Animation of Plant Development

Przemyslaw Prusinkiewicz†, Mark Hammel† and Eric Mjolsness‡

†Department of Computer Science
University of Calgary
Calgary, Alberta, Canada T2N 1N4
e-mail: pwp|hammel@cpsc.ucalgary.ca

‡Department of Computer Science
Yale University
New Haven, CT 06520-2158

From Proceedings of SIGGRAPH 93 (Anaheim, California, August 1–6, 1993),
In *Computer Graphics* Proceedings, Annual Conference Series, 1993,
ACM SIGGRAPH, pp. 351–360.

For technical reasons, the formatting of this paper
is slightly different from the original publication.

Animation of Plant Development

Przemyslaw Prusinkiewicz
Department of Computer Science
University of Calgary
Calgary, Alberta, Canada T2N 1N4

Mark S. Hammel
Department of Computer Science
University of Calgary
Calgary, Alberta, Canada T2N 1N4

Eric Mjolsness
Department of Computer Science
Yale University
New Haven, CT 06520-2158

ABSTRACT

This paper introduces a combined discrete/continuous model of plant development that integrates L-system-style productions and differential equations. The model is suitable for animating simulated developmental processes in a manner resembling time-lapse photography. The proposed technique is illustrated using several developmental models, including the flowering plants *Campanula rapunculoides, Lychnis coronaria*, and *Hieracium umbellatum*.

CR categories: F.4.2 **[Mathematical Logic and Formal Languages]**: Grammars and Other Rewriting Systems: *Parallel rewriting systems*, I.3.7 **[Computer Graphics]**: Three-Dimensional Graphics and Realism: *Animation*, I.6.3 **[Simulation and Modeling]**: Applications, J.3 **[Life and Medical Sciences]**: Biology

Keywords: animation through simulation, realistic image synthesis, modeling of plants, combined discrete/continuous simulation, L-system, piecewise-continuous differential equation.

1 INTRODUCTION

Time-lapse photography reveals the enormous visual appeal of developing plants, related to the extensive changes in topology and geometry during growth. Consequently, the animation of plant development represents an attractive and challenging problem for computer graphics. Its solution may enable us to retrace the growth of organs hidden from view by protective cell layers or tissues, illustrate processes that do not produce direct visual effects, and expose aspects of development obscured in nature by concurrent phenomena, such as the extensive daily motions of leaves and flowers. Depending on the application, different degrees of realism may be sought, ranging from diagrammatic representations of developmental mechanisms to photorealistic recreations of nature's beauty.

Known techniques for simulating plant development, such as L-systems [16, 27, 28, 31], their variants proposed by Aono and Kunii [1], and the AMAP software [4, 10], operate in discrete time, which means that the state of the model is known only at fixed time intervals. This creates several problems if a smooth animation of development is sought [27, Chapter 6]:

- Although, in principle, the time interval can be arbitrarily small, once it has been chosen it becomes a part of the model and cannot be easily changed. From the viewpoint of computer animation, it is preferable to specify this interval as an easy to control parameter, decoupled from the underlying model.

- The continuity criteria responsible for the smooth progression of shapes during animation can be specified more easily in the continuous time domain.

- It is conceptually elegant to separate the model of development, defined in continuous time, from its observation, taking place in discrete intervals.

Smooth animations of plant development have been created by Miller (a growing coniferous tree [19]), Sims (artificially evolved plant-like structures [30]), and Prusinkiewicz *et. al.* (a growing herbaceous plant *Lychnis coronaria* [24]), but the underlying techniques have not been documented in the literature. Greene proposed a model of branching structures [12] suitable for animating accretive growth [11], but this model does not capture the non-accretive developmental processes observed in real plants.

This paper introduces a mathematical framework for modeling plants and simulating their development in a manner suitable for animation. The key concept is the integration of discrete and continuous aspects of model behavior into a single formalism, called *differential L-systems* (dL-systems), where L-system-style productions express qualitative changes to the model (for example, the initiation of a new branch), and differential equations capture continuous processes, such as the gradual elongation of internodes.

The proposed integration of continuous and discrete aspects of development into a single model has several predecessors.

Barzel [2] introduced *piecewise-continuous ordinary differential equations (PODEs)* as a framework for modeling processes described by differential equations with occasionally occurring discontinuities. PODEs lack a formal generative mechanism for specifying changes to system configuration resulting from discrete events, and therefore cannot be directly applied to simulate the development of organisms consisting of hundreds or thousands of modules.

Fleischer and Barr [7] addressed this limitation in a model of morphogenesis consisting of cells developing in a continuous medium.

The configuration of the system is determined implicitly by its geometry. For example, in a simulated neural network, a synapse is formed when a growing dendrite of one cell reaches another cell.

Mjolsness *et. al.* [21] pursued an alternative approach in a *connectionist model of development*. Differential equations describe the continuous aspects of cell behavior during interphase (time between cell divisions), while productions inspired by L-systems specify changes to the system configuration resulting from cell division and death. The connectionist model makes it possible to consider networks with arbitrary topology (not limited to branching structures), but requires productions that operate globally on the entire set of cells constituting the model. This puts a practical limit on the number of components in the system.

Fracchia *et. al.* [9] (see also [27, Chapter 7]) animated the development of cellular layers using a physically-based model in which differential equations simulate cell growth during the interphase, and productions of a *map L-system* capture cell divisions. The productions operate locally on individual cells, making it possible to simulate the development of arbitrarily large layers using a finite number of rules. Unfortunately, this technique does not seem to extend beyond the modeling of cellular layers.

Timed L-systems [27, Chapter 6] were introduced specifically as a formal framework for constructing models of branching structures developing in continuous time. They operate under the assumption that no information exchange between coexisting modules takes place. This is a severe limitation, as interactions between the modules are known to play an important role in the development of many plant species [14, 27, 28]. A practical application of timed L-systems to animation is described by Noser *et. al.* [22].

The model of development proposed in this paper combines elements of PODEs, the connectionist model, and L-systems. The necessary background in L-systems is presented in Section 2. Sections 3 and 4 introduce the definition of differential L-systems and illustrate it using two simple examples. Section 5 applies combined discrete/continuous simulation techniques to evaluate dL-systems over time. Section 6 focuses on growth functions, which characterize continuous aspects of model development. Application of differential L-systems to the animation of the development of higher plants is presented in Section 7, using the models of a compound leaf and three herbaceous plants as examples. A summary of the results and a list of open problems conclude the paper.

2 L-SYSTEMS

An extensive exposition of L-systems applied to the modeling of plants is given in [27]. Below we summarize the main features of L-systems pertinent to the present paper.

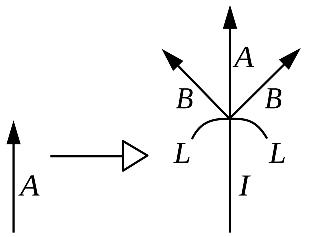


Figure 1: Example of a typical L-system production

We view a plant as a linear or branching structure composed of repeated units called *modules*. An L-system describes the development of this structure in terms of *rewriting rules* or *productions*, each of which replaces the *predecessor* module by zero, one, or more *successor* modules. For example, the production in Figure 1 replaces apex A by a structure consisting of a new apex A, an internode I, and two lateral apices B supported by leaves L.

