

As mentioned, the most common way of translating L-Systems into two- or three-dimensional structures is through turtle graphics. A reasonable summary of this technique is to be found in [Prusinkiewicz et al. 1995], an author of which also helped to develop a software program called CPFG [James et al. 1993] which puts these turtle interpretations principles into practice.

[Prusinkiewicz et al. 1999] describes some of the programming principles behind the graphical interpretation of L-Systems. This paper is particularly useful for readers looking for explanations of L-System translation from formal language to programming language

2.1.4 Other Plant Modelling Techniques

L-Systems are not the only plant modelling technique available. Many other models abound, some of which focus on plant geometry as L-Systems do, and others which model trees based on behavioural aspects.

Some oft-cited models in the former class include the mathematical model developed by de Reffye *et al.* [de Reffye et al. 1988] which formalises known botanical rules. Other models focus more on specific geometric phenomena such as [Borchert and Honda 1984] which models the tendency of a tree to increase the vigour of growth after the loss of a branch. Meyerowitz [Meyerowitz 1994] used the specific plant *Aribidopsis thaliana* to model genes and mutation as they influence plant growth.

A comprehensive study of plant architectures and models which reflects the range available can be found in [Halle et al. 1978]. This book asserts that the thousands of plant species in the world can be modelled using a mere twenty-three models. These models take into account individual species architecture and genetic ”growth program” as well as competition for space and nutrients ina single forest. It goes one step further, too, and models the architecture of forests as separate entities.

Borchert and Slade [Borchert and Slade 1981] and Jenssen and Lindenmayer [Janssen and Lindenmayer 1987] also analysed the growth of a number of tree species in order to create models. [Prusinkiewicz 2004b] contains a summary of current plant modelling techniques including tecniques on modelling plant organs and tissues, genetics and physiological processes.

The second class of plant models focuses on how behaviour and the environment influence the manner in which a plant grows. [Prusinkiewicz et al. 2000], for example, models the effect of gravity on the branching behaviour of certain trees. Another paper by one of those authors looks at pruning - forced and self-imposed - and how this affects branch growth [Prusinkiewicz et al. 1994]. Yet another paper, [Prusinkiewicz et al. 2001], uses positional information to model the growth and survival of a plant. This model uses such techniques as self-thinning and competition for space to model how tree growth is affected by environmental factors.

A model designed by Lück, Lück and Bakkali [Lück et al. 1990] suggests the use of a growth potential or ”vigor” value to be assigned to branches. This value dictates which of a number of competing branches will triumph. Honda *et al.* also focuses on branch interaction between trees [Honda et al. 1981]. The value they use to determine the ”winning” branch is flow rate which dictates the rate of bifurcation. Sorrensen-Cothern, Ford and Sprugel divide a single plant into a number of modules and model the interaction between these rather than the interaction between entire plants [Sorrensen-Cothern

et al. 1993].

2.1.5 Plant Modelling in Practice

For completeness' sake, this section contains mention of a number of papers which describe the use of plant models (both L-Systems and other) in practice. [Mackrill 2003] used L-Systems to model realistic landscapes including trees, grass and terrain. The focus here is on the ability to render in real-time. [Prusinkiewicz et al. 1988] describes the use of L-Systems to model not only branches but leaves and flowers as well. There is a focus here on the ability to show the animation of plant development, hence the choice of L-Systems as the modelling technique.

[Weber and Penn 1995] describes an entirely new plant modelling language which allows for realistic depiction of geometry without strict adherence to botanical principles. They also present a technique for varying detail level which eases the rendering of large landscape shots of complex forests.

2.2 Procedural Modelling

Procedural modelling is a well-used computer graphics technique designed to decrease the computational power required to render a geometrically complex scene. It does so by improving on two areas of scene rendering. Firstly, objects that are not within the view spectrum of the camera are not rendered [Foley 1996]. Secondly, elements of the model are rendered according to the level of detail required. For example, a knot on a tree trunk would only be visible from a metre or less away from the tree and not from an aerial shot.

A number of papers and books exist which explain the concepts behind procedural modelling. Some of these are [Jeschke et al. 2003] and [Macri and Pallister 2004]. Amburn *et al.* [Amburn et al. 1986] suggest a number of ways in which traditional procedural models might be improved. One of these techniques is communication between models which may affect one another that models may adjust their behaviour according to messages from another model. The second improvement is subdivision of labour and the sharing of some subdivided activities between procedures.

Procedural modelling has long been in use in programs which model geometrically complex scenes such as forests, cities and crowds. A number of papers present these programs, describing the use of procedural models to ease the rendering of complex scenes which contain many instances of geometrically similar constructs (such as trees and buildings). Some of the more prominent papers which model large cities are [Birch et al. 2003], [Greuter et al. 2003], [Parish and Muller 2001] and [Wonka et al. 2003]. In

terms of forests or landscape generation, some prominent projects are described in [Guerraz et al. 2003] and [Deussen et al. 1998].

Many such papers describe not only the use of procedural models but also the interactions between the procedural models and other aspects of scene modelling. [Deussen et al. 1998] and [Greuter et al. 2003], for example, describe entire systems for scene modelling - drawing all aspects of modelling and rendering together.

2.3 Software Products

Much research and development has been done into software programs that translate L-Systems into three-dimensional images. Other software products have also been developed which model plants and organic structures without the explicit use of L-Systems.

One of the first L-System translation systems was developed in 1970 [Baker and Herman 1970] in order to simulate organic development for hypothesis testing. Prusinkiewicz and his students and colleagues have developed a number of software products since that time. CPFG [Prusinkiewicz 1993] takes a parameterised procedural model of a plant and outputs a geometric model. This product is used in the modelling system L-Studio [Prusinkiewicz 1998] which provides a graphical user interface for the modelling of plants via L-Systems with comprehensive options for all aspects of L-Systems.

