Goal Oriented Animation of Plant Development

Callum Galbraith*
Dept. of Computer Science
University of Calgary

Przemyslaw Prusinkiewicz†
Dept. of Computer Science
University of Calgary

Campbell Davidson‡
Agriculture and Agri-Food Canada

Abstract

Previous work in the animation of plant development and expansion has centered on the botom-up approach, whereby models are specified in terms of developmental ruls. These models are inherently difficult to use for goal oriented modeling as the final structure and development of these models are emergent properties. Top-down modeling has previously been introduced to remedy this problem by allowing interactive control of local characterists based on global positional information. We explore methods of extending top-down modeling to the field of animation. The ideas of keyframe animation are used in conjunction with L-systems to create goal oriented models of plant growth in preformed branching structures.

Keywords: animation, modeling, plant development, L-systems, *Pellaea falcata*,*Syringa vulgaris* CV Congo,*Syringa reticulata*

1 Introduction

Previous work in the animation of plant development has resulted in visually appealing animations using models specified in terms of developmental rules of individual plant components. We refer to such models as bottom-up models. One well documented method of bottom-up modeling makes use of the formalism of L-systems [LIND68a][LIND68b][PRUS LIND90]. The final structure and development of bottom-up models are emergent properties of the developmental rules that specify these models. Consequently control of local properties is difficult, and a trial and error process is often needed to acheive desired results. This is not a problem when studying the processes that cause plants to grow, but if an animator wishes to exercise direct control over a developmental sequence and final structure for artistic purposes, a more intuitive and controlable method is needed.

To provide a more direct and intuitive control over plant structure, [PRUS et al.99] proposed the idea of top-down modeling. The key idea is to use global positional information of individual plant components to determine their local characteristics. This inverses the procedure of bottom-up modeling.

In this paper we explore methods of using global attributes of individual plant components to control their individual local charcteristics during development, which we refer to as top-down animation. The animation is produced by defining an initial model and a final model of the plant. User-specified interpolation functions are applied to compute the inbetween frames of the animation, in a similar manner to computer-aided keyframe animation techniques. Interpolation between keyframes has already been used to animate the growth of plants [LINT DEUS96] in the software system *xfrog* [LINT DEUS98], we explore this idea in the environment of L-systems. The interpolation of each individual organ is controled by the value of global attributes, such as its relative position within the plant. The process of creating an arbitrary development sequence for animation purposes is much improved using top-down animation. A much narrower range of knowledge is needed to proceed, in particular knowledge of biological processes of plant development is less critical in producing realistic animations of growth.

In Section 2 we outline previous work. An important part of this previous work is the method of top-down modeling which we outline in Section 3. Section 4 introduces top-down animation, which is an extension to top-down modeling. Three applications of top-down animation are presented, the growth of a *Pellaea falcata* leaf (Section 4.1), the growth of *Syringa vulgaris* CV Congo (Section 4.2) and the growth of *Syringa reticulata* (Section 4.3). We present a brief discussion of our results in Section 5.

2 Background

In this paper we use parametric L-systems [PRUS LIND90] as a framework for animation of plant development. L-Systems were formulated to describe the development of simple multi-cellular organisms [LIND68a][LIND68b], and were later extended to simulate the development of branching structures based on processes taking place at the level of plant modules [FRIJ LIND74]. Plant modules are a biological conception of separate entities within a plant such as flowers, leaves, internodes and apices. Parametric L-systems operate on *parametric words* which are strings of *modules*. All plant modules may be represented by parametric L-system modules. But it is not the case that all parametric L-system modules are plant modules, therefore we consider plant modules to be a subclass of parametric L-system modules. In this paper when we refer to modules we are discussing the more general class of parametric L-system modules, unless otherwise stated.

Each module consists of *letters* which belong to some alphabet V, with associated *parameters* which belong to the set of real numbers $\Re$. For example the module with letter $A \in V$ and corresponding parameters $a_1, a_2, ..., a_n \in \Re$ is written as shown in Module 1. The parameters $a_1, a_2, ..., a_n$ characterize attributes of the module. We can apply mathematical expressions called productions or rewriting rules to modules. Each production describes daughter modules which are created by a parent module. For example, Production 1 may be applied to Module 1 provided that $n = m$ and $p_1 < p_2$. If the production is applied then daughter modules B and C are created with one and two parameters respectively. Modules may optionally be specified without parameters, thus the module D which has no parameters is written in a parametric word as

*callum@cpsc.ucalgary.ca
†pwp@cpsc.ucalgary.ca
‡Morden Research Centre, Unit 100-101 Route 100, Morden, MB R6M 1Y5

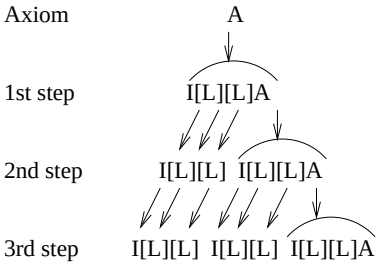


Figure 1: Derivation tree for axiom A and production $A \longrightarrow I[L][L]A$ for the first three derivation steps.

D, without any brackets.

Module 1

$A(a_1, a_2, ..., a_n)$

Production 1

$A(p_1, p_2, ..., p_m) : p_1 < p_2 \longrightarrow B(p_1)C(p_1, p_2)$

Modules may represent plant modules, which may themselves represent plant organs. Modules may also represent some local characteristic of the branching structure. Examples of modules which represent plant modules are A for apex which is a generative module within a plant, I for internode which is a branch or stem, and L for leaf. $I(n)$ can be used to specify an internode of length n. Examples of modules which do not represent plant modules are $+$ which indicates a branching angle, and ! which represents increasing the current width of branches. $!(n)$ can be used to set current width to n. A branching structure is encoded as a parametric word where subbranches are specified within square brackets. Models of branching structures are specified by a set of productions which simulate the growth processes occuring within a plant, and a parametric word called the axiom, which specifies the initial branching structure. The development of a branching structure is captured in a simulation by applying in parallel any productions which apply to the current parametric word, producing a new parametric word. Each application of the productions to the current parametric word is called a derivation step. Production 2 states that an apex A (the predecessor on the left side of the arrow) creates an internode I subtending a pair of leaves L and new apex A and produces a simple branching structure. Starting with the axiom A, successive derivation steps produce the parametric words shown in figure 1.

Production 2

$A \longrightarrow I[L][L]A$

This describes consectutive stages in the development of a simple monopodial branching structure, such as the one shown in figure 3. A monopodial branching structure is defined as one where the main apex exhibits continuous production of new apices, and all lateral apices exhibit terminal growth [PRUS LIND90]. In this paper we will also consider polypodial branching structures, such as the one shown in figure 15. A polypodial branching structure is defined as one where both the main apex and all lower order apices exhibit continuous production of new apices [PRUS LIND90].