In general, productions can be *context free* and depend only on the replaced module, or *context-sensitive* and depend also on the neighborhood of this module. A *developmental sequence* is generated by repeatedly applying productions to the consecutively obtained structures. In each step, productions are applied in parallel to all parts of the structure obtained so far.

The original formalism of L-systems [16] has a threefold discrete character [17]: the modeled structure is a finite collection of modules, each of these modules is in one of a finite number of states, and the development is simulated in discrete derivation steps. An extension called *parametric L-systems* [25, 27] increases the expressive power of L-systems by introducing a continuous characterization of the module states. Each module is represented by an identifier denoting the module *type* (one or more symbols starting with a letter) and a *state vector* of zero, one, or more numerical *parameters*. For instance, $M = A(5, 9.5)$ denotes a module M of type A with two parameters $w_1 = 5$ and $w_2 = 9.5$, forming the vector $\mathbf{w} = (5, 9.5)$. The interpretation of parameters depends on the semantics of the module definition, and may vary from one module type to another. For example, parameters may quantify the shape of the module, its age, and the concentration of substances contained within it.

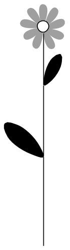


Figure 2: Turtle interpretation of a sample string

In the formalism of L-systems, modeled structures are represented as *strings* of modules. Branching structures are captured using *bracketed* strings, with the matching pairs of brackets [and] delimiting branches. We visualize these structures using a *turtle interpretation* of strings [23, 28], extended to strings of modules with parameters in [13, 25, 27]. A predefined interpretation is assigned to a set of reserved modules. Some of them represent physical parts of the modeled plant, for example a leaf or an internode, while others represent local properties, such as the magnitude of a branching angle. Reserved modules frequently used in this paper are listed below:

$F(x)$	line segment of length x,
$+(\alpha), -(\alpha)$	orientation change of the following line by $\pm\alpha$ degrees with respect to the preceding line,
$@X(s)$	a predefined surface X scaled by the factor s.

The interpretation of a string of modules proceeds by scanning it from left to right and considering the reserved modules as commands that maneuver a LOGO-style turtle. For example, Figure 2 shows the turtle interpretation of a sample string:

$$F(1)[+(45)@L(0.75)]F(0.8)[-(30)@L(0.5)]F(0.6)@K(1),$$

where symbols $@L$ and $@K$ denote predefined surfaces depicting a leaf and a flower.

3 DEFINITION OF dL-SYSTEMS

Differential L-systems extend parametric L-systems by introducing continuous time flow in place of a sequence of discrete derivation steps. As long as the parameters $\mathbf{w}$ of a module $A(\mathbf{w})$ remain in the *domain of legal values* $\mathcal{D}_A$, the module develops in a continuous way. Once the parameter values reach the boundary $\mathcal{C}_A$ of the domain $\mathcal{D}_A$, a production replaces module $A(\mathbf{w})$ by its descendants in a discrete event. The form of this production may depend on which segment $\mathcal{C}_{A_k}$ of the boundary of $\mathcal{D}_A$ has been crossed.

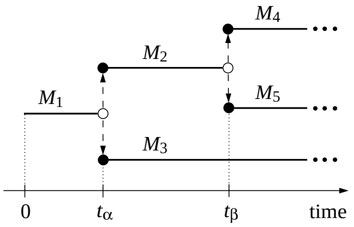


Figure 3: Fragment of the lineage tree of a hypothetical modular structure

For example, module M_2 in Figure 3 is created at time t_α as one of two descendants of the initial module M_1. It develops in the interval $[t_\alpha, t_\beta)$, and ceases to exist at time t_β, giving rise to two new modules M_4 and M_5. The instant t_β is the time at which parameters of M_2 reach the boundary of its domain of legal states $\mathcal{D}$. A hypothetical trajectory of module M_2 in its parameter space is depicted in Figure 4.

In order to formalize the above description, let us assume that the modeled structure consists of a sequence of modules (an extension to branching structures is straightforward if a proper definition of context is used [27, 28]). The state of the structure at time t is represented as a string:

$$\mu = A_1(\mathbf{w}_1) A_2(\mathbf{w}_2) \cdots A_n(\mathbf{w}_n).$$

The module $A_{i-1}(\mathbf{w}_{i-1})$ immediately preceding a given module $A_i(\mathbf{w}_i)$ in the string μ is called the *left neighbor* or *left context* of $A_i(\mathbf{w}_i)$, and the module $A_{i+1}(\mathbf{w}_{i+1})$ immediately following $A_i(\mathbf{w}_i)$ is called its *right neighbor* or *right context*. When it is inconvenient to list the indices, we use the symbols $<$, $>$, and/or subscripts l, r to specify the context of $A(\mathbf{w})$, as in the expression:

$$A_l(\mathbf{w}_l) < A(\mathbf{w}) > A_r(\mathbf{w}_r).$$

The continuous behavior of $A(\mathbf{w})$ is described by an *ordinary differential equation* that determines the rate of change $d\mathbf{w}/dt$ of parameters $\mathbf{w}$ as a function of the current value of these parameters and those of the module's neighbors:

$$\frac{d\mathbf{w}}{dt} = f_A(\mathbf{w}_l, \mathbf{w}, \mathbf{w}_r).$$

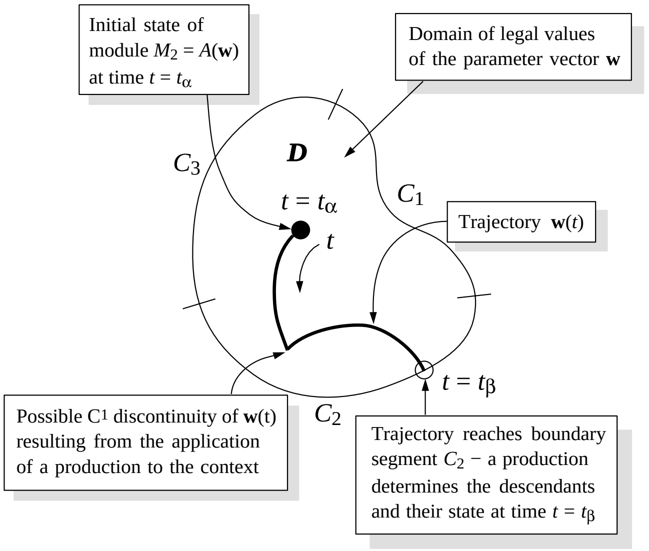
The above equation applies as long as the parameters $\mathbf{w}$ are in the domain $\mathcal{D}_A$ characteristic to the module type A. We assume that $\mathcal{D}_A$ is an open set, and specify its boundary $\mathcal{C}_A$ as the union of a finite number $m \geq 1$ of nonintersecting segments $\mathcal{C}_{A_k}$, $k = 1, 2, \ldots, m$. The time t_β at which the trajectory of module $A(\mathbf{w})$ reaches a segment $\mathcal{C}_{A_k}$ of the boundary of $\mathcal{D}_A$ satisfies the expression:

$$\lim_{t \to t_\beta^-} \mathbf{w}(t) \in \mathcal{C}_{A_k}.$$

The replacement of module $A(\mathbf{w})$ by its descendants at time t_β is described by a *production*:

$$p_{A_k} : A_l(\mathbf{w}_l) < A(\mathbf{w}) > A_r(\mathbf{w}_r) \longrightarrow B_{k,1}(\mathbf{w}_{k,1}) B_{k,2}(\mathbf{w}_{k,2}) \cdots B_{k,m_k}(\mathbf{w}_{k,m_k}).$$