Laurens Lapré, a student of Professor Prusinkiewicz, developed a system called Lparser [Lapré 1993] which is available in open source and is a good place to start for beginners in L-System translation due to the readable, logical code. Another product developed by a student, though not one taught directly by Prusinkiewicz, is L-Arbor developed by Marco Grubert [Grubert 1999]. This program offers simple animation and L-System modelling facilities and is an easy way to learn basic L-Systems.

The Institute of Forest Biometry and Informatics at the University of Guttingen is a special interest plant modelling group comprised of experts in a number of fields including forest science, biology and mathematics. Winfried Kurth is the computer graphics expert and he developed a modelling program called GROGRA (GROwth GRAmmar) [Kurth 1994].

GRAMPS [O'Donnell and Olson 1981] is a graphics language interpreter developed to model molecular structures and to animate their development. It is used in conjunction with an interactive vector display list processor and is focussed on the provision of real-time scene manipulation facilities. A nice feature of this program is the facility to dynamically add new graphics manipulation commands through

the use of macros.

JABOWA [Botkin et al. 1999] is a forest growth simulator developed by the Centre for the Study of the Environment. This simulator does not use L-Systems to model the trees but merely scaled images of the different species required. However, the simulator does model, in detail, many environmental and local factors which affect the growth rate and survival chances of the individual trees.

2.4 Summary

L-Systems were chosen over any of the other available modelling techniques due to their pervasiveness and, therefore, the availability of support. Another reason is their effectiveness in accurately modelling plants which, in conjunction with their pure simplicity, makes them a viable option for the less experienced graphics practitioner.

Procedural modelling was chosen because, again, it is a widely documented technique. It also allows for objects to be rendered only if they can be seen by the camera, an attribute which makes the creation of complex scenes less computationally intensive.

Both of these techniques are well-documented and widely used in the modelling and creation of geometrically complex scenes. This section is intended to be a representative study of the wide range of literature available on both topics. For completeness' sake, other plant modelling techniques are explored as well as instances of the use of procedural models in practice to model complex scenes. These scenes include forests, landscapes and cityscapes. A brief look at the software products available for L-System creation and rendering completes this review.

Chapter 3

Design

The aim here is to create a diagrammatic representation of the interactions between a number of organic structures in a close environment. These interactions are modelled using a domination algorithm which stunts the growth of plants that encroach on the space of others. This same algorithm allows for the dynamic addition to and removal from the population of individual organisms. The plants themselves are modelled using the wide-spread technique of L-Systems which produce dynamic branching structures - perfect diagrams of plants for this purpose and ones that can be easily translated to realistic images should the need arise.

This chapter discusses the design considerations inherent in the creation of this diagrammatic representation. First, the technique of procedural modelling is discussed and reasons given for its choice. Secondly, the L-Systems used in this project are discussed in terms of their syntax and translation within the program. The algorithm used to model the forest growth is discussed in Section 3.3 followed by a discussion of the techniques used for mesh subdivision.

3.1 Procedural Models

The basis of procedural modelling is to have all of the complex geometry needed in a scene generated within a procedure. In this procedural model, the entire scene is modelled by a single procedure (`forest`). Within this procedure, the forest growth simulation is performed. However, the actual generation of the individual trees is delegated to another procedure (`lsystem`).

This procedure parses the L-System files (see Section 3.2) in order to translate them to a graphical image. However, the translation that occurs in this procedure is the changing of orientation according to the symbols in the string. The generation of the actual geometric segments is delegated to another procedure - `segment`.

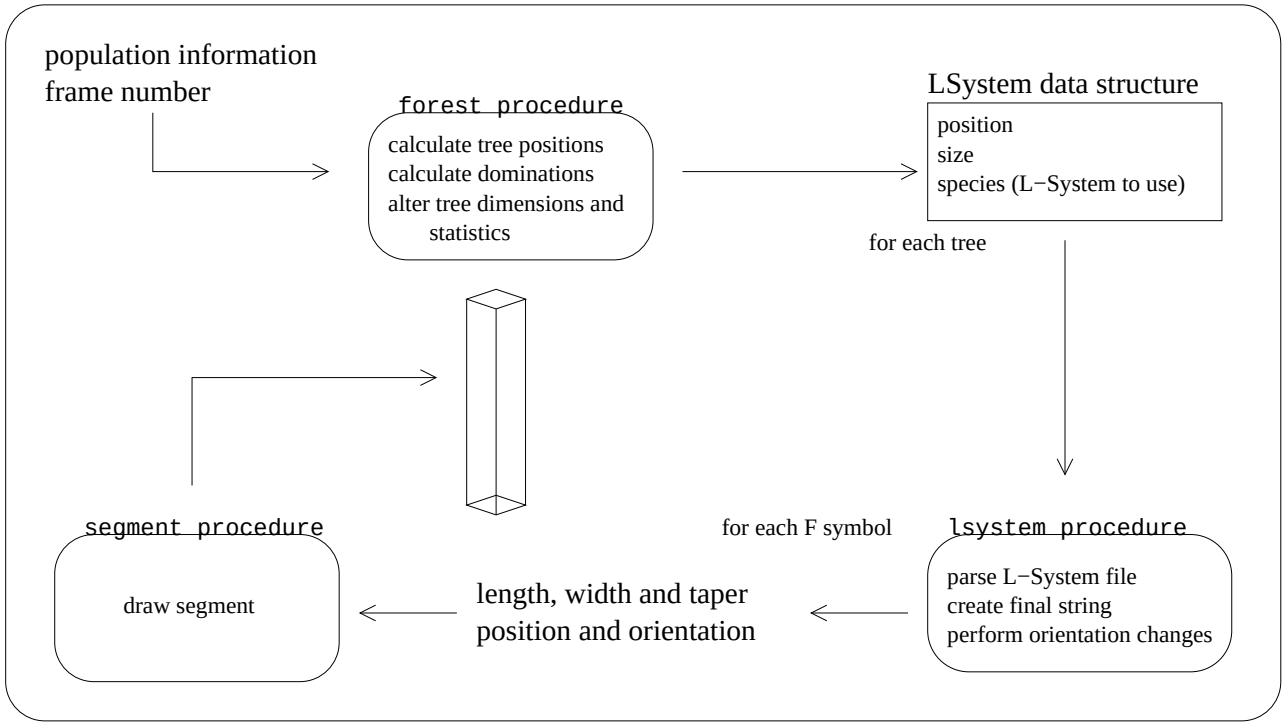

Figure 3.1: A diagram showing the working of the procedural model

This technique means that large scenes can be developed quickly and efficiently as the geometry generation is only tackled after the other calculations. Figure 3.1 is a diagram of the logical progression of the procedural model.