To allow for easy construction of complex productions, we extend the production format as shown in Production 3 [HANN92][MĚCH98], where *pred* is the predecessor, an initial module, and *succ* is the successor or resulting modules. The optional field *cond* is a logical conditional statement which may include parameters from the predecessor, the production will only be applied when this condition holds true. The remaining fields *blockA* and *blockB* are composed of C-like statements, with

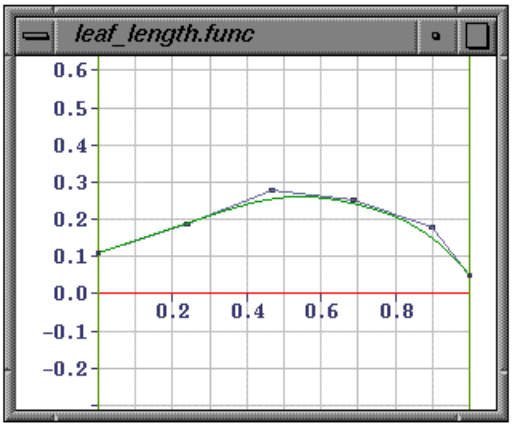


Figure 2: Interactive function editor window.

blockA being evaluated before the condition and *blockB* being evaluated only if the condition is true condition.

Production 3

pred : {*blockA*} *cond* {*blockB*} $\longrightarrow$ *succ*

To allow for even greater flexibility we use four additional blocks of C-like code which the user may specify. These blocks are [HANN92][MĚCH98]:

Start: {block1}
End: {block2}
StartEach: {block3}
EndEach: {block4}

and use global variables, accessible throughout the L-system. They are evaluated at the following times: *block1* is evaluated before the derivation begins; *block2* is evaluated after the derivation concludes; *block3* is evaluated before each derivation step; *block4* is evaluated at the end of each derivation step.

Expressions may also include generic function calls of the form *func(id,x)* [PRUS et al.99]. Parameter *id* is an integer function identifier, and the real number *x* is the argument to that function. Generic functions are defined graphically using an interactive function editor (figure 2). B-spline curves are used to define function plots, and the curves are constrained so that each *x* value is mapped to exactly one *y* value.

Branching structures are visualized by graphically interpreting L-system generated parametric words [SZIL QUIN79][PRUS86] using turtle geometry [ABEL DISE82]. Some examples of modules and there graphical interpretations are:

F Move forward one unit and draw a line of current line width.

F(x) Move forward 'x' units and draw a line of current line width.

! Decrease line width by the current decrement amount.

!(x) Set current line width to 'x'.

+,- Turn left or right by current angle increment.

+(x),-(x) Turn left or right by 'x' degrees.

[,] Push or pop the current turtle state from a stack of such states.

The software environment used for the work described in theis paper is a modified version of the program *cpfg*[PRUS LIND90][HANN92][MĚCH98], that incorporates all of the features mentioned above.

Previous animations of plant growth have been founded on bottom-up modeling of plants. In these models, productions capture the essence of the development of the plant at a local level. Early models were only capable of capturing discrete changes in the branching structure such as the creation of new plant modules, and discrete steps in the expansion of existing plant modules. Two subsequently devised methods to create smoothly growing branching structures using L-systems were timed L-systems [PRUS LIND90] and differential L-systems [PRUS et al.93], both of which are part of the bottom-up modeling methodology. Both continuous and discrete aspects of the development of branching structures could then be captured using biologically motivated local rules.

These formalisms are well suited for biologically motivated simulations of plant development. Growth and form are emerging properties of the rules that define development of the brancing structure and growth of individual plant modules. Effects of rules can be conveniently explored using computer simulations. However exercising direct control over the final form is very diffucult which makes these formalisms poorly suited for goal oriented modeling. This inability in bottom-up modeling has given rise to top-down modeling, which solves the problem of achieving intuitive control over plant form.

3 Top-Down Modeling of Plants

Top-down modeling of plants is the use of global positional information to characterize the properties of local plant components [PRUS et al.99]. Local properties may be determined analytically or procedurally from global properties. To illustrate the basic idea, consider the fern leaf from Figure 3, and the model of this fern leaf shown in Figure 4. To create this model a qualitative characterization of the leaf architecture was made consisting of a stem with leaflets on either side. The final model was defined using the top-down methodology to alter the local characteristics of the plant based on global positional information. Functions which mapped the global attribute position along the stem to local attributes were used to determine the following local attributes:

- Length of leaflets.
- Width of leaflets.
- Distance between leaflets.
- Asymmetry in the distance between leaflets.
- Branching angle.
- Color of leaflets.
- Stem thickness.
- Stem curvature.

In addition some randomization was introduced to each of these properties. And the angle between the stem and the ground was defined. This is the essence of top-down modeling.

Top-down modeling was described by [PRUS et al.99] in terms of Chomsky grammars [CHOM56]. Chomsky grammars define operations on strings of symbols as in Production 4 which indicates that objects represented by symbol A decompose into objects represented by symbols B and C. Productions in Chomsky grammars may be applied in any order, and are context free. Each application

Figure 3: Photograph of a *Pellaea falcata* leaf.

of a production is considered to be a derivation step when dealing with Chomsky grammars, as opposed to the L-systems definition of a derivation step which is the parallel application of all productions which apply to the current parametric word. In Chomsky grammars derivations will typically occur in depth first order until the string consists only of terminal symbols which may not be decomposed any further.

Production 4

$$A \longrightarrow BC$$

In order to use L-systems for top-down modeling, it was necessary to extend them. The notions of decomposition productions (Godin and Prusinkiewizc, personal communication)[MĚCH98] and interpretive productions (Godin and Prusinkiewizc, personal communication) [MĚCH98] (called homomorphisms in the cpfg L-system specifcation) were added to L-systems. These two new classes of productions are specified using the same syntax as "regular" L-system productions, they differ in the way they are applied to the parametric word. Decompositions are applied to the parametric word after each application of the "regular" L-system productions. They are applied until terminal modules are produced or some predefined maximum level of recursion has been reached. The resulting parametric word is the one that will be interpreted in the next derivation step. After the decomposition productions, interpretive productions are applied that alter the parametric word before it is interpreted geometrically, these alterations do not affect the parametric word which is passed to the next derivation step. Interpretive productions are applied recursively as are decompositions, until terminal modules are produced or some predefined maximum level of recursion has been reached. With decomposition and interpretive productions we redefine an L-system derivation step to be a single parallel application of the regular productions, followed by the re-

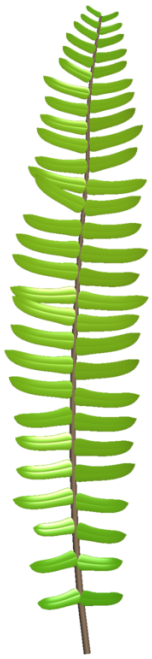


Figure 4: Model of a *Pellaea falcata* leaf.

cursive application of decomposition productions and the recursive application of interpretive productions.

