The use of positional information in the modeling of plants

Przemyslaw Prusinkiewicz, Lars Mündermann, Radoslaw Karwowski, Brendan Lane

Department of Computer Science, University of Calgary, Calgary, Alberta, Canada T2N 1N4
pwp|lars|radekk|laneb@cpsc.ucalgary.ca

Abstract

We integrate into plant models three elements of plant representation identified as important by artists: posture (manifested in curved stems and elongated leaves), gradual variation of features, and the progression of the drawing process from overall silhouette to local details. The resulting algorithms increase the visual realism of plant models by offering an intuitive control over plant form and supporting an interactive modeling process. The algorithms are united by the concept of expressing local attributes of plant architecture as functions of their location along the stems.

CR categories: F.4.2 **[Mathematical Logic and Formal Languages]**: Grammars and Other Rewriting Systems, I.3.5 **[Computer Graphics]**: Computational Geometry and Object Modeling, J.3 **[Life and Medical Sciences]**: Biology.

Keywords: realistic image synthesis, interactive procedural modeling, plant, positional information, phyllotaxis, Chomsky grammar, L-system, differential turtle geometry, generalized cylinder.

1 Introduction

Forward simulation of development is a well established paradigm for modeling plants. It underlies, for example, the AMAP simulation software [9] and modeling methods based on L-systems [28]. In both cases, a plant is modeled using a set of rules that describe the emergence and growth of individual plant components. The simulation program traces their fate over time, and integrates them into the structure of the whole plant.

Over the years, the simulation paradigm has been extended to include a wide range of interactions between plants and their environments [15, 21]. The resulting models have gained acceptance as a research tool in biology and have led to increasingly convincing visualizations. In image synthesis applications, however, the simulation-based approach has several drawbacks:

- Visual realism of the models is linked to the biological and physical accuracy of simulations. This requires the modeler to have a good understanding of the underlying processes, makes comprehensive models complicated, and results in long simulation times.

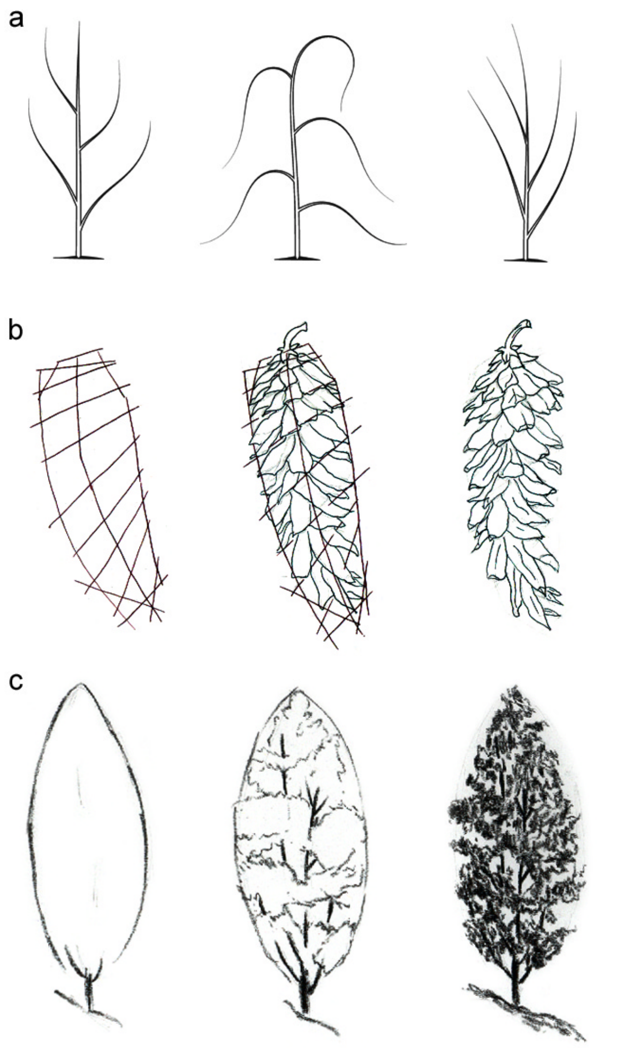

Figure 1: Selected elements of the artistic representation of plants: (a) posture, (b) regular arrangement and gradual variation of organs along an axis, and (c) progression from silhouette to detail in the drawing process. Figure (a) is based on [37, page 41], (b) is redrawn from [38, page 68], and (c) is redrawn from [24, page 13].

- Global characteristics of plant appearance, such as the curvature of plant axes, the density of organ distribution, and the overall silhouette of the plant, are emergent properties of the models and therefore are difficult to control.

Methods for creating visually realistic representations of plants

ACM SIGGRAPH 2001, 12-17 August 2001, Los Angeles, CA, USA
© 2001 ACM 1-58113-374-X/01/08...$5.00

have long been understood by artists (Figure 1). Important elements include plant *posture*, defined by the angles of insertion and curvature of organs, and the *arrangement* and gradual *variation* of organs on their supporting stems. The drawing process progresses in a *global-to-local* fashion, from silhouette to detail. Inspired by the quality of botanical illustrations, we have developed a plant modeling method that supports similar elements and processes. The proposed method *inverts* the local-to-global operation of simulation-based models by progressing from global plant characteristics specified by the user to algorithmically generated details. The algorithms are united by their use of *positional information*, which we define as the position of plant components along the *axes* of their supporting stems or branches. User-defined functions map this information to *morphogenetic gradients* [2], which describe the distributions of features along the axes.

The notions of positional information and morphogenetic gradients unify and generalize several plant-modeling concepts that have already appeared in botanical and computer graphics literature. Following their review (Section 2), we outline our modeling software environment, focusing on the language that we use to formally describe the algorithms and models (Section 3). We then develop the mathematical foundations of plant modeling based on positional information: the modeling and framing of individual axes (Section 4), and their partitioning into internodes (Section 5). In Section 6, we present the resulting modeling method from the modeler's perspective, and illustrate its applications using plants and plant structures organized along a single axis. In Section 7, we address the important special case of organ arrangement in closely packed spiral phyllotactic patterns. Finally, in Section 8, we extend the proposed modeling method to plants with higher-order branches, including trees. We conclude the paper with a discussion of the results, applications, and problems open for further research (Section 9). Proofs of selected mathematical results pertinent to the use of positional information in the modeling of plants are presented in the Appendices.

2 Previous work

Applications of positional information have their origins in early descriptive plant models created by biologists: the poplar model by Burk *et al.* [7] and the larch sapling model by Remphrey and Powell [30]. In both cases, the length of lateral branches was expressed as a function of their position on the main stem. The models were visualized as two-dimensional line drawings.

In computer graphics, a related concept was first applied to model trees by Reeves and Blau [29], who expressed the length of first-order branches as the distance from the branching point to a user-specified surface defining the silhouette of the tree. Higher-order branches were generated algorithmically, with "many parameters inherited from the parent."

A more elaborated model was introduced by Weber and Penn [36]. They characterized a tree using several positional functions, and pointed to an advantage of this technique: "Since our parameters can address the character of an entire stem and not just its segment-to-segment nature, we allow users to make changes on a level they can more easily understand and visualize."

Lintermann and Deussen incorporated positional information into their interactive plant modeling program `xfrog` [18, 19]. The position of a sample point along an axis may affect the length of an internode, the length of a branch, the magnitude of a branching angle, and other attributes. Functions that map positions to attribute values

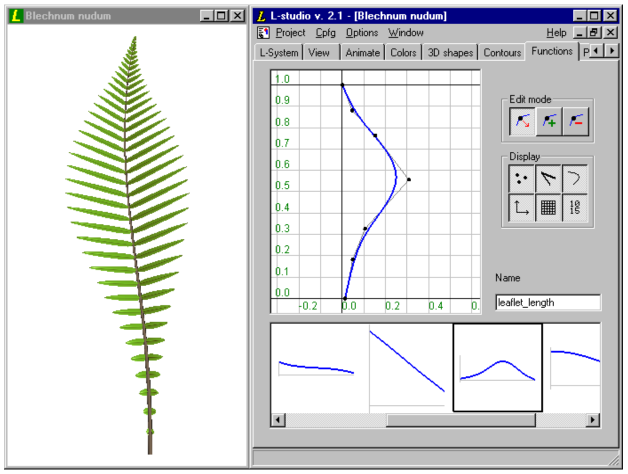

Figure 2: A snapshot of the L-studio/cpfg screen. The model can be manipulated using textual and graphical editors displayed on the right side of the screen. In this example, the outline of the fishbone water fern leaf (*Blechnum nudum*) is being defined using a graphical function editor. A *gallery* under the editor's window provides access to various functions used in this model. The second row of tabs near the top of the screen makes it possible to select other editors, such as the textual editor of the L-system that has been used to specify the algorithmic structure of this model.

can be specified graphically, by editing function plots, or textually, by editing algebraic expressions. The authors did not describe in detail the algorithms underlying their software, but experience with `xfrog` was an inspiration for our work. From the user interface perspective, the editing of function plots is an extension of the interactive manipulation of plant parameters using sliders [23, 28].

3 The modeling environment

We have adapted the L-system-based modeling software `L-studio/cpfg` [27] to the needs of modeling using positional information. A screenshot of the system in operation is shown in Figure 2.

An L-studio model consists of two basic components: a description of a generative algorithm in the `cpfg` modeling language [25], and a set of graphically defined entities. These entities can be defined and manipulated using the L-studio *function*, *curve*, *surface* and *material* editors [27], or imported from external sources.

The fundamental constructs of the `cpfg` language are *rewriting rules*, or *productions*. The program supports both parallel application of productions, characteristic of L-systems [28], and sequential application of productions, characteristic of Chomsky grammars [8]. In the context of plant modeling, these formalisms compare as follows.