Each procedure is supplied with the Transformation matrix from the calling procedure. This means that the geometric mesh produced by the procedure is produced according to its local coordinate system which is then transposed to the correct position in the global coordinate system. Along with the matrix, the procedure is also supplied with a long integer key which can be used to seed a pseudo-random number sequence. It is useful to use the same number to seed the sequence so that the exact same forest can be generated provided the key is known.

3.2 Plants

3.2.1 L-Systems

An L-System is read from a `.sys` file which has the following structure:

`LSYSTEM`	file header
`r`	an integer denoting the number of replacement rules
`A`	axiom (or L-System start string)
`R`	replacement rules with the structure `A R p` (`A` is the symbol to replace, `R` the string to replace it with and `p` is a double value which signifies the probability of the rule being fired)

The L-System strings within this file make use of the alphabet shown in Table 3.1. These L-System files are parsed and the start string translated from a string into a defined L-System string data structure (called `alstr`). This structure is designed to allow easy access to any of the modules within the string and their parameters. An LSystem data structure stores this `alstr` as well as the replacement rules for the L-System, including their probabilities. Below can be seen a pseudo-code definition of the `alstr` data structure as well as the LSystem class.

```
type alstr

    char [] modules
    double [][] parameters
    int number_of_modules
```

`F(l,w,t)`	Draw a tapered rectangular segment of length `l`, base width `w` and tapered tip width `t`.
`<(`θ`)`	Turn right by angle θ around the z axis.
`>(`θ`)`	Turn left by angle θ around the z axis.
`&(`θ`)`	Turn (roll) right by angle θ around the y axis.
`@(`θ`)`	Turn (roll) left by angle θ around the y axis.
`%(`θ`)`	Turn right (pitch down) by angle θ around the x axis.
`#(`θ`)`	Turn left (pitch up) by angle θ around the x axis.
`|`	Turn 180 degrees around the z axis.
`[..]`	Push the current state of the turtle (position, orientation and drawing attributes) onto a pushdown stack.Branch by pushing (and later popping) the current Transformation matrix to (from) a pushdown stack.

Table 3.1: The L-System alphabet used in this project

`class LSystem`

`alstr string`

`int number_of_replacement_rules`

`char [] symbols` → the symbols which have an associated replacement rule

`alstr [] replacements` → the strings which replace the above symbols

`int [] number_of_replacements` → how many replacement rules are associated with each symbol

`double [] probs` → the probabilities for each replacement rule

`LSystem (int species, int iteration)` → constructor

`int get_size ()` → returns the length of the `alstr` (number of modules)

`char get_module (int position)`

`double [] get_parameters (int position)`

3.2.2 Plant Population

As can be seen in Figure 3.1 above, the `forest` procedure takes as an input population information. This information includes the following:

- Total number of trees in population
- Number of trees that have existed since (and including) the start of the simulation
- Certain information regarding each tree:

- x-coordinate
- z-coordinate
- Radius (for simulation purposes)
- Species (for L-System file selection)
- Vigour
- Age (calculated as a function of the radius)
- Health (including whether or not the plant has reached maximum size and whether or not it has been dominated)

3.3 Forest Simulation

There are a huge number of factors which influence the growth of a forest. This project aims to simplify the complex and varied reality of forest growth into a general model. [Botkin 1993] discusses the importance of finding ”generalizations that are the simplest consistent with understanding”. That is to say that one need not model every detail of a complex system but rather attempt to ”determine the simplest conceptual basis consistent with the observation” of reality.

3.3.1 Plant Placement

[Prusinkiewicz and Lane 2002] introduces a number of methods by which the placement of plants in a forest can be determined. These methods can be considered to be local-to-global or global-to-local. The latter category contains methods which place individuals according to pre-defined and constantly updated density functions. As each plant is placed, the probability function is updated according to this new placement and the placement of the next plant will be affected.

Local-to-global plant placement methods simulate ecosystem interaction by focusing on each individual. Multi-set L-Systems are an example of this type of placement methodology. These L-Systems apply a set of production rules to a set of L-Systems strings (which symbolises plant population). This allows for the dynamic addition and removal of individual plants and the dynamic upkeep of the population as a whole.

[Deussen et al. 1998] mentions two methods for placing plants within a modelled forest. The first of these involves inputting a density map to which an error diffusion algorithm is then applied. The specific algorithm implemented in this paper is one introduced by Floyd and Steinberg. The second method

places plants randomly and then iteratively grows or kills individuals based on domination.

In reality, trees produce seeds which are dispersed by various means. A percentage of these seeds then germinate and a percentage of the germinated seeds survive to grow into saplings. This project does not go into such detail in the modelling of placing plants due to the hugely complex nature of the task. Rather, the second method mentioned in [Deussen et al. 1998] is employed and a random number of new plants are randomly placed in the scene.

3.3.2 Competition and Domination

In reality, the growth rates of trees in a forest are affected by a number of environmental factors including the availability of sunlight, water and nutrients. These availabilities are influenced by the surrounding trees - when two trees grow too close together, they must compete for the limited amounts of light, water and nutrients in their shared space. Ultimately, one tree will triumph over another.