Branching structures which are recursively defined can be used to represent many plants. The work described in this paper uses top-down models specified using decomposition productions. The parametric word specifying the final branching structure is computed in a single derivation step. Each instance of a plant module is represented by an individual module in the parametric word, as is each instance of a change in the current value of local characteristics such as branching angle, and color. Each module may have parameters which characterize its attributes. For example an internode may be represented by the module $F(m)$, where the parameter m characterizes the length of line to draw. Changes in the current value of the local characteristic line width may be represented by the module $!(n)$. During graphical interpretation the two modules $!(n)F(m)$ together specify that the local characteristic line width should be set to n, then a line of length m and width n should be drawn.

For example, L-system 1 produces a simple monopodial branching structure using top-down modeling. The derivation of the final parametric word is shown in figure 5, and its graphical interpretation is shown in Figure 6(a).

L-system 1

#define N 2
derivation length: 1
Axiom: $!(N/20)A(0)$
decomposition
$A(n) : n < N \longrightarrow F(N-n)[+B(n)][-B(n)]A(n+1)$
$A(n) : n >= N \longrightarrow B(N-1)$
$B(n) \longrightarrow F(N-n)$

In the resulting branching structure the parameter N controls the number of internodes in the main stem, and the corresponding number of branch pairs that are attached to successive nodes. Figure 6(b) shows the resulting structure with N equal to 20. The global positional attribute index of elements along the stem (represented

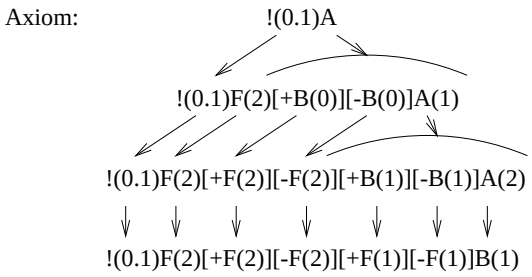


Figure 5: Derivation tree for L-system 1.

by the variable n in Lsystem 1) starting with 0 at the base and reaching $N-1$ at the top, is used to determine the local attribute *length* of internodes and leaves, both of which are represented by straight lines (modules of the form F(x) in the parametric word).

4 Top-Down Animation of Plant Development

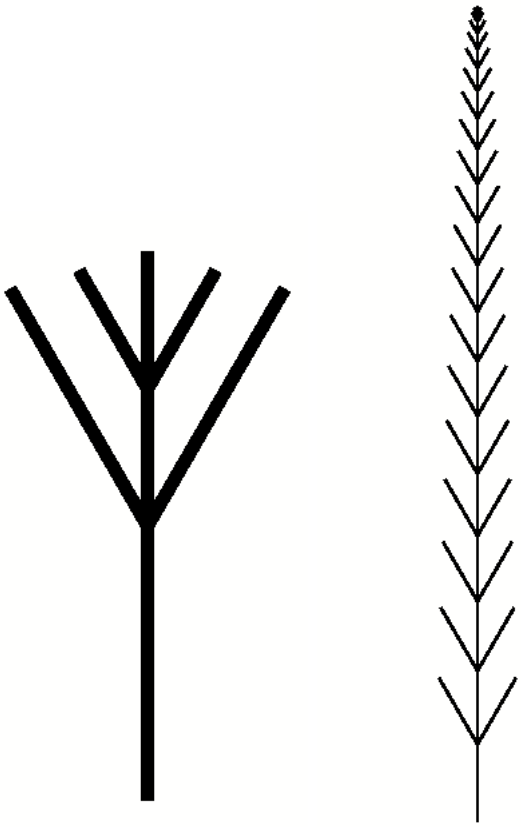
The goal of top-down animation is to produce a growth sequence that yields a predetermined form, in a predetermined way. For example, the series of time lapse photographs in figure 15 details the growth sequence of a *S. vulgaris* CV Congo inflorescense. The photographs define the growth and final state of the plant, and our goal is to reproduce this sequence. The issue of the tradeoff between faithfulness and complexity of plant models has been addressed in [PRUS98] and is not considered here.

To achieve the goal, we borrow ideas from computer aided keyframe animation. Keyframe animation in computer graphics began as an attempt to use computers to do the job of traditional animators [BAECKER69][BURT WEIN71][REEV81]. In computer aided keyframe animation, the animator creates a series of keyframes. Each keyframe is composed of individual elements which may be defined and manipulated separately. The in between frames are computed based on interpolation from one keyframe to the next, a process which is called inbetweening. Interpolation of each element is separately defined and includes specifying the behavior of each of its characteristics between keyframes. Interactive curve editors were introduced to control the change of attributes between keyframes, and the speed at which this change occured[BAECKER69]. In early keyframe animation, keyframes were constrained to have exactly the same number of elements.

In the method of plant animation described in this paper, the initial model and final model of the plant are considered to be key structures which we wish to interpolate between. We use the top-down modeling methodology to create the inital and final models. In our work the inital and final models were designed to match the photographs of actual plants as closely as possible. Interpolation functions are specified for each attribute of the model, and inbetweening produces the animation. As in early keyframe animation, we restrict our models to have the same number of modules to simplify the process of defining the interplolation. This means that the initial model is a preformed and scaled down version of the final model. This is biologically justified for many plant strucutres, as noted in [BELL91](page 90):

> "Such structures, delayed in their appearance, can be preformed, i.e. the whole leaf develops initially but its parts mature in sequence from base to apex."

As in top-down modeling, the parametric word specifying the final branching structure is computed in a single derivation step. Each instance of a plant module is represented by an individual



(a) $N = 2$ (b) $N = 20$

Figure 6: Images generated by L-system 1. The global constant N defines the number of internodes in the main stem.

module, as is each instance of a change in the current value of local characteristics (such as line width, color, etc.). Module 2 shows the basic structure of modules in the animation model. Four parameters are given for each time-varying attribute of the module. All attributes begin at the initial value, which is defined by the initial model, and end at the final value which is defined by the final model. Start time defines when attributes begin to change from their initial value to their final value. Growth time defines the length of time required for this change to occur.