L-system productions capture the *development* of plant components over time. For example, the division of a mother cell A into two daughter cells B and C can be described by the production $A \longrightarrow BC$. In the case of multicellular organisms, L-system productions are applied in *parallel* to advance time consistently in all cells. The simulation is completed when the organism reaches a predefined *terminal age*, corresponding to a given number of derivation steps.

Chomsky grammars, in contrast, characterize the *structure* of plants, that is, the distribution of their features and components in space. The fact that organism A consists of parts B and C can again be expressed by a production, for example $A \rightsquigarrow BC$, but such a *decomposition rule* has a different meaning and functions in a different way than its L-system counterpart. Since non-overlapping substructures can be partitioned independently from each other, the decomposition rules may be applied sequentially. Furthermore, the appropriate condition for terminating a decomposition process is the reaching of *terminal symbols*, which represent components that cannot be divided further.

Our intended use of positional information is to capture the distribution of plant features and components in space. Consequently, the meaning and formal properties of productions used in this paper correspond with the definition of Chomsky grammars. In the big picture of a complete plant modeling software design, the switch from L-systems to Chomsky grammars amounts to a relatively minor modification of the code. Consequently, our modeling language, outlined below, expands, rather than replaces, features of the earlier purely L-system-based implementations of the `cpfg` modeling language [13, 28].

As in the case of L-systems, a branching structure is represented by a *bracketed string* of *modules* (symbols with associated parameters). Matching pairs of brackets enclose branches. Derivation begins with an initial string identified by the keyword `axiom`. *Context-free* productions are specified using the syntax

$$pred : \{block_1\} \;\; cond \;\; \{block_2\} \rightsquigarrow succ, \tag{1}$$

where $pred$ is the predecessor (a single module), and $succ$ is the successor (a bracketed string of modules) [25]. The optional field $cond$ is the condition (logical expression) that guards production application. Fields $block_1$ and $block_2$ are sequences of C statements. The first block is executed before the evaluation of the condition. If the condition is true, the second block is also evaluated and the production is applied. For example, the rule

$$A(x) : \{y = x + 2;\} \;\; y \geq 5 \;\; \{z = y/3;\} \rightsquigarrow B(z)C(z+1) \tag{2}$$

can be applied to module A(4), subdividing it into modules $B(2)C(3)$.

The `cpfg` language also supports *context-sensitive* productions, in which the strict predecessor (module being replaced) $pred$ may be preceded by one or more modules constituting the *left context*, and/or followed by modules constituting the *right context*. These contexts are separated from the strict predecessor by symbols $<$ and $>$ respectively. For example, production

$$A(x) < B(y) > C(z) : x + z > 0 \rightsquigarrow M(y/2)N(y/2) \tag{3}$$

decomposes module B into a pair of modules M and N, provided that module B appears in the context of modules A and C, and the sum of their parameters is greater then 0. In the scope of this paper, context is limited to query symbols, discussed later on.

In order to conveniently specify morphogenetic gradients inherent in the use of positional information, we have extended the `cpfg` modeling language with function calls of the form $\mathrm{func}(id, x)$. The integer number id is the identifier of a planar B-spline curve and the real number x is the function argument. Function plots are manipulated using the *interactive function editor* (Figure 2). It constrains the motion of the control points that define the function plots to guarantee that they assign a unique value y to each argument x.

The modeling language also supports function calls of the form $\mathrm{curveX}(id, s)$, $\mathrm{curveY}(id, s)$, $\mathrm{curveZ}(id, s)$, and $\mathrm{tanX}(id, s)$, $\mathrm{tanY}(id, s)$, $\mathrm{tanZ}(id, s)$, where id is the identifier of an arbitrary B-spline curve. These calls return coordinates of a point on the curve id and of the tangent vector at this point, given the arc-length distance s from the curve origin. The call $\mathrm{curveLen}(id)$ returns the total length of the curve.

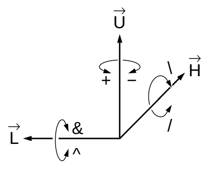

Figure 3: Controlling the turtle in three dimensions.

To create a graphical model, the derived string is scanned sequentially and reserved modules are interpreted as commands to a LOGO-style turtle [28]. At any point within the string, the *turtle state* is characterized by a position vector $\vec{P}$ and three mutually perpendicular orientation vectors $\vec{H}$, $\vec{L}$, and $\vec{U}$ that indicate the turtle's *heading*, the direction to the *left*, and the *up* direction (Figure 3). The coordinates of these vectors can be accessed using *query* modules of the form $?X(x, y, z)$, where X is the vector to be accessed, one of P, H, L, or U [26]. Module F causes the turtle to draw a line in the current direction, while modules f causes the turtle to move without drawing a line. Modules $+$, $-$, &, $\wedge$, $/$, and $\backslash$ rotate the turtle around one of the vectors $\vec{H}$, $\vec{L}$, or $\vec{U}$, as shown in Figure 3. Many symbols are overloaded; for example, $+$ and $-$ denote the modules that rotate the turtle as well as the usual arithmetic operations. The length of the line and the magnitude of the rotation angle can be given globally or specified as parameters of individual modules. Branches are created using a stack mechanism: the opening square bracket pushes the current state of the turtle on the stack, and the closing bracket restores it to the last saved state. Other interpreted symbols will be introduced with the sample models.

4 Modeling curved limbs

The shape of curved limbs, such as stems and elongated leaves, is "vital in capturing the character of a species" [37]. In computer graphics, this was first recognized by Bloomenthal [5], who applied *generalized cylinders* to model tree branches. A generalized cylinder is obtained by sweeping a planar *generating curve*, which determines the organ's *cross section*, along a *carrier curve* [16] that defines the organ's *axis*. The generating curve may be closed, as is typically the case for stems, or open, as for thin leaves, and it may change size and shape while being swept [33]. It also must be properly oriented with respect to the carrier curve. The *Frenet frame* [34], which is frequently used for this purpose, creates well known problems along straight sections of the carrier curve and at inflection points, where it is not defined. It also twists $180°$ in the proximity of the inflection points [6]. To avoid these problems, we propose an alternative solution based on the use of turtle geometry. This solution subsumes the Frenet frame, as well as the twist-minimizing *parallel transport frame* [3, 6, 14], as special cases. The turtle frame was previously used by Jirasek *et al.* [15] in the context of biomechanical modeling of plant branches.

The carrier curve is defined as a sequence of infinitesimal turtle movements. Let s denote the arc-length distance of the turtle from the origin of this curve. To define a smooth curve, we specify functions $\omega_H(s)$, $\omega_L(s)$ and $\omega_U(s)$ that characterize the *rates* of turtle's rotations around the axes $\vec{H}\vec{L}\vec{U}$ as the turtle moves (we use the term "rate of rotation" although s is a spatial coordinate and not time). The infinitesimal rotations $d\Omega_H$, $d\Omega_L$ and $d\Omega_U$ between curve points $\vec{P}(s)$ and $\vec{P}(s+ds)$ are then given by the equations:

$$d\Omega_H = \omega_H(s)ds, \quad d\Omega_L = \omega_L(s)ds, \quad d\Omega_U = \omega_U(s)ds. \qquad (4)$$

This specification yields a uniquely defined curve and moving reference frame (Appendix A.1). After replacing the infinitesimal increments ds by finite increments Δs, we obtain the following straightforward algorithm for modeling elongated plant organs:

Algorithm 1

```
1   #define $\ell$          1.0    /* total axis length */
2   #define $\mathcal{G}$          7      /* cross section ID */
3   #define $\Delta s$         0.02   /* turtle step */
4
5   #define $\omega_L(s)$      func(1,s)
6   #define $\omega_U(s)$      func(2,s)
7   #define $\omega_H(s)$      func(3,s)
8   #define $\phi(s)$       func(4,s)
9   #define width($s$)   func(5,s)
10
11  Axiom: @#($\mathcal{G}$) A(0,0)
12
13  A($s$,$\varphi$): $s \leq \ell$
14        { $\Delta\Omega_L = \omega_L(s)\Delta s$;
15          $\Delta\Omega_H = \omega_H(s)\Delta s$;
16          $\Delta\Omega_U = \omega_U(s)\Delta s$;
17            $\varphi = \varphi + \phi(s)\Delta s$; } $\rightsquigarrow$
18              +($\Delta\Omega_L$) &($\Delta\Omega_U$) /($\Delta\Omega_H$)
19              /($\varphi$) #(width($s$)) F($\Delta s$) \($\varphi$)
20              A($s + \Delta s$,$\varphi$)
21
22  A($s$,$\Phi$): $s > \ell \rightsquigarrow \epsilon$
```

Following the implementation of generalized cylinders in the `cpfg` program [20], the generating curve is selected by expression @#($\mathcal{G}$) in the axiom (line 11). The generalized cylinder is created recursively by the first production (lines 13-20) as a sequence of slices of length Δs. The cross section size is defined by module # with the parameter width(s) (line 19), and is linearly interpolated between points s and $s + \Delta s$. The angles of turtle rotation are calculated according to Equation 4 in lines 14–16, and applied to the turtle in line 18. The order of rotations represented by the symbols $+$, $\&$ and $/$ in line 18 is arbitrary, since infinitesimal rotations commute. Function $\phi(s)$ (line 17) rotates the generating curve around the cylinder axis without affecting the shape of the axis. This is convenient when defining twisted organs. The second production (line 22) removes the apex A at the end of cylinder generation, by replacing it with the empty symbol ϵ. Figure 4 shows sample leaves and stems generated by this Algorithm, with all functions specified using the interactive function editor (Section 3).