The module $A(\mathbf{w})$ is called the *strict predecessor* and the sequence of modules $B_{k,1}(\mathbf{w}_{k,1}) B_{k,2}(\mathbf{w}_{k,2}) \cdots B_{k,m_k}(\mathbf{w}_{k,m_k})$ is called the *successor* of this production. The index k emphasizes that different productions can be associated with individual segments $\mathcal{C}_{A_k}$ of the boundary $\mathcal{C}_A$. The initial value of parameters assigned to a module $B_{k,j}(\mathbf{w}_{k,j})$ upon its creation is determined by a function $h_{A_{k,j}}$ which takes as its arguments the values of the parameters $\mathbf{w}_l$, $\mathbf{w}$, and $\mathbf{w}_r$ at the time immediately preceding production application:



Initial state of module M2 = A(w) at time t = tα
Domain of legal values of the parameter vector w
D
C3
C1
t = tα
t
Trajectory w(t)
t = tβ
C2
Possible C1 discontinuity of w(t) resulting from the application of a production to the context
Trajectory reaches boundary segment C2 – a production determines the descendants and their state at time t = tβ

Figure 4: A hypothetical trajectory of module M_2 in its parameter space

$$\mathbf{w}_{k,j} = \lim_{t \to t_\beta^-} h_{A_{k,j}}(\mathbf{w}_l(t), \mathbf{w}(t), \mathbf{w}_r(t)).$$

The vector $\mathbf{w}_{k,j}$ must belong to the domain $\mathcal{D}_{B_{k,j}}$. (A stronger condition is needed to insure that the number of productions applied in any finite interval $[t, t + \Delta t]$ will be finite.)

In summary, a differential L-system is defined by the initial string of modules μ_0 and the specification of each module type under consideration. The specification of a module type A consists of four components:

$$< \mathcal{D}_A, \mathcal{C}_A, f_A, P_A >,$$

where:

- the open set $\mathcal{D}_A$ is the domain of legal parameter values of modules of type A,
- the set $\mathcal{C}_A = \mathcal{C}_{A_1} \cup \ldots \cup \mathcal{C}_{A_m}$ is the boundary of $\mathcal{D}_A$, consisting of nonintersecting segments $\mathcal{C}_{A_1}, \ldots, \mathcal{C}_{A_m}$,
- the function f_A specifies a system of differential equations that describe the continuous behavior of modules of type A in their domain of legal parameter values $\mathcal{D}_A$,
- the set of productions $P_A = \{p_{A_1}, \ldots, p_{A_m}\}$ captures the discrete behavior of modules of type A.

A production $p_{A_k} \in P_A$ is applied when the parameters of a module M of type A reach segment $\mathcal{C}_{A_k}$ of the boundary $\mathcal{C}_A$. At this time module M disappears, and zero, one, or more descendant modules are created. The functions $h_{A_{k,j}}$ embedded in productions p_{A_k} determine the initial values of parameters in the successor modules.

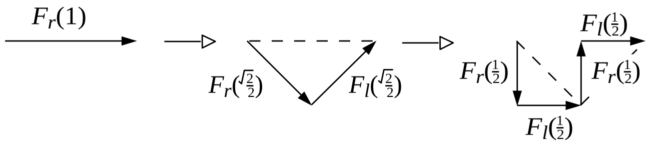


Figure 5: Initial steps in the construction of a dragon curve

4 EXAMPLES OF dL-SYSTEMS

We will illustrate the notion of a dL-system using two sample models suitable for animating the development of the *dragon curve* and the filamentous alga *Anabaena catenula*.

4.1 A dL-system model of the dragon curve

In the discrete case, consecutive iterations of the *dragon curve* (described, for example, in [27, Chapter 1]) can be obtained by the following parametric L-system:

$$\begin{aligned} \omega &: \quad --F_r(1) \\ p_1 &: \quad F_r(s) \rightarrow -F_r(s\tfrac{\sqrt{2}}{2}) + +F_l(s\tfrac{\sqrt{2}}{2})- \\ p_2 &: \quad F_l(s) \rightarrow +F_r(s\tfrac{\sqrt{2}}{2}) - -F_l(s\tfrac{\sqrt{2}}{2})+ \end{aligned}$$

Assuming that symbols $+$ and $-$ represent turns of $\pm 45°$, this L-system encodes a Koch construction [18, Chapter 6] that repeatedly substitutes sides of an isosceles right-angled triangle for its hypotenuse (Figure 5). Subscripts l and r indicate that the triangle is formed respectively on the left or right side of the oriented predecessor segment. A corresponding dL-system that generates the dragon curve through the continuous progression of shapes indicated in Figure 6 is given below:

```
**initial string:** $--F_r(1,1)$
$F_r(x,s)$ :
    **if** $x < s$ **solve** $\frac{dx}{dt} = \frac{s}{T}, \frac{ds}{dt} = 0$
    **if** $x = s$ **produce** $-F_r(0, s\frac{\sqrt{2}}{2}) + F_h(s,s) + F_l(0, s\frac{\sqrt{2}}{2})-$
$F_l(x,s)$ :
    **if** $x < s$ **solve** $\frac{dx}{dt} = \frac{s}{T}, \frac{ds}{dt} = 0$
    **if** $x = s$ **produce** $+F_r(0, s\frac{\sqrt{2}}{2}) - F_h(s,s) - F_l(0, s\frac{\sqrt{2}}{2})+$
$F_h(x,s)$ :
    **if** $x > 0$ **solve** $\frac{dx}{dt} = -\frac{s}{T}, \frac{ds}{dt} = 0$
    **if** $x = 0$ **produce** $\varepsilon$
```

The operation of this model starts with the replacement of the initial module $F_r(1,1)$ with the string:

$$-F_r(0, \frac{\sqrt{2}}{2}) + F_h(1,1) + F_l(0, \frac{\sqrt{2}}{2})-,$$

which has the same turtle interpretation: a line segment of unit length [1]. Next, the horizontal line segment represented by module F_h decreases in length with the speed $\frac{dx}{dt} = -\frac{1}{T}$, while the diagonal segments represented by modules F_r and F_l elongate with the speed $\frac{dx}{dt} = \frac{\sqrt{2}}{2}\frac{1}{T}$. The constant T determines the lifetime of the modules: after time T, the module F_h reaches zero length and is removed from the string (replaced by the empty string ε), while both modules F_r and F_l reach their maximum length of $\frac{\sqrt{2}}{2}$ and are replaced by their respective successors. These successors subsequently follow the same developmental pattern.