Module 2

A(initial value,final value,start time,growth time)

For each module type (eg. internodes, leaves), and each class of local characteristic (eg. branching angle, color) in the model, we assign a function that describes the interpolation from initial value to final value over the period of growth. We call such functions growth rate functions. Our growth rate functions take as input the value $i = (t - t_s)/t_g$ where t is the current time, t_s is the start time of growth from the beginning of the simulation, and t_g is the duration of growth. This function will return 0 when $i <= 0$, $func(id, i)$ when $0 < i < 1$ and 1 when $1 <= i$. A function value of 0 indicates that the attribute is at its initial value, and a function value of 1 indicates that the attribute is at its final value. Intermediate function values specify a linear interpolation between initial and final value of the attribute. Many growth rate functions

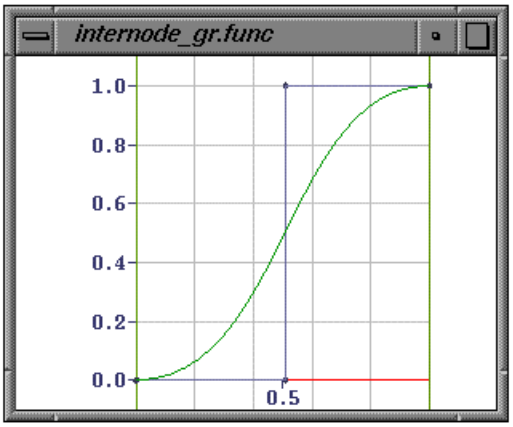


Figure 7: Standard sigmoidal growth rate function.

in nature have a sigmoidal character, meaning that growth is initally very slow, then accelerates, and then tapers off again as the growth of the element nears completion. Accordingly, $func(id, i)$ should be a sigmoidally shaped function. Most of the growth rate functions used in the examples in this paper are defined as shown in Figure 7.

To animate the development of the branching structure, we use time dependant interpretive productions. These productions determine the current value of each attribute in a module at a given time t, based on the four input parameters and the appropriate growth rate function. They then create the appropriate modules for the graphical interpretation. The value of t is stored in a global variable which is updated at the end of each derivation step. The change in time between derivation steps may be arbitrarily small or large to animate growth over any time period. A complete animation is produced by performing a graphical interpretation of the parametric word after each derivation step.

For example, revisiting L-system 1 we rewrite it in preparation for animation as L-system 2.

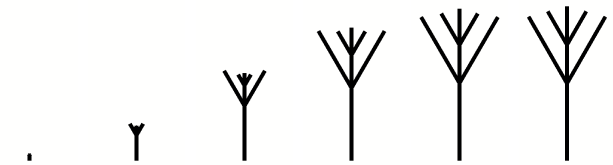
L-system 2

#define N 2
#define delay 0.2
#define MAX_T 6
#define DELTA_T 0.1
derivation length: MAX_T / DELTA_T
Start: $\{t = 0;\}$
EndEach: $\{t = t+$*DELTA_T*$;\}$
Axiom: $!(N/20)A(0)$
decomposition
$A(n) : n < N \longrightarrow$
$\quad I(0, N - n, n/N, 5)[+B(n)][-B(n)]A(n + 1)$
$A(n) : n >= N \longrightarrow B(N - 1)$
$B(n) \longrightarrow L(0, N - n, n/N + delay, 5)$

The application of the decomposition productions produces the following parametric word in the initial derivation step which describes the preformed branching structure. The parametric word does not change during later derivation steps.

$$!(0.1)I(0, 2, 0, 1)[+L(0, 2, 0.5, 1)][-L(0, 2, 0.5, 1)]$$
$$I(0, 1, 0.5, 1)[+L(0, 1, 1, 1)][-L(0, 1, 1, 1)]L(0, 1, 1, 1)$$

The modules $I(0, 2, 0, 1)$ and $I(0, 1, 0.5, 1)$ represent the two successive internodes in the main branch, and the modules $L(0, 2, 0.5, 1)$ and $L(0, 1, 1, 1)$ represent leaflets of different final size attached to the main branch. The number of derivation steps is determined so that if t is incremented by $DELTA\_T$ in each step



(a) $t = 1.0$ (b) $t = 2.0$ (c) $t = 3.0$ (d) $t = 4.0$ (e) $t = 5.0$ (f) $t = 6.0$

Figure 8: Six frames from the animation produced by L-system 2. t indicates the time at which each frame was generated.

then its final value will be MAX_T. The animation is produced by graphically interpreting the parametric word at each derivation step using time-dependant interpretive productions. Production 5 is the interpretive production applied to module I. The local parameters of the production I are respectively *initial length*, *final length*, *start time* and *growth time*. Together with the growth rate function for internodes, which is the function with $id = 1$ in Production 5, they determine the value of length at each time t. The module L is interpreted with a similar interpretive production. Six frames from the animation produced by L-system 2 are seen in Figure 8, Figure 8(f) shows the fully grown branching structure.

Production 5

homomorphism

$I(l_i, l_f, t_s, t_g) : *\{$
$if(t < t_s)\{l = l_i;\}$
$else\{$
$\quad if(t < t_s + t_g)\{$
$\quad\quad l = l_i + func(1, (t - t_s)/t_g) * (l_f - l_i);\}$
$\quad else\{l = l_f;\}$
$\}$
$\} \longrightarrow F(l)$

We can control any number of attributes in a similar manner. The only remaining task is how to best determine the start time and growth time for each individual module in the model. Preformed plants exhibit expansion of individual plant components in a sequential way. The first part of the structure to begin expansion is the base of the branching structure, and after that each part in turn will begin its own expansion until the extremities are reached, which are the last components to expand, creating the final form of the plant. To control this sequential expansion, we first determine the absolute start times from the beginning of the simulation for each internode, so that the sequence of expansion is replicated. The start time of other modules are specified as a time offset from the start time of the internode which the module effects (eg. color modules), or from the internode the module is connected to (eg. leaf modules).

We begin by examining the case of internode modules in the simple monopodial structure generated by L-system 2. In the spirit of the top-down modeling methodology, a simple and intuitive method of mapping a global attribute to the local characteristic start time is desired. An interactively edited function is an intuitive method of doing this. There are two considerations in defining and using such a function. The first is what global attribute do we use as input for the function, and the second is how the local characteristic start time is affected by the resulting function value.

There are four obvious types of global positional information which may be used as the function input. They are: the absolute position of the internode in the branching structure measured from the bottom of the stem; the absolute position measured from the top of the stem; the absolute index of the internode counted from the bottom of the stem, with the first module having index 0, the second module having index 1 and so on; the absolute index counted from the top of the stem. We consider two of these possibilites, the first is the absolute position from the base of the stem, normalized to the interval [0,1] by dividing the position by the total length of the stem. The second is the index of each internode from the base of the stem, normalized to the interval [0,1] by dividing the index by the index of the last internode in the stem. By normalizing our function input to lie in the interval [0,1] we can use the same functions for any size of branching structure. We have observed that the use of the normalized position gives a more intuitive control than the normalized index, as there is a one to one linear mapping between position along the stem in the final model, and the function argument in the graphical function editor. Using the normalized index, we do not have such a simple relationship between position along the stem and the function input. On the other hand, if we are using a model with an exact known number of internodes with start times which we have measured experimentally, then using the normalized index is more convenient.