From the user's perspective, functions ω_L, ω_U, ω_H, ϕ and `width`, control bending, twist, and tapering of a generalized cylinder. Our experience confirms Barr's observation that such deformations are intuitive operations for modeling three-dimensional objects [1]. On the other hand, the user may prefer to specify the shape of an axis directly, for example as a spline curve. If this is the case, we *frame* it (*i.e.*, compute turtle's rotations $d\Omega_U$, $d\Omega_L$ and $d\Omega_H$) as follows.

Let $\vec{P}(s)$, $s \in [0,\ell]$, be a given smooth curve. Assume that it has been framed by a moving turtle; the turtle's heading vector $\vec{H}$ thus coincides with the tangent vector $\vec{T}$ to the curve for all $s \in [0,\ell]$. Denote by $\vec{H}\vec{L}\vec{U}$ the turtle orientation at point $\vec{P}(s)$ of this curve

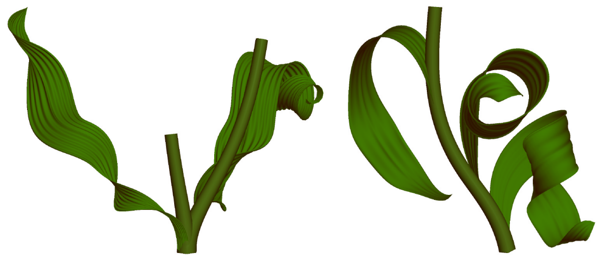

Figure 4: Leaves and stems of a herb lily (left) and tulip (right), modeled using Algorithm 1. The models are based on drawings in [38, pp. 56 and 58].

and by $\vec{H}' = \vec{H} + d\vec{H}$ the direction of the heading vector at point $\vec{P}(s+ds)$. Following [12], the infinitesimal rotation $d\vec{\Omega}$ that changes vector $\vec{H}$ to $\vec{H}'$ satisfies the equation $d\vec{H} = d\vec{\Omega} \times \vec{H}$, hence:

$$\begin{aligned} d\vec{H} = d\vec{\Omega} \times \vec{H} &= (\vec{U}d\Omega_U + \vec{L}d\Omega_L + \vec{H}d\Omega_H) \times \vec{H} && (5)\\ &= (\vec{U} \times \vec{H})d\Omega_U + (\vec{L} \times \vec{H})d\Omega_L + (\vec{H} \times \vec{H})d\Omega_H && (6)\\ &= \vec{L}d\Omega_U - \vec{U}d\Omega_L + 0d\Omega_H. && (7)\end{aligned}$$

By taking dot products of the first and last expression with vectors $\vec{L}$ and $\vec{U}$, we obtain:

$$\begin{aligned} d\vec{H} \cdot \vec{L} &= (\vec{H}' - \vec{H}) \cdot \vec{L} = \vec{H}' \cdot \vec{L} = d\Omega_U, && (8)\\ d\vec{H} \cdot \vec{U} &= (\vec{H}' - \vec{H}) \cdot \vec{U} = \vec{H}' \cdot \vec{U} = -d\Omega_L. && (9)\end{aligned}$$

By substituting $\vec{T}'$ for $\vec{H}'$ to emphasize that $\vec{T}'$ is a given tangent vector to the curve being framed, we obtain finally:

$$d\Omega_U = \vec{T}' \cdot \vec{L} \quad \text{and} \quad d\Omega_L = -\vec{T}' \cdot \vec{U}. \qquad (10)$$

Equations 10 constrain two rotational degrees of freedom. The third angle $d\Omega_H$ remains unconstrained, because it is multiplied by 0 in Equation 7. This implies that a moving turtle frame can be assigned to a given curve in different ways. In particular, if we set $\omega_H(s)$ in such a way that vector $\vec{L}$ (or $\vec{U}$) always lies in the osculating plane, we obtain the Frenet frame, and if $\omega_H(s) \equiv 0$, we obtain the parallel transport frame. We commonly use the latter, because it minimizes rotations of the reference frame around the axis of the generalized cylinder. The resulting algorithm for approximating and framing a given curve $\vec{P}(s)$ using a sequence of turtle motions is given below.

Algorithm 2

```
1   #define $\mathcal{P}$   1       /* curve ID */
2   #define $K$   57.29   /* radians to degrees */
3
4   Axiom: A(0) ?U(0,0,0) ?L(0,0,0)
5
6   A($s$) > ?U($u_x$,$u_y$,$u_z$) ?L($l_x$,$l_y$,$l_z$) : { $s' = s + \Delta s$ } $s' \leq \ell$
7         { $t'_x$ = tanX($\mathcal{P}$,$s'$);   $t'_y$ = tanY($\mathcal{P}$,$s'$);   $t'_z$ = tanZ($\mathcal{P}$,$s'$);
8           $\Delta\Omega_L = K * (t'_x l_x + t'_y l_y + t'_z l_z)$;
9           $\Delta\Omega_U = -K * (t'_x u_x + t'_y u_y + t'_z u_z)$; } $\rightsquigarrow$
10              +($\Delta\Omega_U$) &($\Delta\Omega_L$) F($\Delta s$) A($s'$)
```

The initial structure consists of apex A followed by query modules $?U$ and $?L$ (line 4). The parameter of the apex represents the current position of the turtle, measured as its arc-length distance from the origin of curve $\mathcal{P}$. The production (lines 6 to 10) creates an organ axis as a sequence of generalized cylinder slices of length Δs, as in Algorithm 1 (functions controlling the orientation and size of the generating curve have been omitted here for simplicity). Specifically, rotations $\Delta\Omega_U$ and $\Delta\Omega_L$ are calculated by multiplying (dot product) the vectors $\vec{U}$ and $\vec{L}$ (lines 8 and 9) returned by the query modules $?U$ and $?L$ (line 6) with the tangent vector to the curve $\mathcal{P}$ returned by the tanX, tanY and tanZ function calls (line 7). The values $\Delta\Omega_U$ and $\Delta\Omega_L$ orient the next segment of the curve, represented by module F(Δs) in line 10. House-keeping productions that erase modules A, $?U$ and $?L$ at the end of the derivation have been omitted from this listing.

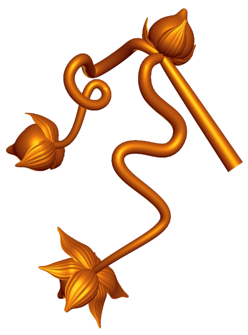

Figure 5: *Allum vineale* (field garlic), modeled using Algorithm 2 after the photograph in [4].

A sample application of Algorithm 2 is shown in Figure 5. Stems of a dry garlic plant have been modeled interactively, then framed using Algorithm 2 to orient the generating curve. Although the generating curve is circular in this case, its orientation is important for proper polygonization of the resulting generalized cylinders.

The turtle frame also plays an important role in orienting the organs and branches that are attached to an axis. Before discussing this in detail, we will consider the spacing of organs along an axis.

5 Organ spacing

We call points at which organs are attached to an axis the *nodes*, and the axis segments delimited by them the *internodes*. Let $\{s_i\}, i = 0, 1, \ldots$, be a sequence of node positions on an axis, and $\{l_i = s_{i+1} - s_i\}$ be the associated sequence of internode lengths (Figure 6a). It is straightforward to define the internode lengths using a function λ of the position of one of its incident nodes, for instance using the formula $l_i = s_{i+1} - s_i = \lambda(s_i)$. Unfortunately, with this definition function λ does not provide a robust control over the node distribution, because a small change in the position of the initial node s_0 may result in a totally different sequence of the nodes that follow. For example, if $s_0 = 0$, the function λ shown in Figure 6b will yield the sequence of node positions $\{s_i\} = 0, 1, 2, 3, \ldots$ (internode length equal to 1), but if $s'_0 = 0.25$, the sequence of node positions will be $\{s'_i\} = 0.25, 0.75, 1.25, 1.75, \ldots$ (internode length 0.5).

To achieve a more stable behavior, we observe that $1/\lambda(s)$ can be interpreted as the local *density* of nodes, in the sense that the integer part of the integral

$$N(s_o, s) = \int_{s_0}^{s} \frac{ds}{\lambda(s)} \tag{11}$$

represents the number of internodes between node s_0 and point s on the axis. Thus, given the initial node s_0, positions of the subsequent nodes correspond to the integer increments of the value of function N, that is, $N(s_o, s_{i+1}) = N(s_o, s_i) + 1$ (Figure 6c). The sequence of nodes $\{s_i\}$ defined this way is no longer critically sensitive to the initial node position s_0. Specifically, in Appendix A.2 we prove

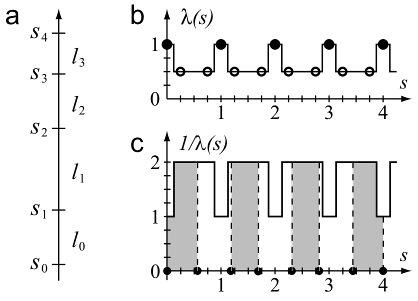

Figure 6: Partitioning an axis into segments. (a) The labeling of nodes and internodes. (b) Positional information represents the internode length. The same function $\lambda(s)$ generates very different node sequences (filled and empty circles), depending on the position of the initial node. (c) Positional information represents node density. Nodes are placed at the locations corresponding to the unit areas under the curve $1/\lambda(s)$. This definition leads to a more stable node spacing than (b).

that for any two node sequences $\{s_i\}$, $\{s'_i\}$ such that $s_0 < s'_0 < s_1$, the elements of both sequences interleave: $s_i < s'_i < s_{i+1}$ for all $i = 0, 1, 2, \ldots$.

Specification of node spacing based on Equation 11 also has other useful properties. First, if $\lambda(s)$ has a constant value l between nodes s_i and s_{i+1}, then l is equal to the internode length:

$$\int_{s_i}^{s_{i+1}} \frac{ds}{l} = 1 \text{ implies } s_{i+1} - s_i = l. \tag{12}$$

Second, if $\lambda(s)$ is a linear function, $\lambda(s) = as + b$, the length of consecutive internodes changes in a geometric sequence, $l_{i+1} = e^a l_i$ (proof in Appendix A.3). The ease of defining geometric sequences is important, because their approximations are often observed in nature (according to Niklas, they form the “null hypothesis” [22]).