It is not accidental that the predecessor and the successor of the productions for $F_r(x,s)$ and $F_l(x,s)$ have identical geometric interpretations. Since productions are assumed to be applied instantaneously, any change of the model's geometry introduced by a production would appear as a discontinuity in the animation. In general, correctly specified productions satisfy *continuity criteria* [27, Chapter 6], which means that they conserve physical entities such as shape, mass, and velocity of modules.

4.2 A dL-system model of *Anabaena catenula*

The continuously developing dragon curve has been captured by a context-free dL-system, in which all productions and equations depend only on the strict predecessor module. A simple example of a context-sensitive model inspired by the development of the blue-green alga *Anabaena catenula* [3, 20, 27] is given below.

Anabaena forms a nonbranching filament consisting of two classes of cells: *vegetative cells* and *heterocysts*. A vegetative cell usually divides into two descendant vegetative cells. However, in some cases a vegetative cell differentiates into a heterocyst. The spacing between heterocysts is relatively constant, in spite of the continuing growth of the filament. Mathematical models explain this phenomenon using a biologically motivated hypothesis that the distribution of heterocysts is regulated by nitrogen compounds produced by the heterocysts, diffusing from cell to cell along the filament, and decaying in the vegetative cells. If the compound concentration in a vegetative cell falls below a specific level, this cell differentiates into a heterocyst (additional factors are captured by more sophisticated models). A model operating in continuous time according to this description can be captured by the following dL-system:

```
**initial string:** $F_h(x_{max}, c_{max})F_v(x_{max}, c_{max})F_h(x_{max}, c_{max})$
$F(x_l, c_l) < F_v(x,c) > F(x_r, c_r)$ :
    **if** $x < x_{max}$ & $c > c_{min}$
        **solve** $\frac{dx}{dt} = rx, \frac{dc}{dt} = D \cdot (c_l + c_r - 2c) - \mu c$
    **if** $x = x_{max}$ & $c > c_{min}$
        **produce** $F_v(kx_{max}, c)F_v((1-k)x_{max}, c)$
    **if** $c = c_{min}$
        **produce** $F_h(x,c)$
$F_h(x,c)$:
        **solve** $\frac{dx}{dt} = r_x(x_{max} - x), \frac{dc}{dt} = r_c(c_{max} - c)$
```

Vegetative cells F_v and heterocysts F_h are characterized by their length x and concentration of nitrogen compounds c. The differential equations for the vegetative cell F_v indicate that while the cell length x is below the maximum value x_{max} and the compound concentration c is above the threshold c_{min}, the cell elongates exponentially according to the equation $\frac{dx}{dt} = rx$, and the compound concentration changes according to the equation:

$$\frac{dc}{dt} = D \cdot (c_l + c_r - 2c) - \mu c.$$

The first term in this equation describes diffusion of the compounds through the cell walls. Following *Fick's law* [5, page 404], the

[1] The turtle interprets the first parameter as the segment length, and ignores the second parameter.

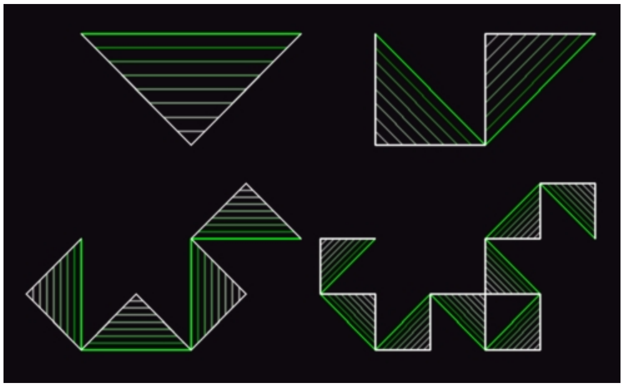


Figure 6: Development of the dragon curve simulated using a dL-system, recorded in time intervals $\Delta t = \frac{1}{8}T$. Top left: Superimposed stages $0-8$, top right: stages $8-16$, bottom row: stages $16-24$ and $24-32$.

rate of diffusion is proportional to the differences of compound concentrations, $c_r - c$ and $c_l - c$, between the neighbor cells and the cell under consideration. The term μc describes exponential decay of the compounds in the cell.

In addition to the differential equations, two productions describe the behavior of a vegetative cell. If the cell reaches maximum length x_{max} while the concentration c is still above the threshold c_{min}, the cell divides into two vegetative cells of length kx_{max} and $(1-k)x_{max}$, with the compound concentration c inherited from their parent cell. Otherwise, if the concentration c drops down to the threshold c_{min}, the cell differentiates into a heterocyst. Both productions satisfy the continuity criteria by conserving total cell length and concentration of nitrogen compounds.

The last line of the model specifies the behavior of the heterocysts. Their length and compound concentration converge exponentially to the limit values of x_{max} and c_{max}. The heterocysts do not undergo any further transformations.

Simulation results obtained using the above model are shown in Figure 7. The cells in the filament are represented as horizontal line segments with the colors indicating the concentration of nitrogen compounds. Consecutive developmental stages are drawn one under another. An approximately equal spacing between the heterocysts (shown in white) is maintained for any horizontal section, as postulated during model formulation.

Note that for incorrectly chosen constants in the model, the spacing between heterocysts may be distorted; for example, groups of adjacent vegetative cells may almost simultaneously differentiate into heterocysts.

5 EVALUATION OF dL-SYSTEMS

Although Figures 6 and 7 were obtained using dL-systems, we have not yet discussed the techniques needed to *evaluate* them. This term denotes the calculation of the sequence of strings $\mu(0) = \mu_0$, $\mu(\Delta t) = \mu_1, \ldots, \mu(n\Delta t) = \mu_n$ representing the states of the modeled structure at the desired intervals Δt. We address the problem of dL-system evaluation in the framework of the combined discrete/continuous paradigm for system simulation introduced by Fahrland [6] and presented in a tutorial manner by Kreutzer [15].

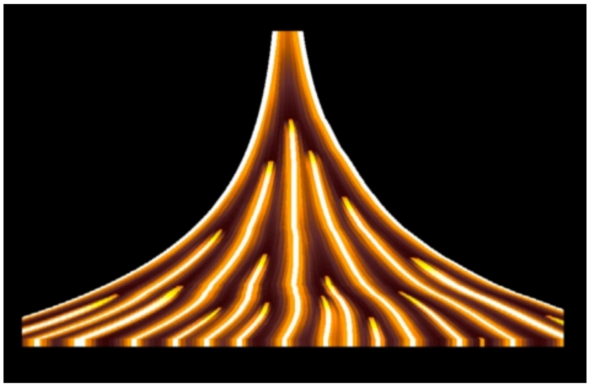


Figure 7: Diagrammatic representation of the development of *Anabaena catenula*, simulated using a dL-system with the constants set to the following values: $x_{max} = 1$, $c_{max} = 255$, $c_{min} = 5$, $D = \mu = 0.03$, $r = 1.01$, $k = 0.37$, $r_x = 0.1$, $r_c = 0.15$. The development was recorded from $t_{min} = 200$ to $t_{max} = 575$ at the intervals $\Delta t = 1$. Developmental stages are shown as horizontal lines with the colors indicating the concentration c of nitrogen compounds. Dark brown represents c_{min}; white represents c_{max}.