To determine the effect the function value has on start time, we have also considered two approaches. One is to use the function value as the absolute start time from the beginning of the simulation, and the other is to use the function value as the time difference from the start time of the previous internode in the stem to the start time of the current internode. For the monopodial structure we are considering, using the function value as the absolute start time was the easier method to use. The second method defines the absolute start time of an internode as the sum of delay times of internodes from index 0 to the current internode. This is similar to integrating the function to define the absolute start time. Intuitive control is more difficult here. Imagine we wish to delay the growth of internodes near the base of the stem more than they currently are, while leaving the start times of the internodes at the top of the stem unaffected. Using the first method we simply alter the function in the region of interest. Using the second method, we must increase the delay time of the internodes of interest, and then decrease the delay time of later internodes. Even having decreased the delay of some later internodes, it is difficult to gauge by how much they should be decreased to maintain the original start times of the internodes at the top of the stem.

To determine growth time we pass the same argument we passed to start time to another interactively edited function, whose function value is interpreted as the amount of time required for the attribute to go from its initial value to its final value. The start time and growth time for all other modules in the model are determined in a similar way. Two interactively edited functions are specified for each type of module in the model, one for start time and one for growth time. Both of these functions will be passed the same input as the start time function for internodes. They are interpreted respectively, as the time offset as previously described, and the time needed to grow.

The simple monopodial structure we have considered thus far is easy to work with as there is only one level of branching. However, many plants also have branches of higher order. We have applied the concept of top-down animation recursively to animate the growth of higher-order branching structures. In a species which has higher-order branches, individual plants may have different final geometry as well as a different maximum order of branches. Nevertheless, they grow in a similar way. Consequently it is possible to have a single parameterized animation model for a given plant species, that can then be used to animate the development of an arbitrarily preformed branching structure of the species in question. To keep these models relatively simple, we use the same functions for determining the value of local characteristics based on global

attributes, for each order of branch in the branching structure.

As an example we consider a simple polypodial branching structure, and begin by determining the start time for internodes. In the simple monopodial structure we found it most convenient to use the relative position along the stem, measured from the base, as the input to our function, and to assign the return value the meaning of absolute start time. The same considerations apply to the function input when dealing with a polypodial structure. So we can use either the normalized position or the normalized index as the function input depending on the situation. We specify the return value of the start time function to be the time lapse from the start time of the previous internode, as it is more intuitive to think of start time in this way when dealing with recursively defined multi-order branching structures.

On the main branch indices fall in the range $[0, N-1]$, and are normalized by dividing the index by $N-1$. Subbranches usually have fewer elements than their parent branch, and thus we must determine how to normalize the function input when dealing with a subbranch. In the case where we use the normalized index as the function input, we have explored two ways of normalizing the index to work with subbranches. Let us consider an arbitrary subbranch with M elements, (we assume that $M <= N$). If we assign indices in the range $[0, M-1]$ then we normalize by dividing the index by $M-1$. An alternate method is to assign indices in the range $[N-M, N-1]$ and then normalize by dividing the index by $N-1$. These two approaches provide drastically different results when using the same function for branches with different numbers of elements. We have found the second method easier to work with. Both possibilities also exist when using the position along the stem rather than the index.

Growth time is treated as before, passing it the same argument as we pass to start time and interpreting the function value as the growth time. For higher order branching structures we introduce an additional function, delay time. Every time a branch gives rise to a new subbranch, the growth of the subbranch may be delayed beyond the time when the parent branch began growing. The function is passed the same value as start time and growth time, and the return value is the amount of time by which growth of the subbranch will be delayed. The start time of the first internode on the subbranch will now be its own start time plus the delay time, plus the absolute start time of the internode on the parent branch.

Rewriting L-system 2 as L-system 3 to include polypodial growth we note the following changes. Instead of producing lateral apices which are terminal branches, the lateral apices are also polypodial structures. The normalized index is the global attribute being used to control local characteristics, but in the subbranches we use the normalizing scheme where indices are in the range $[N-M, N-1]$ and are then normalized by dividing the index by $N-1$. Start time is determined using an interactively edited function with id equal to 1 (shown in Figure 9). The same function is used to specify start time for every order of branch in the model. When creating the preformed branching structure we pass the cumulative start time of the current module to all daughter modules which it creates, so that absolute starting times can be computed. If the daughter module is a subbranch then we add the delay time to the cumulative start time. The cumulative start time is the second parameter to production A. Figure 10 shows six frames from the animation produced by L-system 3, Figure 10(f) shows the fully grown branching structure.

L-system 3

#define N 2
#define delay 0.5
#define MAX_T 8
#define DELTA_T 0.01
derivation length: MAX_T / DELTA_T
Start: $\{t = 0;\}$

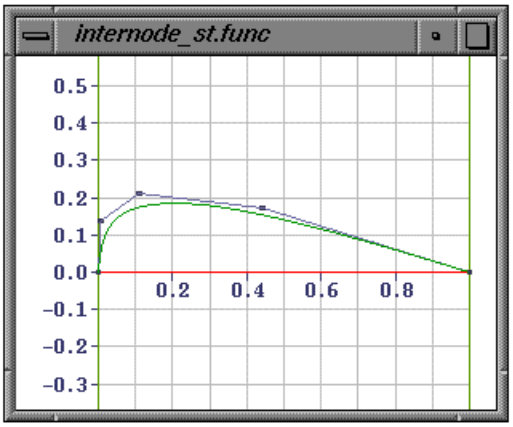
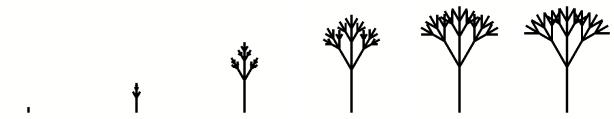


Figure 9: Start time function for internodes in L-system 3.



(a) $t = 1.0$ (b) $t = 2.0$ (c) $t = 3.0$ (d) $t = 4.0$ (e) $t = 5.0$ (f) $t = 6.0$

Figure 10: Six frames from the animation produced by the L-system 3. t indicates the time at which each frame was generated.

EndEach: $\{t = t+DELTA\_T;\}$
Axiom: $!(N/20)A(0,0)$
decomposition
$A(n,s) : n < N\{ns = func(1, n/N) + s;\} \longrightarrow$
$I(0, N-n, ns, 5)$
$[+A(n+1, ns+delay)]$
$[-A(n+1, ns+delay)]$
$A(n+1, ns)$
$A(n,s) : n >= N \longrightarrow L(0, 1, func(1, n/N) + s, 5)$

As before we will also need time-dependant interpretive productions to allow graphical interpretation of modules I and L. These productions will be specified in exactly the same way as before, with appropriate growth rate functions for internodes and leaves.

The models presented thus far in this section are simple examples of the underlying concepts. We now describe three fully realized models where all of the local characteristics are controled by global attributes via interactively edited functions.