The algorithm for placing nodes according to a given function $\lambda(s)$ is presented below.

Algorithm 3

```
1   Axiom: A(0,0)
2
3   A(s,a) : { $s' = s + \Delta s$ }  $s' \le \ell$
4       { $a' = a + \Delta s/\lambda(s)$ ;
5         if ($a' < 1$) { $flag = 0$; }
6         else { $a' = a' - 1$; $flag = 1$; } } $\rightsquigarrow$
7             F($\Delta s$) B($flag$) A($s'$,$a'$)
8
9   B($flag$) : $flag == 0 \rightsquigarrow \epsilon$
10  B($flag$) : $flag == 1 \rightsquigarrow$ @o
```

The initial structure consists of apex A (line 1). The first parameter represents the distance of the current point on the axis from the axis base, as in Algorithms 1 and 2. The second parameter represents the fractional part of the integral $N(0, s)$ given by Equation 11. The production in lines 3 to 7 creates the axis as a sequence of segments F of length Δs, separated by markers of potential node locations B. If the $flag$ is zero, module B is subsequently erased (line 9). When a exceeds 1, the $flag$ is set (line 6) to produce a node marked by symbols @o (line 10).

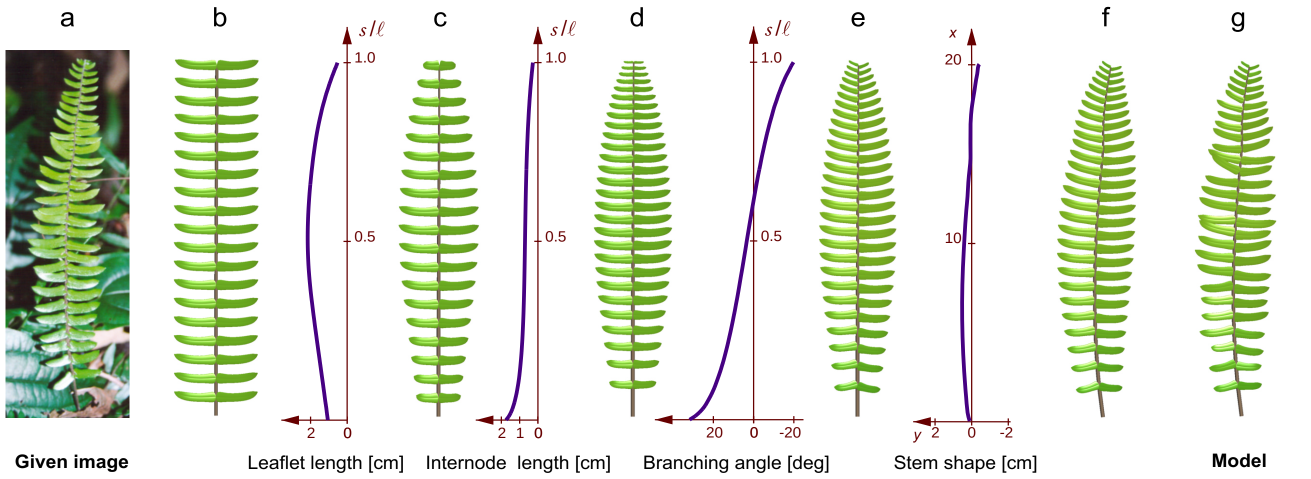

Figure 7: Using positional information to model a *Pellaea falcata* (sickle fern) leaf.

6 Modeling single-compound plant structures

We have combined the methods for framing and partitioning an axis into the following algorithm, which makes it possible to model a variety of *single-compound* structures (sequences of organs supported by a single stem). Definitions of graphical functions and constants used in previous algorithms have not been included. Secondary features, such as the randomization of values returned by functions, have also been omitted.

Algorithm 4

```
1   #define Φ    0         /* phyllotactic angle */
2
3   Axiom: A(0,0,0) ?U(0,0,0) ?L(0,0,0)
4
5   A($s$,$a$,$\varphi$) > ?U($u_x$,$u_y$,$u_z$) ?L($l_x$,$l_y$,$l_z$) :
6               { $s' = s + \Delta s$ } $s' \leq \ell$
7       { $t'_x$ = tanX($\mathcal{P}$,$s'$); $t'_y$ = tanY($\mathcal{P}$,$s'$); $t'_z$ = tanZ($\mathcal{P}$,$s'$);
8         $\Delta\Omega_L = K * (t'_x l_x + t'_y l_y + t'_z l_z)$;
9         $\Delta\Omega_U = -K * (t'_x u_x + t'_y u_y + t'_z u_z)$;
10          $a$   $= a + \Delta s / \lambda(s)$
11        if ($a < 1$) { $flag = 0$; }
12        else { $a = a - 1$; $flag = 1$; $\varphi = \varphi + \Phi$; } } $\rightsquigarrow$
13              +($\Delta\Omega_U$) &($\Delta\Omega_L$) #(stem_width($s$))
14              F($\Delta s$)B($s$,$\varphi$,$flag$) A($s'$,$a$,$\varphi$)
15
16  B($s$,$\varphi$,$flag$) : $flag == 0 \rightsquigarrow \epsilon$
17  B($s$,$\varphi$,$flag$) : $flag == 1$
18      { $l$ = length($s$); $w$ = width($s$); } $\rightsquigarrow$
19              [ /($\varphi$) [ +(brangle($s$)) ~L($l$,$w$) ]
20                    [ −(brangle($s$)) ~L($l$,$w$) ] ]
```

The key new element is the third production (lines 17 to 20), which inserts a pair of organs at the node. The organs are defined as instances of a predefined surface L, with the length, width and angle of insertion determined by functions of position s.

In order to present the operation of Algorithm 4 from a user's perspective, let us consider the process of modeling a *Pellaea falcata* (sickle fern) leaf. The photograph of the target structure is shown in Figure 7a. Construction begins with a generic single-compound (pinnate) leaf (b), which is generated when all graphically defined functions are set to their default constant values. The length of the leaflets is then modified as a function of their position on the stem (c). Since the leaf silhouette is determined by the extent of its component leaflets, this function controls the overall leaf shape. The next two functions define the lengths of the internodes (d) and the values of the branching angles between the stem and the leaflets (e). The stem shape is then established by manipulating a parametric curve (f). Finally, the branching angles and the leaflet lengths are randomized to capture the unorganized variation present in the original leaf (g). The model also makes use of functions that have not been shown in Figure 7, which define the taper of the stem and the width of the leaflets.

In the above example, the individual leaflets have been modeled as predefined surfaces L, scaled in length and width using functions of their position on the stem (lines 18 to 20 in Algorithm 4). Leaves, petals and similar organs can also be modeled as generalized cylinders with Algorithm 1. We use this technique in most

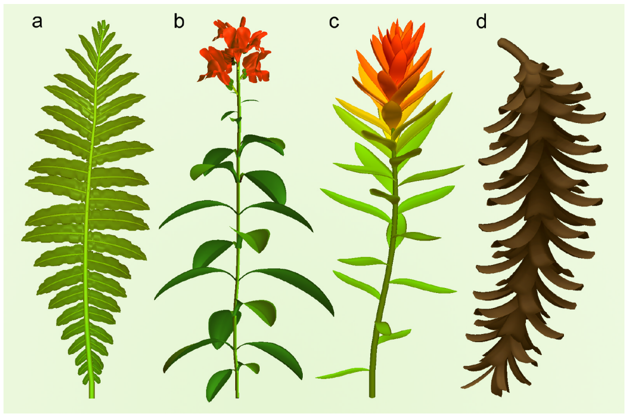

Figure 8: Plants and plant organs with different phyllotactic patterns: (a) *Blechnum gibbum* leaf with the distichous arrangement of leaflets, (b) *Antirrhinum majus* (snapdragon) plant with a decussate arrangement of leaves, (c,d) *Casilleja coccinea* (Indian paintbrush) plant and *Pinus strobus* (white pine) cone with spiral arrangements of leaves and scales.

models, because it allows us to define and manipulate organ shapes more easily. For example, the rippled surface of the *Blechnum gibbum* leaflets (Figure 8a) was obtained by randomly changing the shape, size and orientation of the generating curve.

Constant Φ in Algorithm 4 controls *phyllotaxis*, or the arrangement of organs around the stem [28]. If $\Phi = 0$, organs are arranged in a planar *distichous* pattern, as in Figures 7 and 8a. If $\Phi = 90°$, consecutive pairs of organs are issued in mutually perpendicular planes, forming a *decussate* pattern (Figure 8b). Finally, if $\Phi = 137.5°$ (the golden angle), and only one organ is attached to each node (line 20 of Algorithm 4 is removed), a *spiral* phyllotactic pattern results (Figure 8c and d). Thus, a change in a single constant extends Algorithm 4 to three dimensions.

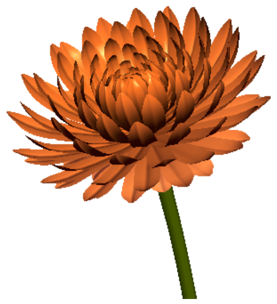
Figure 9: *Helichrysum bracteatum* (strawflower).

A distinctive feature of *Helichrysum bracteatum* (a strawflower, Figure 9) is the posture of petals (ray florets), which are more curved near the center of the flower head than on the outside. To capture this gradient, the position of the petals on the main axis of the flower head was used to interpolate between two curves that describe the extreme postures of the petals. A similar technique made it possible to control the shape of leaves and petals in the beargrass model (Figure 10). Photographs of the inflorescences that we used as a reference to construct this model are shown in Figure 11.