According to this paradigm, the evaluation can be viewed as a *dynamic process* governed by a *scheduler*: a part of the simulation program that monitors the state of the model, advances time, and dispatches the activities to be performed. In the absence of discrete events (productions), the scheduler repeatedly advances time by the *time slice* Δt. During each slice, the differential equations associated with the modules are integrated numerically (using an integration technique appropriate for the equations in hand), thus advancing the state of the structure from $\mu(t)$ to $\mu(t + \Delta t)$. If the scheduler detects that a discrete event should occur (i.e., a production should be applied) at time t' within the interval $[t, t + \Delta t)$, this interval is divided into two subintervals $[t, t')$ and $[t', t + \Delta t)$. The differential equations are integrated in the interval $[t, t')$ and yield parameter values for the production application at time t'. The production determines the initial values for the differential equations associated with the newly created modules; these equations are integrated in the remaining interval $[t', t + \Delta t)$. Each of the intervals $[t, t')$ and $[t', t + \Delta t)$ is subdivided further if more discrete events occur during $[t, t + \Delta t)$.

Plant structures generated using dL-systems may consist of large numbers (thousands) of modules. If many modules are replaced at different times t' during the interval $[t, t + \Delta t)$, the global advancement of time may require an excessive subdivision of this interval, leading to a slow evaluation of the model. This problem can be solved by detecting and processing events the interval $[t, t + \Delta t)$ individually for each module. The increase of simulation speed is obtained at the expense of accuracy, since the state of the context of a module replaced at time $t' \in (t, t + \Delta t)$ must be approximated, for example, by its state at time t. No accuracy is lost in the context-free case.

In the above description we assumed that the scheduler is capable of detecting each instant t' at which a discrete event occurs. If the differential equations are sufficiently simple, we can solve them analytically and determine time t' explicitly. In general, we need numerical techniques for special event location in piecewise-continuous ordinary differential equations, as described by Shampine *et. al.* [29], and Barzel [2, Appendix C].

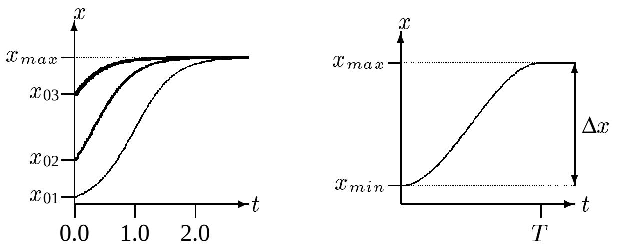


Figure 8: Examples of sigmoidal growth functions. a) A family of logistic functions plotted using $r = 3.0$ for different initial values x_0. b) A cubic function $G_{\Delta x, T}$.

6 GROWTH FUNCTIONS

Growth functions describe continuous processes such as the expansion of individual cells, elongation of internodes, and gradual increase of branching angles over time. For example, the differential equations included in the dL-system for the dragon curve (Section 4.1) describe linear elongation of segments F_r and F_l, and linear decrease in length of segments F_h. The dL-system model of *Anabaena* (Section 4.2) assumes exponential elongation of cells.

In higher plants, the growth functions are often of *sigmoidal* (S-shaped) type, which means that they initially increase in value slowly, then accelerate, and eventually level off at or near the maximum value. A popular example of a sigmoidal function is Velhurst's *logistic* function (c.f. [5, page 212]), defined by the equation:

$$\frac{dx}{dt} = r\left(1 - \frac{x}{x_{max}}\right)x$$

with a properly chosen initial value x_0 (Figure 8a). Specifically, x_0 must be greater than zero, which means that neither the initial length nor the initial growth rate of a module described by the logistic function will be equal to zero. In order to obtain a continuous progression of forms, it is often convenient to use a growth function that has zero growth rates at both ends of an interval T within which its value increases from x_{min} (possibly zero) to x_{max}. These requirements can be satisfied, for example, by a *cubic* function of time. Using the Hermite form of curve specification [8, page 484], we obtain:

$$x(t) = -2\frac{\Delta x}{T^3}t^3 + 3\frac{\Delta x}{T^2}t^2 + x_{min},$$

where $\Delta x = x_{max} - x_{min}$ and $t \in [0, T]$. The equivalent differential equation is:

$$\frac{dx}{dt} = -6\frac{\Delta x}{T^3}t^2 + 6\frac{\Delta x}{T^2}t = 6\frac{\Delta x}{T^2}\left(1 - \frac{t}{T}\right)t$$

with the initial condition $x_0 = x_{min}$. In order to extend this curve to infinity (Figure 8b), we define:

$$\frac{dx}{dt} = G_{\Delta x, T}(t) = \begin{cases} 6\frac{\Delta x}{T^2}\left(1 - \frac{t}{T}\right)t & \text{for } t \in [0, T] \\ 0 & \text{for } t \in (T, +\infty). \end{cases}$$

Although the explicit dependence of the function G on time is questionable from the biological point of view (a plant module does not have a means for measuring time directly), parametric cubic functions constitute a well understood computer graphics tool [8, Chapter 11.2] and can be conveniently used to approximate the observed changes of parameter values over time.

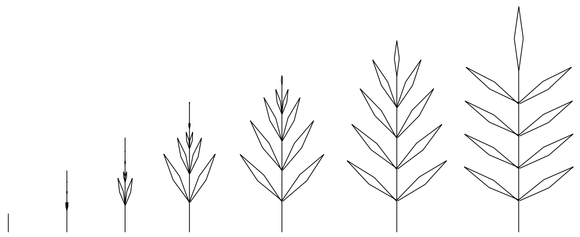


Figure 9: Development of a compound leaf simulated using a dL-system. Parameter values are: $n_0 = 4$, $x_0 = 1.0$, $x_{th} = 2.0$, $k = 0.5$, $r_a = 2.0$, $x_{amax} = 3.0$, $r_i = 1.0$, $x_{imax} = 3.0$, $s_0 = 0.05$, $r_s = 2.0$, $s_{max} = 6.0$, $\alpha_0 = 2.0$, $r_\alpha = 1.0$, $\alpha_{max} = 60.0$, and $\Delta t = 0.01$. The stages shown represent frames 50, 215, 300, 400, 500, 600, and 900 of an animated sequence.

7 MODELING OF HIGHER PLANTS

In this section we present sample applications of dL-systems to the animation of the development of higher plants.