4.1 Animating Growth of *Pellaea falcata* Fern Leaf

Earlier we saw a photograph and corresponding top-down model of the sickle fern leaf *Pellaea falcata* (Figures 3 and 4). This section describes the process of animating its growth using top-down animation. A total of nine attributes were determined using top-down modeling, they are:

- Length of leaflets.
- Width of leaflets.
- Distance between leaflets.

(a) (b) (c) (d) (e) (f)

Figure 13: Frames from intermediate animation of growthing *Pellaea falcata.*

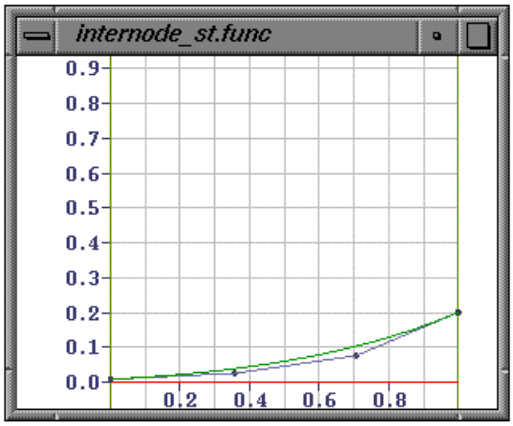


Figure 11: Internode start time function for animation in Figure 13.

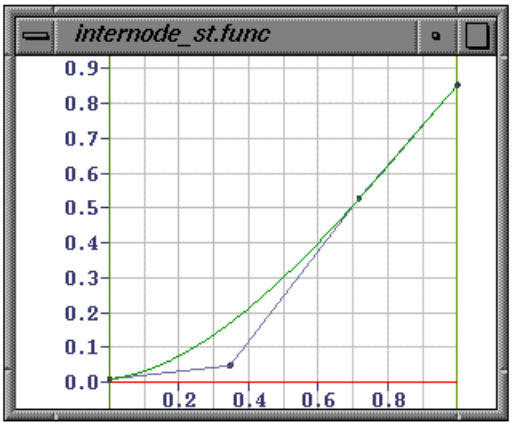


Figure 12: Internode start time function for animation in Figure 14.

- Asymmetry of leaflets.
- Branching angle of leaflets.
- Color of leaflets.
- Curvature of stem.
- Thickness of stem.
- Angle between stem and ground.

Curvature of the stem was actually specified using the intrinsic definition of a curve, that is its turn, pitch and roll were separately defined along its length. All nine of these attributes were controlled in the animation. The interactive function editor was used to specify the initial and final values of all local characteristics based on a globally derived input (with the exception of angle between stem and ground, which is a simple scalar value), including start and growth times, and the growth rate. The global attribute used was the normalized position from the base of the stem. Start time and growth time were determined using the method described for a simple monopodial structure, using the normalized position as the function input, and using the function value as the absolute start time from the beginning of the simulation.

Six frames from two animations are shown in figures 13 and 14. The second animation is the final result. The only attribute altered between these animations is the start time of various internodes, the functions defining start times for each animation are shown in figure 11 for the first animation, and figure 12 for the second animation. Each attribute was interactively edited using feedback from the intermediate animations until the desired result was achieved. The process of interactive refinement only took about one half hour to produce the final animation.

(a) (b) (c) (d) (e) (f)

Figure 14: Frames from final animation of growthing *Pellaea falcata.*

4.2 Animating Growth of *S. Vulgaris CV Congo*

Figure 15 shows a developmental sequence of photographs of a *Syringa vulgaris* CV Congo inflorescence, which is an example of a preformed polypodial plant. The initial and final models were built based on experimental data detailing the exact branching structure for a particular *S. vulgaris* CV Congo inflorescence (for technical reasons it is not the same inflorescence as seen in Figure 15). The data was collected at the Agriculture/Agrifood Canda Research Station in Morden, Manitoba, and describes four orders of branches. Internodes on the main stem are of order 0, internodes on the child branches of the main stem are of order 1,and so on up to branches of order 3. Each branch is composed of internodes, terminates in a flower and may have child branches of the next order. Subbranches are created on each branch in symmetrycal pairs, each consecutive pair of subbranches are attached to the parent branch after rolling on the axis of the parent branch by an angle of 85 degrees. Branches of each order have a maximum number of internodes defined by an array whose values are accessed using the order of the branch as an index (table 1). If the number of internodes for a given branch is less than the maximum number of internodes for that order branch, we consider the branch to be a copy of the top portion of a branch of the same order which does have the maximum number of internodes. When accessing a data table, the internodes indices on a particular branch with M internodes are defined to be in the range $[N - M, N - 1]$ where N is the maximum number of internodes for branches of that order, and the index is used directly as an index into the data table. The number of internodes in each pair of subbranches for each order of parent branch is defined in a 2-D array, which is indexed with the order of the parent branch, and the index of the internode to which the subbranches are attached (table 2). The lengths of internodes for each order of branch is defined in another 2-D array, which is indexed with the order of the branch, and the index of the internode (table 3).

The models interpolation was defined using the method described for the polypodial L-system 3. The global attribute used for input to interactively edited functions is the normalized index, which is calculated as follows. The internode indices on a particular branch with M internodes are defined to be in the range $[N - M, N - 1]$ where N is the maximum number of internodes on the branch with order 0, and are normalized by dividing the index by N. Initial and final branching angles, and start times and growth times for internodes, are all specified using interactively edited functions taking as input the normalized index. Start times are defined as a time delay from the start time of the previous internode.

Parameters for flowers are constant regardless of their position in the branching struction. They begin to grow after the internode they are attached to begins to grow plus a constant time delay. The growth time and unfolding motion of the petals are also constant throughout the model.

The results of the model can be seen in Figure 15 which includes six photographs from the growth sequence and the corresponding frames from the animation.

Branch Order	Maximum Internodes
0	13
1	7
2	2
3	1

Table 1: Maxmimum number of internodes in branches of each order (*S. vulgaris* CV Congo).