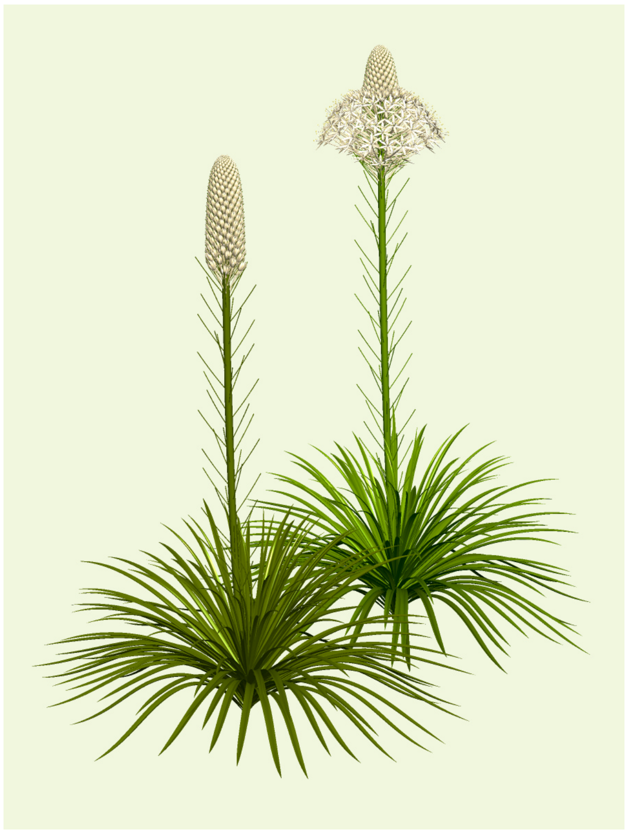
Figure 10: Model of *Xerophyllum tenax* (beargrass).

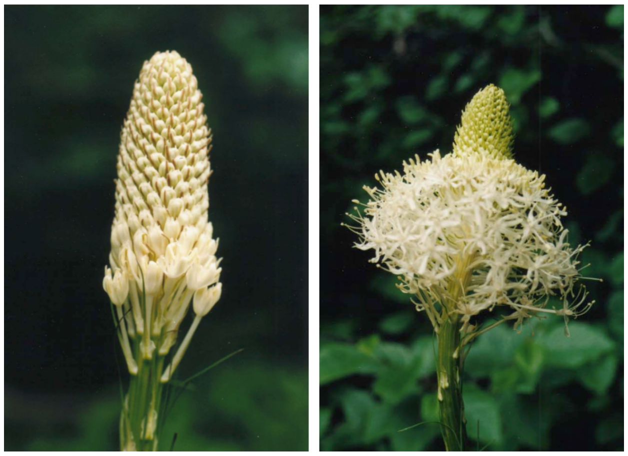
Figure 11: Photographs of *Xerophyllum tenax* inflorescences.

7 Compact phyllotactic patterns

In spiral phyllotactic patterns, the individual organs, *e.g.* petals, florets, or scales, are often densely packed on their supporting surface (the *receptacle*), as illustrated by the model of beargrass. Modeling such patterns using Algorithm 4 requires a coordinated manipulation of the radius of the receptacle, the size of the organs being placed, and their vertical displacement (corresponding to the internode length). In this section we facilitate the modeling process by relating the vertical displacement to the radius of the supporting surface and the size of organs. Both the radius and the organ size can be defined as functions of organ position on the receptacle, making it possible to capture a wide range of forms and patterns. The proposed model has the same generative power as the *collision-based* model of phyllotaxis introduced by Fowler *et al.* [11], but operates faster because it avoids the explicit detection of collisions between organs.

Vogel [35] provided the first mathematical description of phyllotactic patterns used for computer graphics purposes [28]. His model places equally sized organs on the surface of a flat disk, stating that the n-th organ will have polar coordinates:

$$\phi = n \cdot 137.5°, \quad r = c\sqrt{n}, n = 1, 2, \ldots \qquad (13)$$

where c is a constant. The angular displacement of $137.5°$ between consecutive organs is treated as empirical data, reproduced but not explained by the model. The formula for the radial displacement r is justified by two observations: (a) since organs are placed from the disk center outwards, the ordering number n of the organ placed at a distance r from the center is equal to the total number of organs that occupy a disk of radius r, and (b) if all organs occupy the same area, the total number n of organs in a disk of radius r will be proportional to r^2, hence $r = c\sqrt{n}$.

Vogel's model abstracts from the shape of organs and places them in a disk according to the area they occupy. Lintermann and Deussen proposed a similar approximation to derive a formula for placing organs on the surface of a sphere [19]. Both approaches are subsumed by the model of Ridley [31], which operates on arbitrary surfaces of revolution. Our algorithm is based on Ridley's analysis.

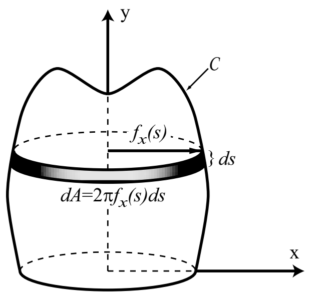

Figure 12: A receptacle.

Let $(f_x(s), f_y(s))$, $s \in [0, L]$ be a parametric definition of a planar curve $\mathcal{C}$ that generates the receptacle when rotated around the y axis of the coordinate system (Figure 12). We assume natural parameterization of the curve $\mathcal{C}$, which means that parameter s is the arc-length distance of point $(f_x(s), f_y(s))$ from the origin of this curve. The area dA of the infinitesimal slice of the receptacle generated by the arc $[s, s + ds]$ is then equal to $2\pi f_x(s)ds$ (Figure 12). We denote by $\pi\rho^2(s)$ the area occupied by an organ placed on the receptacle at a distance s from the origin of the generating curve $\mathcal{C}$. As in the case of partitioning an axis into internodes (Section 5), we can interpret $1/\pi\rho^2(s)$ as the organ density at s. The integer part of the integral

$$N(0, s) = \int_0^s \frac{2\pi f_x(s)}{\pi\rho^2(s)} ds = \int_0^s \frac{2 f_x(s)}{\rho^2(s)} ds \qquad (14)$$

is then equal to the total number of organs placed in the portion $[0, s]$ of the receptacle. Consecutive organs are placed at locations that increment $N(0, s)$ by one. This leads to the following algorithm:

Algorithm 5

```
1   #define $\mathcal{C}$      1              /* generating curve ID */
2   #define $\ell$      curveLen($\mathcal{C}$)   /* length of curve $\mathcal{C}$ */
3   #define $\rho(s)$   func(2,s)      /* density function */
4   #define $\Delta s$     0.001          /* integration step */
5
6   Axiom: A(0,0)
7
8   A($s$,$a$) : $s < \ell$
9       { while( $a < 1$ && $s < \ell$)
10          { $x = $ curveX($\mathcal{C}, s$);
11            $a = a + (2x/\rho^2(s))\Delta s$;
12            $s = s + \Delta s$;
13          }
14        $a = a - 1$; $y = $ curveY($\mathcal{C}, s$);
15      }
16      $\rightsquigarrow$ [f($y$)-(90)f($x$)˜O($\rho(s)$)] \(137.5) A($s$,$a$)
```

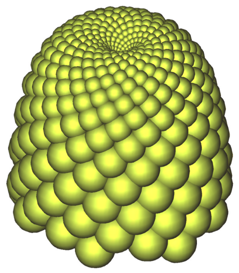

Figure 13: Example of a compact phyllotactic pattern generated using Algorithm 5.

The first parameter of module A represents the arc-length distance s of the current point from the base of the receptacle. The second parameter is the fractional part a of the integral $N(0, s)$ (Equation 14). The integration is performed incrementally by the *while* loop inside the production (lines 9 to 13). When the integral reaches 1, an organ O of radius $\rho(s)$ is placed at height y and distance x from the receptacle axis y (line 16). Consecutive organs are rotated with respect to each other by the golden angle $137.5°$ measured around this axis. A sample pattern generated by Algorithm 5 is shown in Figure 13.

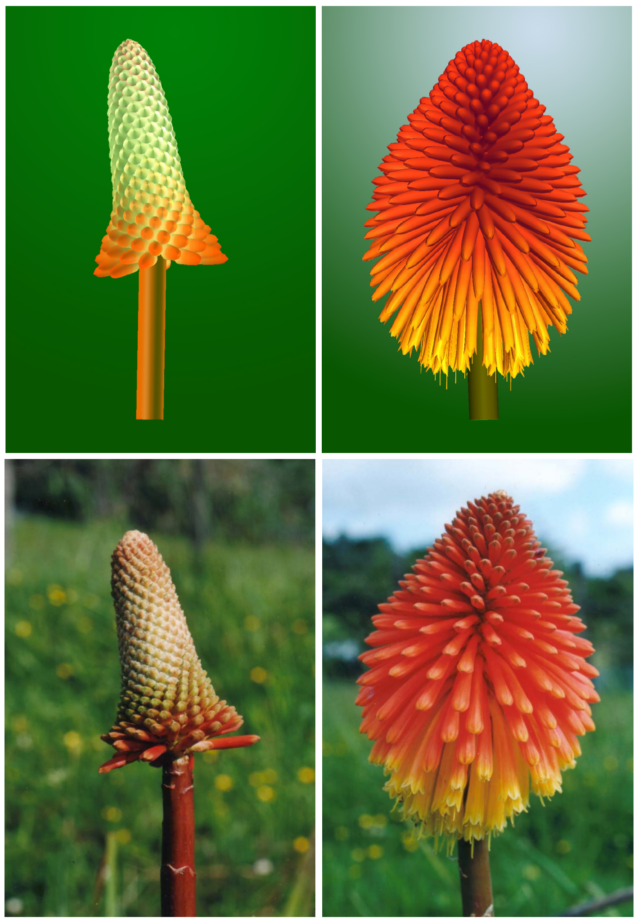

Figure 14: Inflorescences of *Kniphofia* sp. (red-hot poker plant) generated using Algorithm 5: models of two developmental stages (top) and the photographs used as a reference (bottom).