This simulation does not model the competition between trees for every element necessary for their survival. Rather, the competition is simplified into a domination algorithm. For the purposes of the simulation, plants are viewed as circles on the viewing plane. These circles have dynamic radii which change according to the growth rate of the plant in question which is calculated as a function of the plant's health. When two circles intersect, the plants are said to have come into competition with one another. The vigour values of both individuals are compared and the stronger plant dominates the weaker. This means that the weaker plant's health is decreased, resulting either in stunted growth or even death. Thus, the circles can be said to represent the ”footprint” of the plant in the forest.

The radii of the circles also form the basis for the calculation of the age of the plant. For this project, the age is an integer value ranging from 0 to the maximum size of the plant's species. This age value is used to determine how many iterations of the L-System rules must be completed to yield the final L-System string. An age value of 0 does not necessarily result in an empty mesh. Rather the axiom of the L-System is translated and, should it contain any `F` symbols, drawn accordingly.

3.4 Summary

This chapter discusses the various design decisions that were taken in creating this simulation. The workings of the procedural models are discussed as are the various ways in which L-Systems are stored and read. In terms of the forest simulation, plants are randomly placed about the scene and grow according to defined growth rates. These growth rates are affected by the competition a plant receives from other

plants in the scene which is modelled by checking for intersections between circles which represent the plants.

Chapter 4

Results

4.1 Plants

L-Systems are translated to images according to the alphabet in Table 3.1. Figure 4.1 shows an example of an L-System image where the originating string is (ignore '-' symbols in this text):

F[>F][<F]F[>F[>F][<F]F][<F[>F][<F]F]F[>F][<F]F[>F[>F][<F]F[>F[>F][-
<F]F][<F[>F][<F]F]F[>F][<F]F][<F[>F][<F]F[>F[>F][<F]F][<F[>F][<F]F-
]F[>F][<F]F]F[>F][<F]F[>F[>F][<F]F][<F[>F][<F]F]F[>F][<F]F.

Generalised mesh subdivision techniques are introduced in [Maierhofer 2002] and allow for the smoothing of meshes using a combination of a number of common subdivision methods (such as Catmull-Clark, Doo-Sabin and $\sqrt{3}$-Subdivision). Implemented in this project are some of these techniqes in order to create more realistic, smooth trees. Figure 4.2 shows a section of an L-System tree before application of these techniques and after smoothing.

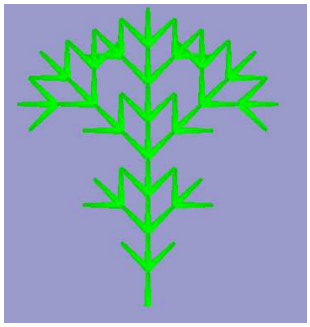

Figure 4.1: An example of an L-System tree

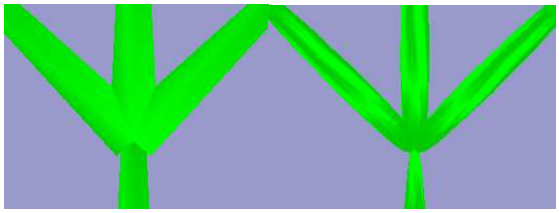

Figure 4.2: Two versions of the same joint showing the difference between the generated cube (left) and the same cube after smoothing (right)

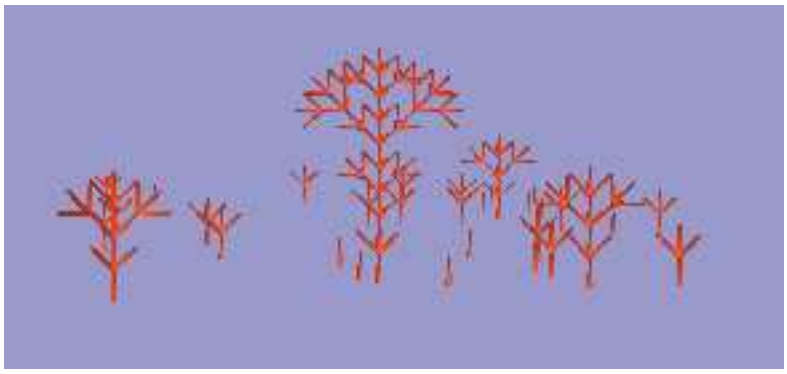

Figure 4.3: Plant distribution using random numbers

4.2 Forest Simulation

4.2.1 Plant Placement

As has been mentioned in Section 3.3.1, the plants in this scene are randomly placed. This is achieved by generating a random number in the range -100 to 200 for the x-coordinate and another in the range 100 to 900 for the z-coordinate. These ranges represent the area of the forest in coordinate terms.

[Prusinkiewicz and Lane 2002] suggests that random placement of the plants within the scene leads to a uniform distribution which is unrealistic. They apply a clustering algorithm in order to counter the uniformity of the plant distrubution. No clustering algorithms are employed in this project; however, the area used is small enough that no uniform distribution can be discerned. Figure 4.3 shows an example of the distribution achieved by random placement.

4.2.2 Competition and Domination

A simple test, suggested by [Botkin 1993], can be conducted to test the realism of the simulator's model of competition. This test involves allowing a large, sturdy tree to exist in the area and remvoing it after a number of cycles. If the simulation models competition realistically, the other trees in the area should grow at extremely slow rates, not grow at all or even die while the large, sturdy tree survives. Once the larger tree is removed, the smaller, dominated trees and others that join the population should grow at increased rates.

If, as a measure of forest growth, the number of plants present per year/frame is taken, then, in general, the number of plants should be less while the large, sturdy tree exists. Once the large, sturdy tree is removed, the number of trees should increase until it is approximately equal to the number of trees present without the large, sturdy tree.