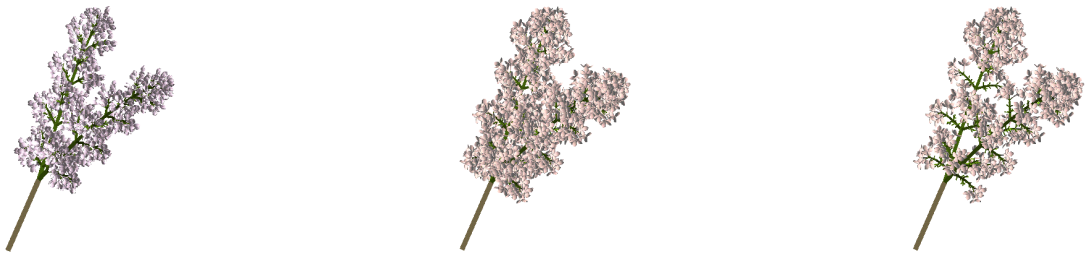
4.3 Animating Growth of *S. Reticulata*

A more complex example of a preformed polypodial plant is *Syringa reticulata.* Figure 16 shows a developmental sequence of photographs of a *S. reticulata* inflorescence. To animate this developmental sequence we will use the same methodology as we did for *S. vulgaris* CV Congo. Once again we have available experimental data detailing the length of respective internodes and number of internodes in subbranches of every order. The data is summarized as before in tables 4, 5 and 6 and was collected at the same time as the data for *S. vulgaris* CV Congo. The main difference between the

(a) May 19 (b) May29 (c) June 1

(d) June 5 (e) June 12 (f) June 15

(g) May 19 (h) May29 (i) June 1



(j) June 5 (k) June 12 (l) June 15

Figure 15: Time lapse photography of growing *S. vulgaris* CV Congo and corresponding animation frames. Note: images 15(h) and 15(i) were scaled by approx. 1.5, and image 15(g) was scale by approx. 6.

(a) May 19 (b) May26 (c) June 1

(d) June 15 (e) June 22 (f) June 26

(g) May 19 (h) May26 (i) June 1

(j) June 15 (k) June 22 (l) June 26

Figure 16: Time lapse photography of growing *S. reticulata* and corresponding animation frames. Note: images 16(h) and 16(i) were scaled by approx. 1.5, and image 16(g) was scale by approx. 6.

Internode Index	Branch Order			
	0	1	2	3
0	7	2	1	0
1	7	2	0	
2	7	1		
3	7	1		
4	6	1		
5	6	1		
6	6	0		
7	5			
8	4			
9	3			
10	2			
11	1			
12	0			

Table 2: Number of internodes in subbranches attached at different indices of parent branches of each order (*S. vulgaris* CV Congo).

Internode Index	Branch Order			
	0	1	2	3
0	15.5	8.8	4.0	5.0
1	13.1	8.0	4.9	
2	38.9	7.7		
3	19.8	4.1		
4	33.3	4.5		
5	21.2	4.4		
6	25.6	1.1		
7	16.2			
8	12.4			
9	9.0			
10	1.4			
11	0.0			
12	0.0			

Table 3: Length of internodes in branches of each order (*S. vulgaris* CV Congo).

two models is that our *S. reticulata* model has 6 orders of branches as opposed to four for the congo lilac.

The only other difference is that we have specified a functional relationship between two of the attributes in the model. The program cpfg has a built-in mechanism called tropism [MĚCH98] which can be used to simulate the effect of gravity on branching structures. Tropisms are defined as a vector and an elasticity ratio, and effect the path of the turtle during graphical interpretation by causing the turtle's path to tend to become parallel with the tropism vector at a rate based on the current elasticity ratio. By setting a tropism vector in the down direction, branches will tend to first sag, and then point straight down as we add more segments. Examination of the photographs in Figure 16 shows that over time the *S. reticulata* inflorescence first grows straight up (Figure 16(a)), then sags due to gravity as the weight of the plant increases (Figure 16(b)), and then straightens out again as the strength of the stem increases (Figures 16(c) - 16(f)). In order to capture this behaviour, we have added a tropism attribute to the internode modules within the model, with additional parameters begin given as shown in Module 3.

Module 3

I(initial length,final length,length start time,length growth time,initial tropism ratio,final tropism ratio,tropism ratio start time,tropism ratio growth time)

The initial Tropism values are set very high, and the final tropism values are set so that a slight sagging is visible. The result of this is that initially the plant almost immediately grows straight down. To fix this we use the following formula:

$$r_{tf} = r_{ti} * (l_f/l_t)^3$$

where l_f is final length of the internode, l_t is the length of the internode at time t, r_{ti} is the initial tropism elasticity ratio at time t, and r_{tf} is the final tropism elasticity ratio at time t which will affect the direction of the turtle during graphical interpretation. The effect of this is that when the length of the internode is small, and the initial elasticity ratio is great, the ratio $(l_f/l_t)^3$ will greatly reduce the final elasticity ratio that is applied. As $l_t \longrightarrow l_f$ we get the corresponding result that $r_{tf} \longrightarrow r_{ti}$, so by appropriate modifications of the elasticity ratio growth rate function, we can define an interpolation of the elasticity ratio that begins at 0 when $t = 0$, increases as the internodes lengthen, and decreases as the internodes achieve their final length, thus simulating the effect of gravity on the growing plant. The power of 3 was determined experimentally to give good results.

The results can be seen in Figure 16. Which includes both the time lapse photographs and the corresponding frames from the animation.

Branch Order	Maximum Internodes
0	20
1	12
2	9
3	5
4	2
5	1

Table 4: Maxmimum number of internodes in branches of each order (*S. reticulata*).

Internode Index	Branch Order					
	0	1	2	3	4	5
0	12	9	5	2	1	0
1	12	7	4	1	0	
2	12	7	4	1		
3	11	6	2	1		
4	10	5	1	0		
5	10	3	1			
6	9	2	1			
7	8	2	1			
8	6	2	0			
9	5	2				
10	4	1				
11	4	1				
12	3					
13	2					
14	2					
15	2					
16	2					
17	1					
18	1					
19	0					

Table 5: Number of internodes in subbranches attached at different indices of parent branches of each order (*S. reticulata*).

5 Discussion

If we look at the results of our animations in Figures 14, 15 and 16 we can see that we have captured the overall essence of the developmental sequence, but that some details are not quite right. This is

Internode Index	Branch Order					
	0	1	2	3	4	5
0	1.5	16.3	15.1	3.0	2.0	3.0
1	25.3	23.6	11.3	5.0	1.0	
2	46.6	15.9	9.7	5.0		
3	33.2	11.6	7.8	2.0		
4	28.7	8.8	3.8	1.0		
5	23.7	8.1	3.9			
6	23.2	4.5	1.8			
7	18.4	4.4	2.1			
8	13.9	3.4	1.3			
9	12.7	1.8				
10	11.2	1.5				
11	6.5	1.4				
12	10.7					
13	6.5					
14	5.4					
15	3.9					
16	3.1					
17	2.2					
18	2.1					
19	1.5					

Table 6: Length of internodes in branches of each order (*S. reticulata*).

due to the recursive nature of our models. As the order of branching structure increases, the level of intuitive control available decreases. Thus we have the most direct control when animating the growth of a single order branching structure, such as the leaf in Figure 3, and the least control when animating plants with higher order branching structures such as in Figure 16.