7.1 Pinnate Leaf

A pinnate leaf provides a simple example of a *monopodial* branching structure. *Monopodial* branching occurs when the apex of the main axis produces a succession of *nodes* bearing *organs* — leaves or flowers — which are separated by *internodes*. In the case of pinnate leaves with the leaflets occurring in pairs (termed *opposite* arrangement), the essence of this process can be captured by the L-system production [27, page 71]:

$$F_a \longrightarrow F_i[+@L][-@L]F_a,$$

where F_a denotes the apex, F_i — an internode, and $@L$ — a leaflet. The dL-system model given below extends this L-system with growth functions that control the expansion of all components and gradually increase branching angles over time.

```
initial string: $F_a(x_0, n_0)$
$F_a(x, n)$ :
    if $x < x_{th}$
        solve $\frac{dx}{dt} = r_a\left(1 - \frac{x}{x_{amax}}\right)x$, $\frac{dn}{dt} = 0$
    if $x = x_{th}$ & $n > 0$
        produce $F_i(kx)[+(\alpha_0)@L(s_0)][-(\alpha_0)@L(s_0)]$
                $F_a((1-k)x, n-1)$
    if $x = x_{th}$ & $n = 0$
        produce $F_i(x)@L(s_0)$
$F_i(x)$ :  solve $\frac{dx}{dt} = r_i\left(1 - \frac{x}{x_{imax}}\right)x$
$L(s)$ :  solve $\frac{ds}{dt} = r_s\left(1 - \frac{s}{s_{max}}\right)s$
$\pm(\alpha)$ :  solve $\frac{d\alpha}{dt} = r_\alpha\left(1 - \frac{\alpha}{\alpha_{max}}\right)\alpha$
```

The apex F_a has two parameters x and n which indicate its current length and the remaining number of internodes to be produced. The apex elongates according to the logistic function with parameters r (controlling growth rate) and x_{amax} (controlling the asymptotic apex length). Upon reaching the threshold length x_{th}, the apex produces a pair of leaflets $@L$ and subdivides into an internode F_i of length kx and a shorter apex of length $(1-k)x$. Once the predefined number n_0 of leaf pairs have been created, the apex

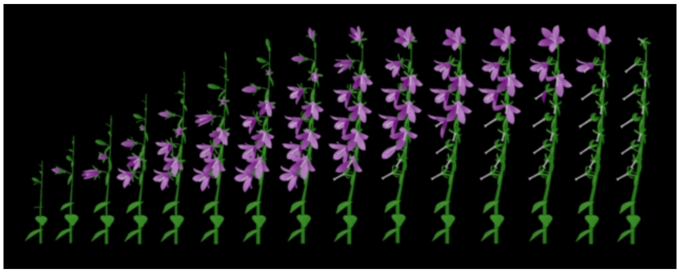


Figure 10: Development of the herbaceous plant *Campanula rapunculoides*. The snapshots show every 25^{th} frame of a computer animation, starting with frame 175.

transforms itself into an internode and produces the terminal leaflet. The length of internodes, the size of leaflets, and the magnitude of the branching angles increase according to the logistic functions. Snapshots of the leaf development simulated by the above model are shown in Figure 9.

7.2 *Campanula rapunculoides*

The inflorescence of *Campanula rapunculoides* (creeping bellflower) has a monopodial branching structure similar to that of a pinnate leaf; consequently, it is modeled by a similar dL-system:

```
initial string: $F_a(x_0, n_0)$
$F_a(x, n)$ :
    if $x < x_{th}$
        solve $\frac{dx}{dt} = v$, $\frac{dn}{dt} = 0$
    if $x = x_{th}$ & $n > 0$
        produce $F_i(kx)[+(\alpha_0)@K]F_a((1-k)x, n-1)$
    if $x = x_{th}$ & $n = 0$
        produce $F_i(x)@K$
$F_i(x)$ :  solve $\frac{dx}{dt} = G_{\Delta x, T_1}(t)$
$+(\alpha)$ :  solve $\frac{d\alpha}{dt} = G_{\Delta\alpha, T_2}(t)$
```

The apex is assumed to grow at a constant speed. Cubic growth functions describe the elongation of internodes and the gradual increase of branching angles. The combination of the linear growth of the apex with the cubic growth of the internodes results in first-order continuity of the entire plant height (except when apex F_a is transformed into internode F_i and terminal flower $@K$).

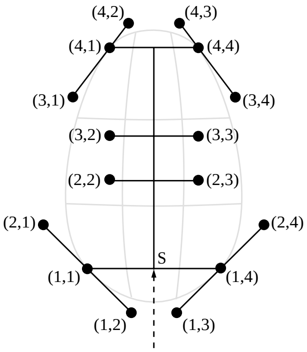


Figure 11: A Bézier patch defined by a branching structure

Figure 10 presents a sequence of snapshots from an animation of *Campanula*'s development. It was obtained using the above dL-system augmented with rules that govern the development of flowers $@K$ from a bud to an open flower to a fruit. The petals and sepals have been modeled as Bézier patches, specified by control points placed at the ends of simple branching structures (Figure 11). Each structure is attached to the remainder of the model at point S. The lengths of the line segments and the magnitudes of the branching angles have been controlled by cubic growth functions, yielding the developmental sequence shown in Figure 12. When the flower transforms into a fruit, productions instantaneously remove the petals from the model (it is assumed that the time over which a petal falls off is negligible compared to the time slice used for the animation of development). Manipulation of Bézier patches using L-systems has been described in detail by Hanan [13].

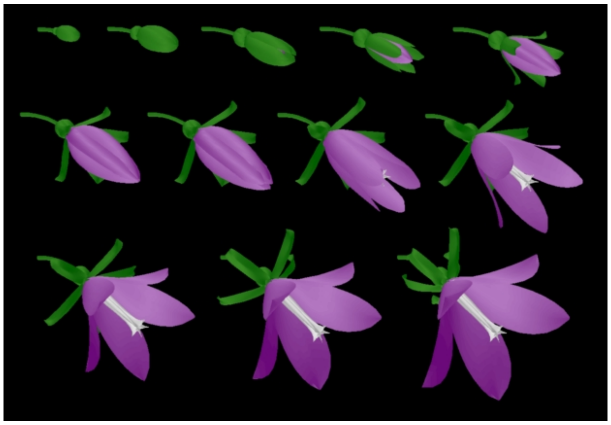


Figure 12: Development of a single flower of *Campanula rapunculoides*

7.3 *Lychnis coronaria*

The inflorescence of *Lychnis coronaria* (rose campion) is an example of a sympodial branching structure, characterized by large branches that carry the main thrust of development. As presented in [27, page 82] and [28], the apex of the main axis turns into a flower shortly after the initiation of a pair of lateral branches. Their apices turn into flowers as well, and second-order branches take over. The lateral branches originating at a common node develop at the same rate, but the development of one side is delayed with respect to the other. This process repeats recursively, as indicated by the following L-system:

$$\begin{array}{lll} \omega: & A_7 & \\ p_1: & A_7 \longrightarrow F[A_0][A_4]F@K & \\ p_2: & A_i \longrightarrow A_{i+1} & 0 \le i < 7 \end{array}$$

Production p_1 shows that, at their creation time, the lateral apices have different states A_0 and A_4. Consequently, the first apex requires eight derivation steps to produce flower $@K$ and initiate a new pair of branches, while the second requires only four steps.