Figure 4.4 shows the numbers generated whilst performing the above test on the simulator. As can be seen, the number of trees present in the forest is much less than in the "normal" simulator while the large, sturdy tree exists. At frame 100 (highlighted by a thick black line in Figure 4.4), the large, sturdy tree is removed and forest growth increases until the number of trees present is approximately equal to that of an ordinary simulation. Growth then stabilises to keep the number of trees approximately level.

4.2.3 Full Simulation

As mentioned in Section 3.3, the plants are represented by their footprints in the forest growth simulator. Figure 4.5 shows a moment in the simulation showing the various plant footprints. Here, the footprints are shown as squares rather than the aforementioned circles due to the ease of representation. When the footprints overlap, the plants' vigour values are compared and one plant is dominated - its growth rate is decreased.

4.2.3.1 Comparison

In order to test the quantitative realism of the produced simulator, the output of a detailed forest simulator, JABOWA III [Botkin et al. 1999] was compared to the output of the forest simulator created. Figure 4.6 shows how the number of plants generated by the created simulator was similar to the number of trees generated by the detailed simulator (JABOWA III). The larger discrepancies that can be seen (after frame 225 or so) can be attributed to the fact that the JABOWA simulator took into account environmental differences which would decrease the growth rate of trees after a certain number of years. This phenomenon

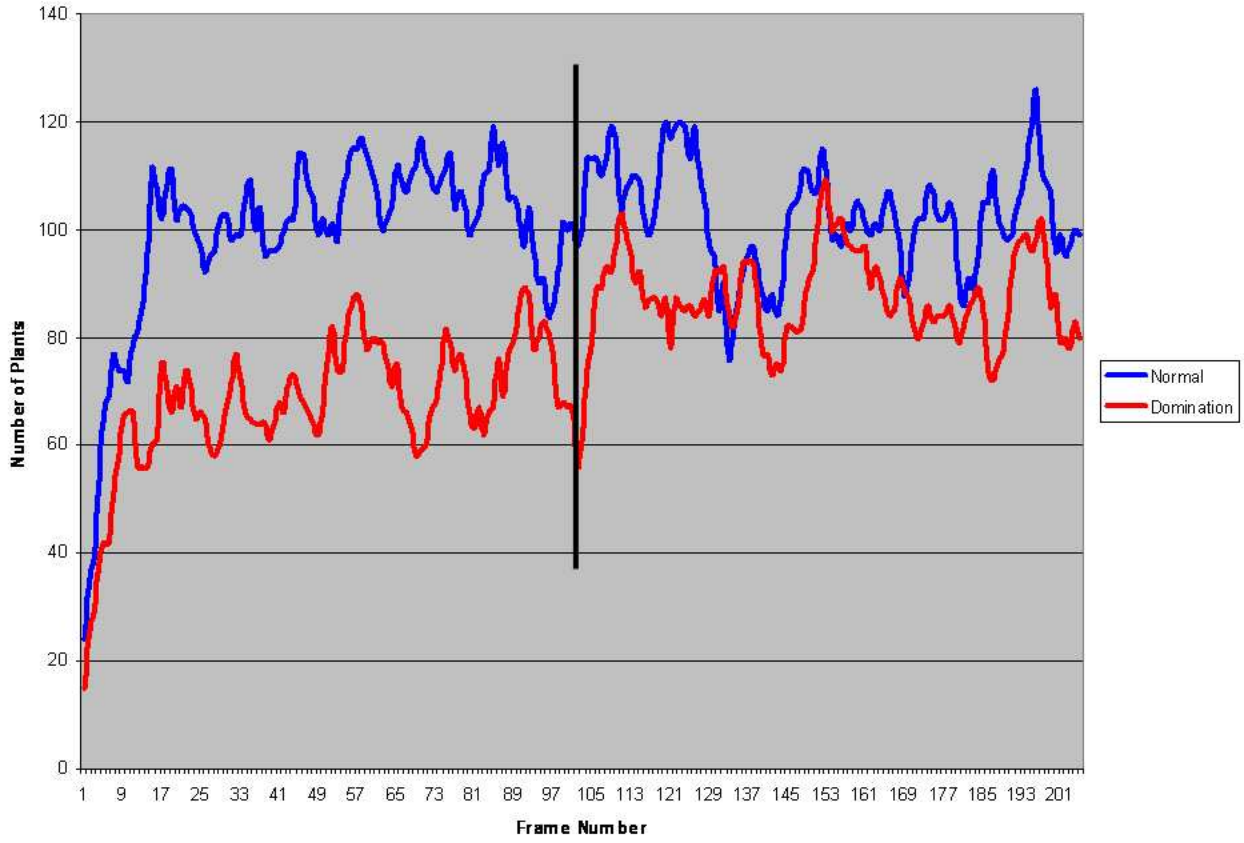

Figure 4.4: A graph showing the progression of an ordinary simulation versus a domination test.

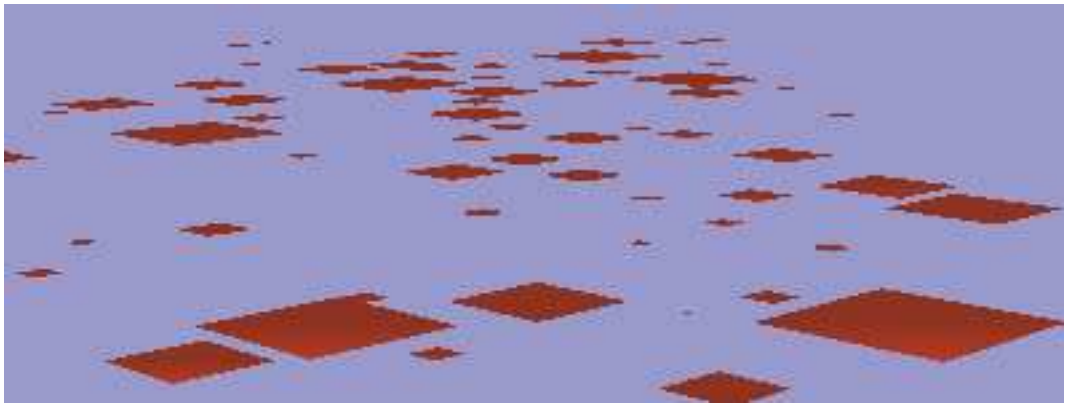

Figure 4.5: A snapshot of the forest simulation. The squares pictured represent the footprints of the plants in the forest.

was not modelled by the simulator in this project.