This represents a problem as the purpose of top-down animation is to provide easy and intuitive control over such processes. The answer would seem to be to specify different growth control functions for each order of branch, however second order branches at the base of the main stem and at the top of the main stem may behave in qualitatively different ways, so this may not be sufficient, and even if it is sufficient, it adds significantly to the complexity of the model. Specifying separate functions for each individual branch would certainly give us the control we need, but would be far too complex to deal with (consider the model of *S. reticulata*, which has approximately 1800 branches). In addition when working with polypodial branching structures we do not know what the final branching structure will be, unless we model each instance separately. The use of allometric relations and domain mapping which have been introduced in top-down modeling [PRUS et al.99] may well yield advancements to top-down animation. Easy and intuitive control of higher order branching structures is the largest area for improvement in this work.

The model for animating growth presented in this paper works well for preformed plants which experience a single season of growth, but what about longer term plants, or ones which produce new structures as they grow? The idea of modeling plants as sets of parametric productions, graphically visualized at any time t using interpretive productions can be extended to these two areas. Creation of new organs can be incorporated with regular L-system productions which can produce several new parametric productions from an existing one thus modeling new growth. Additional L-system rules could take an existing parametric production and rewrite it with new parameters, thus allowing for multiple stages of growth. The difficulty with these two methods are that we again lose direct control over these processes. These two extensions form a part of the bottom-up modeling methodology. A more intuitive means of capturing these processes is needed for top-down animation.

Lastly, defining attributes which are functionally related, such as the tropism elasticity ratio and internode length in the model of *S. reticulata* requires more study. The approach described in this paper was experimental and difficult to control in an intuitive way, and the results obtained are not as good as we would like.

6 Conclusion

Most of the previous work on animation of plant development has centered on bottom-up modeling. Plant structures are emerging phenomena of a set of rules which govern a derivation of the developmental process. These rules can be quite compact while capturing complex physiological processes which control plant development in nature. A simple assumption based upon this is that bottom-up modeling of plant development is easy. In fact the opposite case is true. Rules must be fine tuned to correctly simulate biological reality, and detailed biological knowledge is required for every plant the animator wishes to create. Even with a biologically correct model, control over the visual characteristics of such models is difficult.

If we are presented with a predetermined final goal for the structure and visual characteristics of a plant we require a more intuitive means of modeling to achieve our goal. Top-down modeling was devised to allow easy and intuitive control over plant form. It involves mapping global attributes, typically global positional information, to local characteristics such as the current length of leaves or thickness of the stem.

Top-down animation is an extension to top-down modeling which allows us to animate the growth of a plant so that it arrives at a predetermined structure which is our goal. Top-down animation works by specifying an initial and final model using top-down modeling, and then specifying the interpolation of local characteristics betweeen these two models based on the value of global attributes. The process is very similar to to computer aided keyframe animation.

Several methods were explored to map global attributes to local characteristics. Some strengths and weaknesses were discovered with each of the methods, but no determination was made of their suitabiliy in different circumstances, or their general practicality in top-down animation. Theoretical concepts of positional information, allometry, grammars and fractals all bear on this topic, and although these areas have been researched for many years, there remain interesting topics for further study in this area.

7 Acknowledgements

This research was supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada.

References

[ABEL DISE82] H. Abelson and A. A. diSessa. *Turtle Geometry*. M.I.T. Press, Cambridge, 1982.

[BAECKER69] R. M. Baecker. *Picture Driven Animation*. Springer Joint Computer Conference, 1969.

[BELL91] Adrian D. Bell. *Plant Form*. Oxford University Press, 1991

[BURT WEIN71] N. Burtnyk and M. Wein. *Computer-Generated Key-Frame Animation*. Journal of the SMPTE (Society of Motion Picture and Television Engineers), volume 80, pages 149-153, 1971.

[CHOM56] N. Chomsky. *Three Models for the Description of Language.* IRE Trans. on Information Theory, volume 2, number 3, pages 113-124, 1956.

[FRIJ LIND74] D. Frijters and A. Lindenmayer. *A model for the growth and flowering of Aster novae-angliae on the basis of table (1,0)L-systems.* In *L Systems*, edited by G. Rozenberg and A. Salomaa. Springer-Verlag, pages 24-52, 1974.

[HANN92] J. S. Hannan. *Parametric L-systems and their Application to the Modelling and Visualization of Plants.* Ph.D. Thesis, University of Regina, June 1992.

[HAMM96] M. Hammel. *Differential L-systems and their Application to the Simulation and Visualization of Plant Development.* Ph.D. Thesis, University of Calgary, June 1996.

[LIND68a] Aristid Lindenmayer. *Mathematical Models for Cellular Interaction in Development. I. Filaments and One-Sided Inputs.* J. Theor. Biol. volume 18, pages 280-299. 1968.

[LIND68b] Aristid Lindenmayer. *Mathematical Models for Cellular Interaction in Development. II. Simple and Branching Filaments with Two-Sided Inputs.* J. Theor. Biol. volume 18, pages 300-315. 1968.

[LINT DEUS96] B. Lintermann, O. Deussen. *Interactive Modelling and Animation of Branching Botanical Structures.* Computer Animation and Simulation '96, Springer Wien New York, pages 139-151. 1996.

[LINT DEUS98] B. Lintermann, O. Deussen. *XFROG 2.0.* www.greenworks.de, December 1998.

[MĚCH98] R. Měch. *CPFG Version 3.4 User's Manual.* Department of Computer Science, University of Calgary. May 1998.

[PRUS86] Przemyslaw Prusinkiewicz. *Graphical Applications of L-systems.* In Proceedings of Graphics Interface '86 - Vision Interface '86, pages 247-253, 1986.

[PRUS LIND90] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants.* Springer Verlag, 1990.

[PRUS et al.93] Przemyslaw Prusinkiewicz, Mark S. Hammel and Eric Mjolness. *Animation of Plant Development.* Proceedings of SIGGRAPH'93, pages 351-360.

[PRUS et al.94] P.W. Prusinkiewicz, W.R. Remphrey, C.G. Davidson and M.S. Hammel. *Modeling the architecture of expanding Fraxinus pennsylvanica shoots using L-systems.* Canadian Journal of Botany, volume 72, pages 701-714, 1994.

[PRUS98] Przemyslaw Prusinkiewicz. *In Search of the Right Abstraction: The Synergy Between Art, Science, and Information Technology in the Modeling of Natural Phenomena.* Manuscript, Department of Computer Science, University of Calgary. 1998.

[PRUS et al.99] Przemyslaw Prusinkiewicz, Radik Karwowski, Campbell Davidson. *Top-down Modeling of Plants.* Manuscript, Department of Computer Science, University of Calgary. Feb. 1999.

[REEV81] William T. Reeves. *Inbetweening for Computer Animation Utilizing Moving Point Constraints.* Computer Graphics, ACM SIGGRAPH, volume 15, number 3, pages 263-269.

[SZIL QUIN79] . A. L. Szilard and R. E. Quinton. *An Interpretation for DOL Systems by Computer Graphics.* The Science Terrapin, volume 4, pages 8-13. 1979.