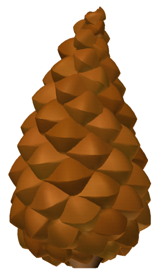

Figure 15: A *Pinus banksiana* (Jack pine) cone.

In realistic models, we replace spheres O by models of plant organs, as in [11]. For example, Figure 14 shows two developmental stages of the inflorescence of *Kniphofia* sp. (red-hot poker plant), in which florets have been modeled using generalized cylinders. In Figure 15 the algorithm has been additionally modified to allow for a curved cone axis. This modification is equivalent to the deformation of a straight cone, performed as a post-processing step.

8 Modeling multiple-compound structures

Algorithms 4 and 5, introduced in the previous sections, have been illustrated using examples of single-compound monopodial structures, each consisting of a sequence of organs placed along an axis or on a receptacle. The same algorithms can also be used, how-

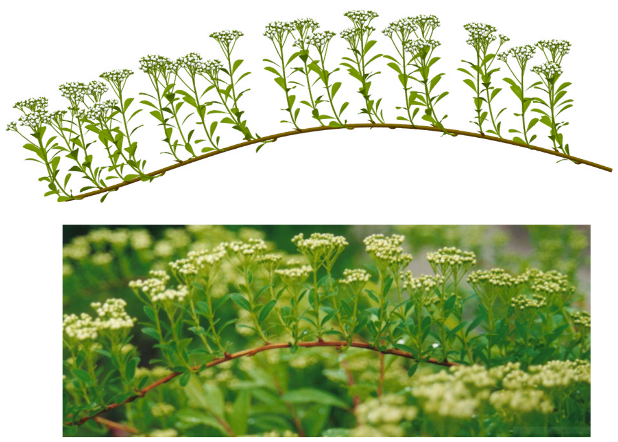

Figure 16: A photograph and a model of a *Spiraea* sp. twig. The arrangement of shoots on the twig and the arrangement of leaves and flowers in each shoot follow the spiral phyllotactic pattern. The approximately vertical posture of all shoots reflects strong orthotropism, which has been simulated by biasing the turtle's heading vector in the vertical direction as described in [28, page 58].

ever, to generate structures in which the main axis supports entire substructures. For example, the *Spiraea* sp. twig shown in Figure 16 was constructed using Algorithm 4 twice: first to place the flower-bearing shoots along the main stem, then to place the leaves and the flowers within each shoot. In this case, all shoots have been assumed equal, except for the different shoot axis shapes caused by their *orthotropism* (tendency to grow vertically). In general, however, the supported structures may vary in a systematic manner, reflecting a morphogenetic gradient along the main stem.

To capture this gradient, we assume that, given two branches of the same order, the shorter branch is identical (up to the effects of tropisms and random variation) to the *top* portion of the longer branch. This concept of *branch mapping* is supported by both biological arguments and simulation results.

Biologically, it is related to the fact that apical meristems, the main engines of plant development, are located at the distal ends of branches. Thus, if branch $\mathcal{B}$ develops over a shorter time or at a slower rate than an otherwise equivalent branch $\mathcal{A}$, branch $\mathcal{B}$ will resemble the *top* portion of $\mathcal{A}$.

A modeling example supporting the use of branch mapping is shown in Figure 17. An inflorescence of common lilac *Syringa vulgaris* (a) has been measured and reconstructed at three levels of accuracy: with all architectural information present (b), using the assumption that all flowers are identical (c), and using the assumption that shorter branches are identical to the top portions of the longer branches of the same order (d). Although reconstruction (d) is visually the least accurate, it still matches the real structure well.

Branch mapping makes it possible to define all branches of the same order using one set of functions. This concept is captured by the following algorithm.

Algorithm 6

```
1   Axiom: A(0,0)
2
3   A($o$,$s$) : $o < \mathrm{MAX}$ && $s < \mathrm{max\_len}[o]$
4            { $rel = s/\mathrm{max\_len}[o]$; } $\rightsquigarrow$
5        #(int_width($o$,$rel$)) F(int_len($o$,$rel$))
6        [+(branch_ang($o$,$rel$))
7             A($o+1$,max_len[$o+1$] - branch_len($o$,$rel$)) ]
8        [$-$(branch_ang($o$,$rel$))
9             A($o+1$,max_len[$o+1$] - branch_len($o$,$rel$)) ]
10       /(90) A($o$,$s$+int_len($o, rel$))
11
12  A($o$,$s$) : $s \geq \mathrm{max\_len}[o] \rightsquigarrow$ ~K
```

Algorithm 6 can be viewed as a recursive version of Algorithm 4, with the mechanism for creating curved axes removed for simplicity, and the internode length determined using point-sampled positional information as in Figure 6b for the same reason. Parameters o and s of the apices A represent the axis order and position along this axis, respectively. The array max_len$[o]$ specifies the length ℓ_{max} of the longest axis of each order $o < \mathrm{MAX}$. This value is used to represent positional information in relative terms, as a fraction rel of ℓ_{max} (line 4). This facilitates the specification of all functions, since they have fixed domain $[0, 1]$. Functions int_width(o, rel), int_len(o, rel), branch_ang(o, rel) and branch_len(o, rel) characterize morphogenetic gradients: the width and length of internodes, the branching angles at which the child branches are inserted, and the length of these child branches. All axes of the same order share the same set of functions. Within an axis of length ℓ, parameter s ranges from the initial value of $\ell_{max} - \ell$ (assigned to the newly created apices A in lines 7 and 9) to the maximum value of ℓ_{max} (condition in line 3). Thus, morphogenetic gradients along shorter axes are aligned with the distal portion of the longest axis of the same order, as required for branch mapping. Predefined flowers K are placed at the ends of the branches (line 12).

Examples of lilac inflorescences generated by Algorithm 6 are shown in Figure 18. Lilac inflorescences have decussate phyllotaxis. As was the case for Algorithm 4, a small modification of Algorithm 6 makes it possible to generate structures with spiral phyllotaxis. An example of the resulting structure — the inflorescence of an *Astilbe plant* — is shown in Figure 19.

Algorithm 6 can also be applied to approximate trees with clearly delineated branch axes (many young trees satisfy this criterion). If the axes of first-order branches are approximately straight and higher-order branches are relatively short, the outline of the tree crown is determined by the extent of the first-order branches and

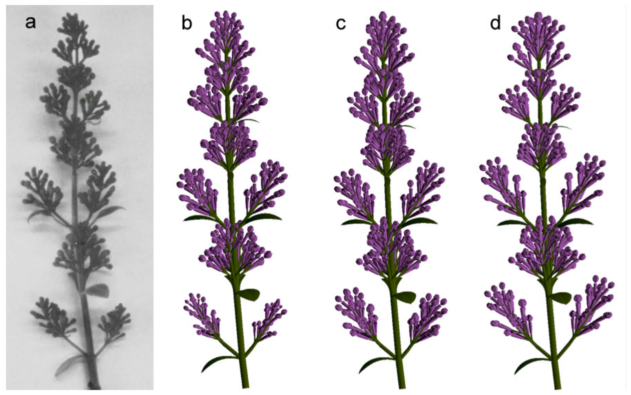

Figure 17: The effect of branch mapping. (a) An inflorescence of common lilac *Syringa vulgaris*. (b) Reconstruction of this inflorescence based on the measurements of all branches and flowers. (c) The same structure, all flowers assumed to be identical. (d) An approximate reconstruction based on branch mapping.

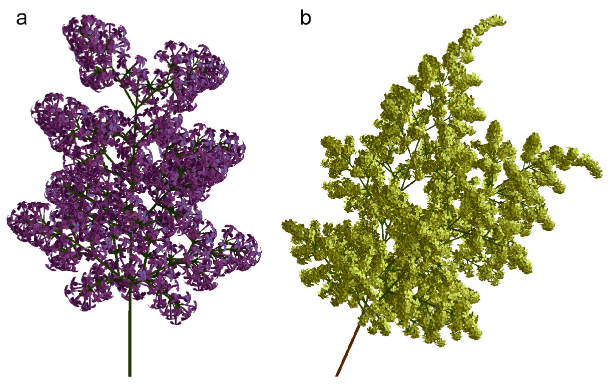

Figure 18: Inflorescences of two lilac species modeled using Algorithm 6: (a) *Syringa chinensis* CV. Rubra and (b) *Syringa reticulata*.

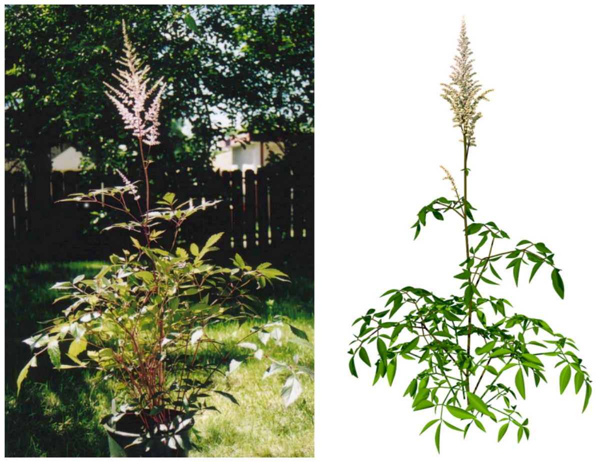

Figure 19: A photograph and a model of an *Astilbe x arendsii* CV. Diamant plant.

can easily be controlled by function branch_len$(0, rel)$ (Figure 20). In this sense, the use of positional information addresses the problem of progressing from silhouette to detail in the modeling process, exemplified by Figure 1c.