4.3 Summary

This section reviews the results achieved by the forest simulator. Images are provided to illustrate the method employed in the simulation and the final animation. Graphs are also provided to illustrate the results of a number of tests that were conducted on the simulator. These tests include a simple domination test to establish whether forest growth is modelled realistically and a more complex test involving comparison with another forest simulator.

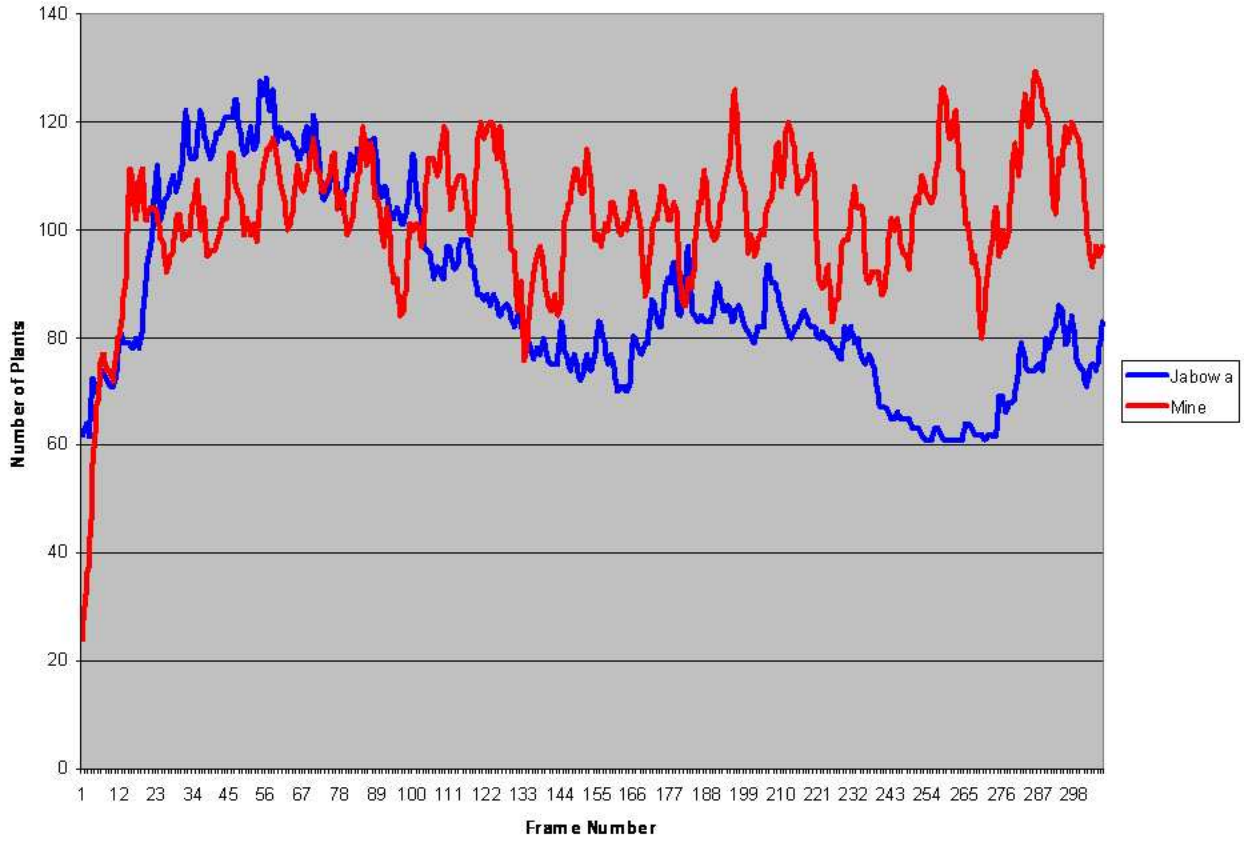

Figure 4.6: A graph showing the progression of a JABOWA III simulation versus this forest simulator.

Chapter 5

Conclusion

This project creates a diagrammatic representation to simulate the development and growth of a forest. The program shows the addition of new trees to the forest population and the removal of old ones. It also models the interactions which occur between the trees, resulting in the domination of one individual by another and the subsequent change in growth rate and ultimate death of the weaker specimen.

This project simplifies the complex nature of natural interactions and thus does not model forest growth in minute detail. Rather, the significant aspects of forest growth are investigated and simulated, assuming that the smaller details will affect the growth only in very small ways. This approach has meant that the simulator produced shows a realistically growing forest but one that fails more rigorous testing.

The testing conducted was twofold: first, a simple domination test was conducted which tests the realism of the competition modelling. The outcome of this test was successful, proving that the simulator models interactions between trees and the subsequent effects on growth rate realistically.

The second test was more complex and involved comparing the output of the simulator with that of another, established forest simulator. Again, the test was successful, illustrating the viability of the simulator as a depiction of reality. Certain differences were noted, however, and the negative results of the broader model could be plainly seen during a longer simulation.

The simulator produced is a viable one which depicts realistic forest growth to a degree that is acceptable for computer graphics needs. While much detail could be added to the simulator in order to correct some of the more general modelling aspects, the simulator as it stands could be employed in a computer game or movie. The simulator, however, is only diagrammatic and thus more work would have to be done on the realistic depiction of the individual trees.