A corresponding dL-system using cubic growth functions to describe the elongation of internodes F is given below:

```
initial string: $A(\tau_{max})$
$A(\tau)$ :
    if $\tau < \tau_{max}$ solve $\frac{d\tau}{dt} = 1$
    if $\tau = \tau_{max}$ produce $F(0)[A(0)][A\left(\frac{\tau_{max}}{2}\right)]F(0)@K$
$F(x)$ :    solve $\frac{dx}{dt} = G_{\Delta x, T}(t)$
```

For simplicity, we have omitted leaves and symbols controlling the relative orientation of branches in space. The operation of the model

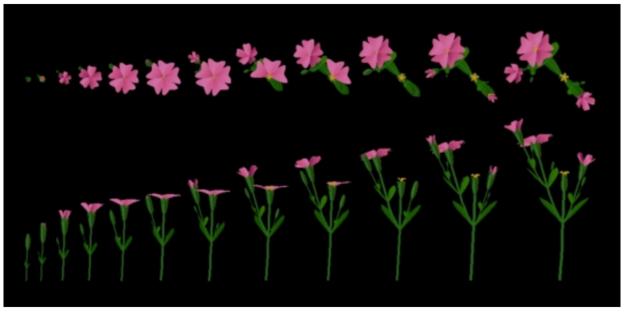


Figure 13: Development of *Lychnis coronaria*. The snapshots show every 25th frame of a computer animation, starting with frame 150.

is governed by apices A characterized by their age τ and assumed to have negligible size. Upon reaching the maximum age τ_{max}, an apex splits into two internodes F, creates two lateral apices A with different initial age values 0 and $\frac{\tau_{max}}{2}$, and initiates flower $@K$. In order to satisfy continuity criteria, the initial length of internodes is assumed to be zero.

Figure 13 shows selected snapshots from an animation of the development of *Lychnis* obtained using an extension of this dL-system. As in *Campanula*, the individual flowers have been modeled using Bézier patches controlled by the dL-system.

7.4 *Hieracium umbellatum*

The compound leaf and the inflorescences of *Campanula* and *Lychnis* have been captured by context-free dL-systems, assuming no flow of information between coexisting modules. Janssen and Lindenmayer [14] (see also [27, Chapter 3] and [28]) showed that context-free models are too weak to capture the whole spectrum of developmental sequences in plants. For example, the *basipetal* flowering sequence observed in many compound inflorescences requires the use of one or more signals that propagate through the developing structure and control the opening of buds. Such a sequence is characterized by the first flower opening at the top of the main axis and the flowering zone progressing downward towards the base of the plant.

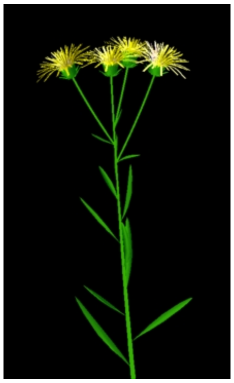


Figure 14: A model of *Hieracium umbellatum*.

Figure 14 shows a synthetic image of *Hieracium umbellatum*, a sample composite plant with a basipetal flowering sequence. Following *model I* postulated by Janssen and Lindenmayer, we assume that the opening of buds is controlled by a *hormone* generated at some point of time near the base of the plant and transported towards the apices. The hormone propagates faster in the main axis than in the lateral branches. As a result, it first reaches the bud of the main axis, then those of the lateral branches in the basipetal sequence. The growth of the main axis and of the lateral branches stops when the hormone attains their respective terminal buds. In addition, the hormone penetrating a node stops the development of a leaf originating at this node. Snapshots from a diagrammatic animated developmental sequence illustrating this process are shown in Figure 15.

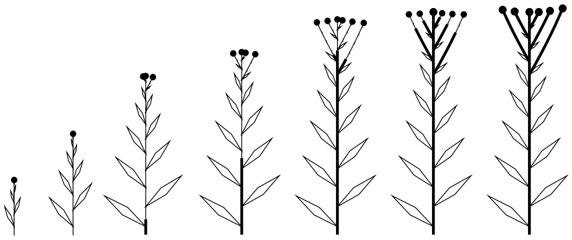


Figure 15: Development of *Hieracium umbellatum*. The stages shown represent frames 170, 265, 360, 400, 470, 496, and 520 of an animated sequence.

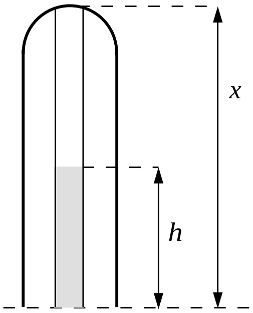


Figure 16: A conceptual model of the apex.

The complete listing of the dL-system capturing the development of *Hieracium* is too long to be included in this paper, but a specification of the activities of the main apex provides a good illustration of the context-sensitive control mechanism involved. We conceptualize this apex as a growing and periodically dividing tube of length x, which may be penetrated by the hormone to a height $h \leq x$ (Figure 16). The apex can assume three states: F_{a0} (not yet reached by the hormone), F_{a1} (being penetrated by the hormone), and F_{a2} (completely filled with the hormone). The apical behavior is captured by the following rules:

```
$F_l(x_l, h_l) < F_{a0}(x)$ :
    **if** $x_l > h_l$ & $x < x_{th}$
        **solve** $\frac{dx}{dt} = G(x)$
    **if** $x = x_{th}$
        **produce** $F_{i0}(kx)[F_{a0}(0)]F_{a0}((1-k)x)$
    **if** $x_l = h_l$ & $x < x_{th}$
        **produce** $F_{a1}(x, 0)$
$F_{a1}(x, h)$:
    **if** $x > h$ & $x < x_{th}$
        **solve** $\frac{dx}{dt} = G(x)$, $\frac{dh}{dt} = v$
    **if** $kx > h$ & $x = x_{th}$
        **produce** $F_{i1}(kx, h)[F_{a0}(0)]F_{a0}((1-k)x)$
    **if** $x > h \geq kx$ & $x = x_{th}$
        **produce** $F_{i2}(kx, kx)F_{a1}((1-k)x, h-kx)$
    **if** $x = h$
        **produce** $F_{a2}(x, x)$
```

The first three rules model the apex without the hormone. If the preceding internode F_l is not yet completely penetrated by the hormone $(x_l > h_l)$ and the length x of the apex is below the threshold value x_{th}, the apex elongates according to the growth function $G(x)$. Upon reaching the threshold length $(x = x_{th})$, the apex F_{a0} subdivides, producing an internode F_{i0} and a lateral apex F_{a0}. Finally, once the hormone penetrates the entire internode F_l (as indicated by the condition $x_l = h_l$), it flows into the apex, which then changes its state to F_{a1}.