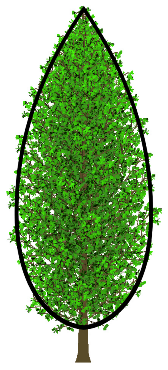

Figure 20: A generic tree model and its silhouette specfciation.

The problem of generating trees given their silhouettes occurs in several applications. One of them is the modeling and rendering of plant ecosystems. According to the approach proposed by Deussen *et al.* [10], the complexity of ecosystem modeling can be addressed by performing an individual-based simulation of the whole ecosystem, then replacing the coarse plant models used in this simulation with their detailed counterparts. The modeling method described in the present paper provides a means of creating plant models that match silhouettes determined at the ecosystem level (Figure 21).

9 Conclusions

We have explored the idea of plant modeling with functions that relate features of a plant to their positions along plant axes. Our experience confirms previous observations that this use of positional information is intuitive and well suited to the interactive modeling of plants. Visually important aspects of plant appearance — posture, the arrangement of components, and the overall silhouette — can easily be captured and controlled, while the procedural approach removes the tedium of specifying and placing each plant component individually. The algorithms are sufficiently fast to support interactive plant modeling on current personal computers.

We demonstrated the power of the modeling with positional information by recreating the form of several plants found in nature, presented on photographs, or depicted in drawings. The modeled structures range from individual leaves to compound herbaceous plants and trees.

The use of positional information is not limited to interactive modeling applications. We showed this by incorporating detailed tree models into a plant ecosystem model that only provided coarse characteristics of tree silhouettes. A related potential application is the automatic generation of plant models that match silhouettes of real trees, given their photographs [32].

At the technical level, our paper contributes: (a) a conceptual distinction between L-systems and Chomsky grammars as formal bases of developmental and structural plant models; (b) a generalized method for framing plant axes, free of the artifacts of the Frenet frame; (c) a robust method for spacing organs along plant axes; (d) an analytic method for generating phyllotactic patterns on arbitrary surfaces of revolution, based on Ridley's model; (e) the notion of branch mapping and its application to the modeling of compound plant structures; and (f) an example of the modeling system that integrates all of these concepts.

One open research problem is the use of constraints. In Algorithm 5 we introduced a relation between organ size and available space to constrain organ position in phyllotactic patterns. Many other relations have also been identified by biologists and can be applied to plant modeling [17]. By incorporating them into the algorithms we may further facilitate the modeling process. Specifically, constraints may reduce the number of parameters and functions that must be specified explicitly, while enforcing biological plausibility of the resulting structures.

Another interesting problem falls in the domain of interactive modeling techniques. In the present implementation, the user manipulates function plots, curves, and surfaces that are displayed separately from the model. A direct manipulation interface, in which the user would interact with the modeled structure itself, may lead to an even more intuitive modeling process.

A Appendices

A.1 Fundamental theorem of differential turtle geometry

The method for modeling curved limbs presented in Section 4 is based on the following extension of the fundamental theorem of differential geometry for three-dimensional curves [34, page 61] to the turtle reference frame.

Theorem. Let $\vec{H}(s)\vec{L}(s)\vec{U}(s)$ denote a moving reference frame defined on an interval $[0, \ell]$. Furthermore, let $\vec{H}(0)\vec{L}(0)\vec{U}(0)$ be the

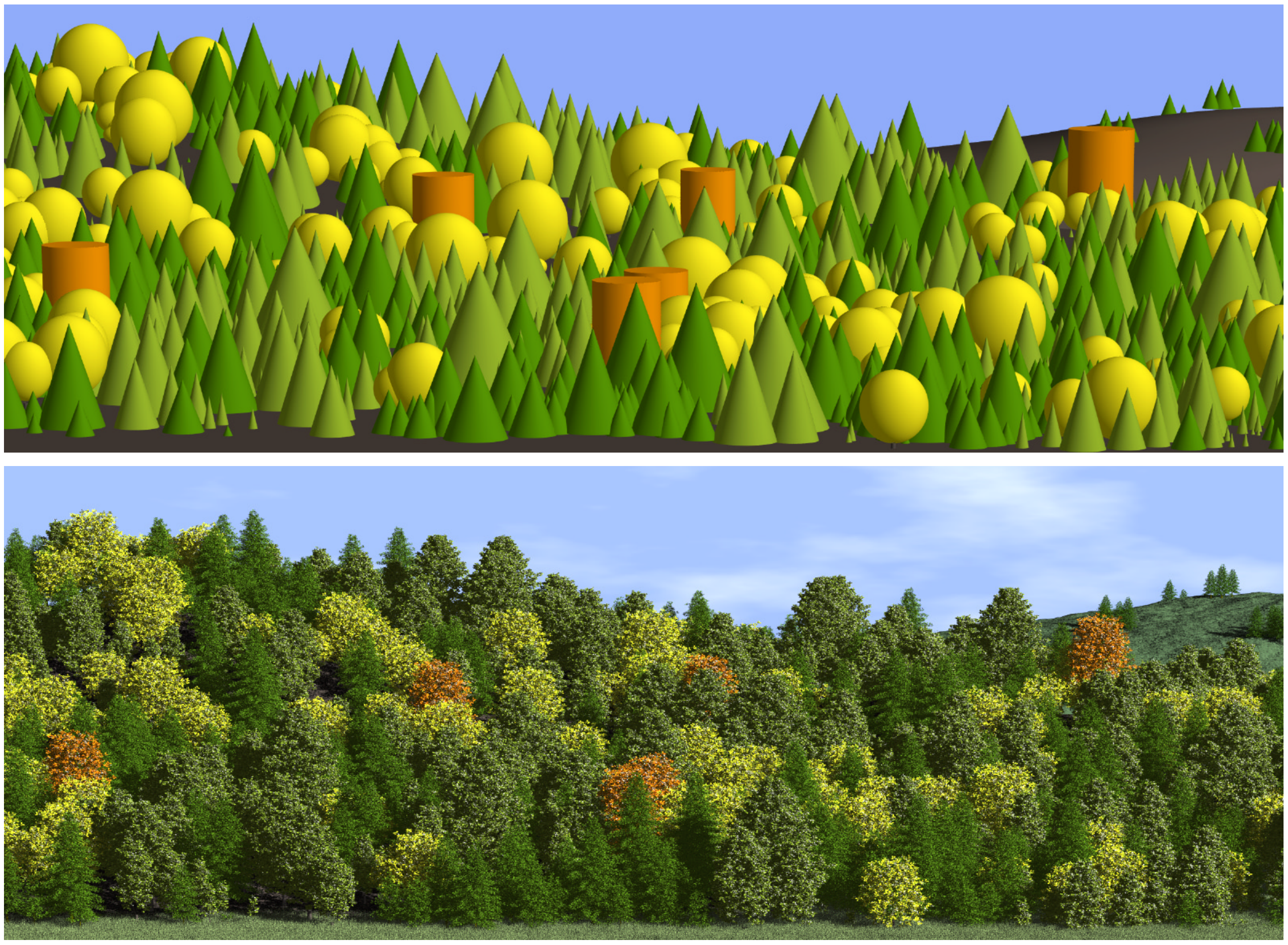

Figure 21: Visualization of an ecosystem simulation. Top: direct visualization. Bottom: realistic visualization. Tree silhouettes match the shapes coarsely defined at the ecosystem level.

initial orientation of this frame, and differentiable functions $\omega_H(s)$, $\omega_L(s)$ and $\omega_U(s)$ be its rates of rotation around the axes $\vec{H}(s)$, $\vec{L}(s)$ and $\vec{U}(s)$. The orientation of this frame is then uniquely defined for all $s \in [0, \ell]$. Moreover, given the initial frame position $\vec{P}(0)$, there is a unique differentiable curve $\vec{P}(s)$ for which s is the natural (arc-length) parameter, such that $\vec{H}(s)$ is tangent to $\vec{P}(s)$ for all $s \in [0, \ell]$.

Proof. Following [12], an infinitesimal rotation vector $d\vec{\Omega}$ acting on an arbitrary vector $\vec{A}$ changes it by $d\vec{A} = d\vec{\Omega} \times \vec{A}$. Thus, changes of the $\vec{H}\vec{L}\vec{U}$ reference frame due to the rotation rate vector $\vec{\omega} = \omega_H \vec{H} + \omega_L \vec{L} + \omega_U \vec{U}$ satisfy the system of equations:

$$\frac{d\vec{H}}{ds} = \vec{\omega} \times \vec{H}, \quad \frac{d\vec{L}}{ds} = \vec{\omega} \times \vec{L}, \quad \frac{d\vec{U}}{ds} = \vec{\omega} \times \vec{U}. \tag{15}$$

Given the initial frame orientation $\vec{H}(0)\vec{L}(0)\vec{U}(0)$, vectors $\vec{H}(s)$, $\vec{L}(s)$ and $\vec{U}(s)$ are thus the unique solution to the initial value problem for the system of differential equations (15) in the interval $[0, \ell]$. Moreover, curve $\vec{P}(s)$ is given by the integral:

$$\vec{P}(s) = \vec{P}(0) + \int_0^s \vec{H}(s)ds. \quad \square \tag{16}$$

A.2 Stability of node distribution

The fact that the distribution of nodes defined by integer values of Equation 11 does not depend critically on the choice of the initial node can be formally stated as follows.