References

[Amburn et al. 1986] Amburn, P., E. Grant and T. Whitted (1986). Managing geometric complexity with enhanced procedural models. In *SIGGRAPH 1986 Conference Proceedings* (1986), ACM Press, pp. 189–195.

[Baker and Herman 1970] Baker, R. and G. Herman (1970). Celia - a cellular linear iterative array simulator. In *Proceedings of the Fourth Conference on Applications of Simulation* (1970), pp. 64–73.

[Birch et al. 2003] Birch, P., S. Browne, V. Jennings, A. Day and D. Arnold (2003). Rapid procedural modelling of architectural structures. In *SIGGRAPH 2003 Conference Proceedings* (2003), ACM Press.

[Borchert and Honda 1984] Borchert, R. and H. Honda (1984). Control of development in the bifurcating branch system of tabebuia rosea: A computer simulation. *Botanical Gazette 145*, 2 (1984), pp. 184–195.

[Borchert and Slade 1981] Borchert, R. and N. Slade (1981). Bifurcation ratios and the adaptive geometry of trees. *Botanical Gazette 142*, 3 (1981), pp. 394–401.

[Botkin 1993] Botkin, D. (1993). *Forest Dynamics: An Ecological Model*. Oxford University Press, UK, 1993.

[Botkin et al. 1999] Botkin, D., D. Woodby and J. Bergengren (1999). Jabowa: A model of forest growth. Tech. rep., The Center for the Study of the Environment, New York, 1999.

[Deussen et al. 1998] Deussen, O., P. Hanrahan, B. Lintermann, R. Měch, M. M. Pharr and P. Prusinkiewicz (1998). Realistic modeling and rendering of plant ecosystems. In *SIGGRAPH 1998 Conference Proceedings* (1998), ACM Press.

[Eichhorst and Savitch 1980] Eichhorst, W. and J. Savitch (1980). Growth functions of stochastic lindenmayer systems. *Information and Control 45*, 3 (1980), pp. 217–228.

[Foley 1996] Foley, D. (1996). *Computer Graphics Principles and Practice Second Edition*. Addison-Wesley, Reading, Massachusetts, 1996.

[Greuter et al. 2003] Greuter, S., J. Parker, N. Stewart and G. Leach (2003). Undiscovered worlds. In *melbourneDAC - 5th International Digital Arts and Culture Conference* (2003).

[Grubert 1999] Grubert, M. (1999). L-arbor. Master's thesis, Technical University of Berlin, Germany, 1999.

[Guerraz et al. 2003] Guerraz, S., F. Perbet, D. Raulo, F. Faure and M. Cani (2003). A procedural approach to animate interactive natural sceneries. In *CASA03* (2003).

[Halle et al. 1978] Halle, F., R. Oldeman and P. Tomlinson (1978). *Tropical trees and forests*. Springer-Verlag, Heidelberg, New York, 1978.

[Hammel and Prusinkiewicz 1996] Hammel, M. and P. Prusinkiewicz (1996). Visualization of developmental processes by extrusion in space time. In *Proceeding of Graphics Interface '96* (1996), pp. 246–258.

[Hanan 1992] Hanan, J. (1992). *Parametric L-systems and their application to the modelling and visualization of plants*. PhD thesis, University of Regina, Canada, 1992.

[Herman and Rozenberg 1975] Herman, G. and G. Rozenberg (1975). *Developmental systems and languages*. North-Holland, Amsterdam, 1975.

[Honda et al. 1981] Honda, H., P. Tomlinson and J. Fisher (1981). Computer simulation of branch interaction and regulation by unequal flow rates in botanical trees. *American Journal of Botany*, 68 (1981), pp. 569–585.

[James et al. 1993] James, M., J. Hanan and P. Prusinkiewicz (1993). *CPFG version 2.0 User's Manual*. PhD thesis, Department of Computer Science, University of Calgary, Canada, 1993.

[Janssen and Lindenmayer 1987] Janssen, J. and A. Lindenmayer (1987). Models for the control of branch positions and flowering sequences of capitula in mycelis muralis (l.) dumont (compositae). *New Phytologist 105*, 2 (1987), pp. 191–220.

[Jeschke et al. 2003] Jeschke, S., H. Birkholz and H. Schumann (2003). A procedural model for interactive animation of breaking ocean waves. In *WSCG 2003* (2003).

[Kurth 1994] Kurth, W. (1994). *GROGRA*. PhD thesis, Institute of Forest Biometry and Informatics, Faculty of Forest Sciences and Forest Ecology at the University of Göttingen, 1994.

[Lapré 1993] Lapré, L. (1993). Lparser.

[Lindenmayer 1968] Lindenmayer, A. (1968). Mathematical models for cellular interaction in development. *Journal of Theoretical Biology* (1968).

[Lück et al. 1990] Lück, J., H. Lück and M. Bakkali (1990). A comprehensive model for acrotonic, mesotonic, and basitonic branching in plants. *Acta Biotheoretica 38* (1990), pp. 257–288.

[Mackrill 2003] Mackrill, B. (2003). Real-time rendering of realistic landscape, 2003, <http://www.dcs.shef.ac.uk/teaching/eproj/ug2003/pdf/u0bm.pdf>.

[Macri and Pallister 2004] Macri, D. and K. Pallister (2004). Procedural 3d content generation. Tech. Rep. 20182, Intel Developer Service, 2004.

[Maierhofer 2002] Maierhofer, S. (2002). *Rule-Based Mesh Growing and Generalised Subdivision Meshes*. PhD thesis, Vienna University of Technology, Austria, 2002.

[McCormack 1993] McCormack, J. (1993). Interactive evolution of l-system grammars for computer graphics modelling. In *Complex Systems: from Biology to Computation*, D. Green and T. Bossomaier, Eds. ISO Press, Amsterdam, 1993.

[Meyerowitz 1994] Meyerowitz, E. (1994). The genetics of flower development. *Scientific American 271*, 5 (1994), pp. 56–65.

[Mock 1998] Mock, K. (1998). Wildwood. In *International Conference on Evolutionary Computing (ICEC '98)* (1998).

[O'Donnell and Olson 1981] O'Donnell, T. and A. Olson (1981). Gramps: A graphics language interpreter for real-time, interactive, three-dimensional picture editing and animation. *Computer Graphics 15*, 3 (1981), pp. 133–142.

[Parish and Muller 2001] Parish, Y. and P. Muller (2001). Procedural modeling of cities. In *SIGGRAPH 2001 Conference Proceedings* (2001), ACM Press, pp. 301–308.

[Prusinkiewicz 1986] Prusinkiewicz, P. (1986). Graphical applications of l-systems. In *Proceedings of Graphics Interface '86 - Vision Interface '86* (1986), pp. 247–253.

[Prusinkiewicz 1993] Prusinkiewicz, P. (1993). Cpfg version 2.0. Tech. rep., Virtual Laboratory, University of Calgary, 1993.

[Prusinkiewicz 1998] Prusinkiewicz, P. (1998). L-studio. Tech. rep., University of Calgary Computer Science, 1998.

[Prusinkiewicz 2004a] Prusinkiewicz, P. (2004a). Art and science for life: Designing and growing virtual plants with l-systems. *Acta Horticulturae 630* (2004), pp. 15–28.

[Prusinkiewicz 2004b] Prusinkiewicz, P. (2004b). Modeling plant growth and development. *Current Opinion in Plant Biology 7*, 1 (2004), pp. 79–83.

[Prusinkiewicz et al. 1999] Prusinkiewicz, P., J. Hanan and R. Měch (1999). An l-system-based plant modeling language. In *Applications of graph transformations with industrial relevance*, M. Nagl, A. Schuerr, and M. Muench, Eds. Springer, Berlin, 1999, pp. 395–410.

[Prusinkiewicz et al. 2000] Prusinkiewicz, P., C. Jirasek and B. Moulia (2000). Integrating bioměchanics into developmental plant models expressed using l-systems. In *Plant BioMěchanics 2000*, H. Spatz and T. Speck, Eds. Georg Thieme Verlag, Stuttgart, 2000.

[Prusinkiewicz and Lane 2002] Prusinkiewicz, P. and B. Lane (2002). Generating spatial distributions for multilevel models of plant communities. In *Proceedings of Graphics Interface* (2002), pp. 69–80.

[Prusinkiewicz et al. 2001] Prusinkiewicz, P., B. Lane, L. Muendermann and R. Karwowski (2001). The use of positional information in the modeling of plants. In *SIGGRAPH 2001 Conference Proceedings* (2001), ACM Press, pp. 289–300.

[Prusinkiewicz and Lindenmayer 1990] Prusinkiewicz, P. and A. Lindenmayer (1990). *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, 1990.

[Prusinkiewicz et al. 1988] Prusinkiewicz, P., A. Lindenmayer and J. Hanan (1988). Developmental models of herbaceous plants. *Computer Graphics 22*, 4 (1988), pp. 141–150.

[Prusinkiewicz et al. 1993] Prusinkiewicz, P., E. Mjolsness and M. Hammel (1993). Animation of plant development. In *SIGGRAPH 1993 Conference Proceedings* (1993), ACM Press.

[Prusinkiewicz and Měch 1996] Prusinkiewicz, P. and R. Měch (1996). Visual models of plants interacting with their environment. In *SIGGRAPH 1996 Conference Proceedings* (1996), ACM Press, pp. 397–410.

[Prusinkiewicz et al. 1995] Prusinkiewicz, P., R. Měch and M. Hammel (1995). The artificial life of plants. In *Artificial Life for Graphics, Animation, and Virtual Reality, v7 of SIGGRAPH 1995 Course Notes* (1995), ACM Press, pp. 1.1–1.38.

[Prusinkiewicz et al. 1996] Prusinkiewicz, P., R. Měch, M. Hammel and J. Hanan (1996). Visual models of plant development. In *Handbook of formal languages*, G. Rozenberg and A. Salomaa, Eds. Springer-Verlag, Berlin, 1996.

[Prusinkiewicz et al. 1994] Prusinkiewicz, P., R. Měch and M. James (1994). Synthetic topiary. In *SIGGRAPH 1994 Conference Proceedings* (1994), ACM Press.

[de Reffye et al. 1988] Reffye, P. de, C. Edelin, J. Francon, M. Jaeger and C. Peuch (1988). Plant models faithful to botanical structure and development. *Computer Graphics 22*, 4 (1988), pp. 151–158.

[Rodkaew et al. 2002] Rodkaew, Y., C. Lursinsap, T. Fujimoto, S. Suripant, P. Chongstitvatana and N. Chiba (2002). Modelling leaf shapes using l-systems and genetic algorithms. In *International Conference NICOGRAPH* (April, Japan, 2002), April, Japan.

[Rozenberg 1973] Rozenberg, G. (1973). T0l-systems and languages. *Information and Control 23*, 4 (1973), pp. 357–381.

[Sorrensen-Cothern et al. 1993] Sorrensen-Cothern, K., E. Ford and D. Sprugel (1993). A model of competition incorporating plasticity through modular foliage and crown development. *Ecological Monographs 63*, 3 (1993), pp. 277–304.

[Weber and Penn 1995] Weber, J. and J. Penn (1995). Creation and rendering of realistic trees. In *SIGGRAPH 1995 Conference Proceedings* (1995), ACM Press, pp. 119–128.

[Wonka et al. 2003] Wonka, P., M. Wimmer, F. Sillion and W. Ribarsky (2003). Instant architecture. In *SIGGRAPH 2003 Conference Proceedings* (2003), ACM Press, pp. 669–677.