Theorem. Consider a function λ such that $\lambda(s) > 0$ for all $s > 0$, and let $s_o, s'_0 > 0$ be two numbers. Using function N specified by Equation 11, define sequences $\{s_i\}$ and $\{s'_i\}$ such that $s_{i+1} = N(s_0, s_i) + 1$ and $s'_{i+1} = N(s'_0, s'_i) + 1$ for all $i = 0, 1, 2, \ldots$. If $s_0 < s'_0 < s_1$ then $s_i < s'_i < s_{i+1}$ for all $i = 0, 1, 2, \ldots$.

Proof by induction on i. The assumption $\lambda(s) > 0$ implies that $F(s) \equiv N(s_0, s)$ is an increasing function of the argument s. Thus, $s_i < s'_i < s_{i+1}$ implies $F(s_i) < F(s'_i) < F(s_{i+1})$, and therefore $F(s_i)+1 < F(s'_i)+1 < F(s_{i+1})+1$. By substituting $F(s_i)+1 = F(s_{i+1})$, $F(s'_i) + 1 = F(s'_{i+1})$, and $F(s_{i+1}) + 1 = F(s_{i+2})$, we obtain $F(s_{i+1}) < F(s'_{i+1}) < F(s_{i+2})$, hence $s_{i+1} < s'_{i+1} < s_{i+2}$. $\square$

A.3 Distribution of nodes defined by a linear function λ.

Theorem. Consider the sequence of nodes s_i defined by integer values of Equation 11, and let $\lambda(s) = as + b$. The length of consecutive internodes $l_i = s_{i+1} - s_i$ satisfies the equation $l_{i+1} = e^a l_i$ for $i = 0, 1, 2, \ldots$.

Proof. From Equation 11 we obtain:

$$\begin{aligned} 1 &= N(s_0, s_{i+1}) - N(s_0, s_i) && (17) \\ &= \int_{s_i}^{s_{i+1}} \frac{ds}{as+b} = \frac{1}{a} \ln \frac{as_{i+1}+b}{as_i+b} \,. && (18) \end{aligned}$$

Thus, $as_{i+1} + b = e^a(as_i + b)$ and, similarly, $as_{i+2} + b = e^a(as_{i+1}+b)$. By subtracting these equations sidewise and dividing by a we obtain $s_{i+2} - s_{i+1} = e^a(s_{i+1} - s_i)$, or $l_{i+1} = e^a l_i$. □.

Acknowledgments

We would like to thank: Lynn Mercer for contributing Figures 1b and c, Josh Barron for Figure 8a, Laura Marik for Figure 15, Enrico Coen for joint work on the snapdragon model (Figure 8b), Campbell Davidson for joint work on the lilac models (Figure 18), Christophe Godin for joint work on the decomposition rules, Bernd Lintermann and Oliver Deussen for a detailed demo of `xfrog`, and the referees for their insightful comments. The support of the Natural Sciences and Engineering Research Council of Canada, the International Council for Canadian Studies, the Alberta MACI project and the University of Calgary is gratefully acknowledged.

References

[1] A. H. Barr. Global and Local Deformations of Solid Primitives. Proceedings of SIGGRAPH 84, in *Computer Graphics*, 18, 3, July 1984, pages 21–30.

[2] D. Barthélémy, Y. Caraglio, and E. Costes. Architecture, Gradients Morphogénétiques et Age Physiologique ches les Végétaux. In J. Bouchon, Ph. De Reffye, and D. Barthélémy, editors, *Modélisation et Simulation de l'Architecture des Végétaux*, pages 89–136. INRA Editions, Paris, 1997.

[3] R. L. Bishop. There Is More Than One Way to Frame a Curve. *Amer. Math. Monthly*, 82(3):246–251, March 1975.

[4] H. Bjornson. *Weeds*. Chronicle Books, San Francisco, 2000.

[5] J. Bloomenthal. Modeling the Mighty Maple. Proceedings of SIGGRAPH 85, in *Computer Graphics*, 19, 3, July 1985, pages 305–311.

[6] J. Bloomenthal. Calculation of Reference Frames Along a Space Curve. In A. Glassner, editor, *Graphics Gems*, pages 567–571. Academic Press, Boston, 1990.

[7] T. E. Burk, N. D. Nelson, and J. G. Isebrands. Crown Architecture of Short-rotation, Intensively Cultured *Populus*. III. A Model of First-order Branch Architecture. *Canadian Journal of Forestry Research*, 13:1107–1116, 1983.

[8] N. Chomsky. Three Models for the Description of Language. *IRE Trans. on Information Theory*, 2(3):113–124, 1956.

[9] P. de Reffye, C. Edelin, J. Françon, M. Jaeger, and C. Puech. Plant Models Faithful to Botanical Structure and Development. Proceedings of SIGGRAPH 88, in *Computer Graphics* 22, 4, August 1988, pages 151–158.

[10] O. Deussen, P. Hanrahan, B. Lintermann, R. Měch, M. Pharr, and P. Prusinkiewicz. Realistic Modeling and Rendering of Plant Ecosystems. Proceedings of SIGGRAPH 98, Annual Conference Series, July, 1998, pages 275–286.

[11] D. R. Fowler, P. Prusinkiewicz, and J. Battjes. A Collision-based Model of Spiral Phyllotaxis. Proceedings of SIGGRAPH 92, in *Computer Graphics*, 26, 2, July 1992, pages 361–368.

[12] H. Goldstein. *Classical Mechanics*. Addison-Wesley, Reading, 1980.

[13] J. S. Hanan. *Parametric L-systems and Their Application to the Modelling and Visualization of Plants*. PhD thesis, University of Regina, June 1992.

[14] A. J. Hanson. Quaternion Gauss Maps and Optimal Framings of Curves and Surfaces. Technical Report 518, Computer Science Department, Indiana University, Bloomington, IN, 1998.

[15] C. Jirasek, P. Prusinkiewicz, and B. Moulia. Integrating Biomechanics into Developmental Plant Models Expressed Using L-systems. In H.-Ch. Spatz and T. Speck, editors, *Plant Biomechanics 2000*, pages 615–624. Georg Thieme Verlag, Stuttgart, 2000.

[16] J. J. Koenderink. *Solid Shape*. MIT Press, Cambridge, 1993.

[17] P. Kruszewski and S. Whitesides. A General Random Combinatorial Model of Botanical Trees. *Journal of Theoretical Biology*, 191(2):221–236, 1998.

[18] B. Lintermann and O. Deussen. XFROG 2.0. www.greenworks.de, December 1998.

[19] B. Lintermann and O. Deussen. Interactive Modeling of Plants. *IEEE Computer Graphics and Applications*, 19(1):56–65, 1999.

[20] R. Měch. *Modeling and Simulation of the Interactions of Plants with the Environment using L-systems and their Extensions*. PhD thesis, University of Calgary, October 1997.

[21] R. Měch and P. Prusinkiewicz. Visual Models of Plants Interacting with their Environment. Proceedings of SIGGRAPH 96, Annual Conference Series, August, 1996, pages 397–410.

[22] K. J. Niklas. *Plant Allometry: The Scaling of Form and Process*. The University of Chicago Press, Chicago, 1994.

[23] P. Oppenheimer. Real Time Design and Animation of Fractal Plants and Trees. Proceedings of SIGGRAPH 86, in *Computer Graphics*, 20, 4, August 1986, pages 151–158.

[24] W. F. Powell. *Drawing Trees*. Walter Foster Publishing, Inc., Laguna Hills, CA, 1998.

[25] P. Prusinkiewicz, J. Hanan, and R. Měch. An L-system-based Plant Modeling Language. Lecture Notes in Computer Science 1779, pages 395–410. Springer-Verlag, Berlin, 2000.

[26] P. Prusinkiewicz, M. James, and R. Měch. Synthetic Topiary. Proceedings of SIGGRAPH 94, Annual Conference Series, July, 1994, pages 351–358.

[27] P. Prusinkiewicz, R. Karwowski, R. Měch, and J. Hanan. L-studio/cpfg: A Software System for Modeling Plants, 2000. Lecture Notes in Computer Science 1779, pages 457–464. Springer-Verlag, Berlin, 2000.

[28] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, 1990. With J. S. Hanan, F. D. Fracchia, D. R. Fowler, M. J. M. de Boer, and L. Mercer.

[29] W. T. Reeves and R. Blau. Approximate and Probabilistic Algorithms for Shading and Rendering Structured Particle Systems. Proceedings of SIGGRAPH 85, in *Computer Graphics*, 19, 3, July 1985, pages 313–322.

[30] W. R. Remphrey and G. R. Powell. Crown Architecture of *Larix laricina* Saplings: Quantitative Analysis and Modelling of (nonsylleptic) Order 1 Branching in Relation to Development of the Main Stem. *Canadian Journal of Botany*, 62(9):1904–1915, 1984.

[31] J. N. Ridley. Ideal Phyllotaxis on General Surfaces of Revolution. *Mathematical Biosciences*, 79:1–24, 1986.

[32] T. Sakaguchi. Botanical Tree Structure Modeling Based on Real Image Set. SIGGRAPH 98 Conference Abstracts and Applications, 1998.

[33] J. M. Snyder and J. T. Kajiya. Generative Modeling: A Symbolic System for Geometric Modeling. Proceedings of SIGGRAPH 92, in *Computer Graphics*, 26, 2, July 1992, pages 369–378.

[34] I. Vaisman. *A First Course in Differential Geometry*. Marcel Dekker, New York, 1984.

[35] H. Vogel. A Better Way to Construct the Sunflower Head. *Mathematical Biosciences*, 44:179–189, 1979.

[36] J. Weber and J. Penn. Creation and Rendering of Realistic Trees. Proceedings of SIGGRAPH 95, Annual Conference Series, August, 1995, pages 119–128.

[37] K. West. *How to Draw Plants. The Techniques of Botanical Illustration*. Timber Press, Portland, OR, 1997.

[38] E. Wunderlich. *Botanical Illustration in Watercolor*. Watson–Guptill, New York, 1991.