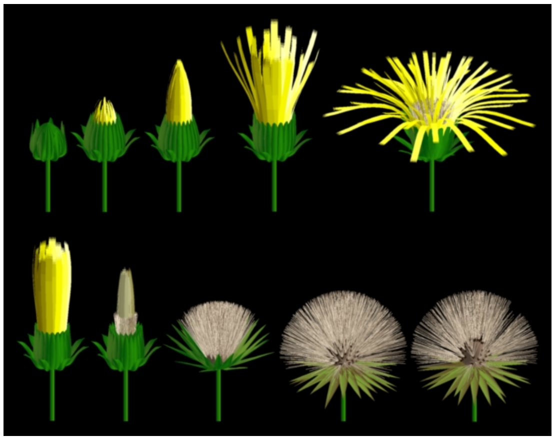


Figure 17: Development of a single flower head of *Hieracium umbellatum*

The continuous rule for F_{a1} describes the growth of the apex with rate $G(x)$ and the propagation of the hormone with constant speed v. The next two productions capture the alternate cases of the apex subdivision, with the hormone level h below or above the level kx at which the new internode splits from the apex. The last production is applied when the hormone reaches the tip of the apex, and changes its state to the flowering state F_{a2}.

The complete model of *Hieracium umbellatum* contains additional rules that describe the elongation of internodes, the propagation of the hormone within and between the internodes, and the development of flower heads. The heads undergo the sequence of transformations illustrated in Figure 17. The bracts (green parts of the flower head) have been represented using Bézier patches controlled by the dL-system, while the petals have been formed as extending chains of filled rectangles, with the angles between consecutive rectangles controlled by cubic growth functions. This technique allowed us to represent each petal with a relatively modest number of polygons (10).

8 CONCLUSIONS

We have introduced differential L-systems as a combined discrete/continuous model suitable for computer simulation and animation of plant development. Continuous aspects of module behavior are described by ordinary differential equations, and discontinuous qualitative changes are captured by productions. The link between L-systems and dL-systems makes it possible to use existing discrete developmental models as a starting point for constructing dL-systems suitable for animation.

Differential L-systems have a wide spectrum of prospective applications, ranging from modest projects, such as the diagrammatic animation of developmental mechanisms employed by plants, to ambitious ones, such as the realistic animation of the growth of extinct plants. On the conceptual level, dL-systems expand piecewise-continuous differential equations with a formal specification of discrete changes to system configuration. The resulting formalism makes it possible to model developing branching structures with a theoretically unlimited number of modules. From a different perspective, dL-systems can be considered as the continuous-time extension of parametric L-systems.

The following problems still require solutions:

- **Combined differential-algebraic specification of continuous processes.** In some cases it is convenient to describe continuous aspects of model behavior using explicit functions of time instead of differential equations. For example, the expression of the cubic growth function using the differential equation presented in Section 6 is somewhat artificial. In order to accommodate explicit function specifications, the definition of dL-systems should be extended to comprehend differential-algebraic equations.

- **Incorporation of stochastic rules.** Differential L-systems have been formulated in deterministic terms. Stochastic rules should be incorporated to capture the specimen-to-specimen variations in modeled plants, as has been done for L-systems.

- **Development of the simulation software.** The simulations discussed in this paper were carried out using a programming language based on parametric L-systems [13, 26]. In this environment, the user must explicitly specify the formulae for numerically solving the differential equations included in the models (the forward Euler method was used in all cases). From the user's perspective, it would be preferable to incorporate a differential equation solver into the simulator, and specify the models directly in terms of dL-systems.

- **Improved realism of dL-system models.** We have not addressed many practical problems related to the construction of realistic models, such as the avoidance of intersections between modules, the improved modeling of growing plant organs (petals, leaves, and fruits), and the simulation of wilting.

The simulation and visualization of natural phenomena has the intriguing charm of blurring the line dividing the synthesis of images from the re-creation of nature. The animation of plant development adds a new phenomenon to this (un)real world.

Acknowledgements

We would like to thank Jim Hanan for his essential work on the plant modeling software *cpfg* used in the simulations, and for valuable references and comments. At different points in time, Gavin Miller, Karl Sims and Alvy Ray Smith revealed to us the techniques used in their developmental animations. M. Raju and C. C. Chinnappa explained the details of the development of *Lychnis coronaria* and *Hieracium umbellatum*. We also gained many insights from illuminating discussions with Bill Remphrey, John Reinitz, Stan Letovsky, and Keith Ferguson. This research was sponsored by an operating grant and a graduate scholarship from the Natural Sciences and Engineering Research Council of Canada, and a grant from the U.S. Air Force Office of Scientific Research.

References

[1] M. Aono and T. L. Kunii. Botanical tree image generation. *IEEE Computer Graphics and Applications*, 4(5):10–34, 1984.

[2] R. Barzel. *Physically-based modeling for computer graphics — a structured approach.* Academic Press, Boston, 1992.

[3] C. G. de Koster and A. Lindenmayer. Discrete and continuous models for heterocyst differentiation in growing filaments of blue-green bacteria. *Acta Biotheoretica*, 36:249–273, 1987.

[4] P. de Reffye, C. Edelin, J. Françon, M. Jaeger, and C. Puech. Plant models faithful to botanical structure and development. Proceedings of SIGGRAPH '88 (Atlanta, Georgia, August 1–5, 1988), in *Computer Graphics* 22, 4 (August 1988), pages 151–158, ACM SIGGRAPH, New York, 1988.

[5] L. Edelstein-Keshet. *Mathematical models in biology*. Random House, New York, 1988.

[6] D. A. Fahrland. Combined discrete event – continuous systems simulation. *Simulation*, 14(2):61–72, 1970.

[7] K. W. Fleischer and A. H. Barr. A simulation testbed for the study of multicellular development: Multiple mechanisms of morphogenesis. To appear in *Artificial Life III*, Addison-Wesley, Redwood City, 1993.

[8] J. D. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer graphics: Principles and practice*. Addison-Wesley, Reading, 1990.

[9] F. D. Fracchia, P. Prusinkiewicz, and M. J. M. de Boer. Animation of the development of multicellular structures. In N. Magnenat-Thalmann and D. Thalmann, editors, *Computer Animation '90*, pages 3–18, Tokyo, 1990. Springer-Verlag.

[10] J. Françon. Sur la modélisation de l'architecture et du développement des végétaux. In C. Edelin, editor, *L'Arbre. Biologie et Développement*. Naturalia Monspeliensia, 1991. No hors série.

[11] N. Greene. Organic architecture. SIGGRAPH Video Review 38, segment 16, ACM SIGGRAPH, New York, 1988.

[12] N. Greene. Voxel space automata: Modeling with stochastic growth processes in voxel space. Proceedings of SIGGRAPH '89 (Boston, Mass., July 31–August 4, 1989), in *Computer Graphics* 23, 4 (August 1989), pages 175–184, ACM SIGGRAPH, New York, 1989.

[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] J. M. Janssen and A. Lindenmayer. Models for the control of branch positions and flowering sequences of capitula in *Mycelis muralis* (L.) Dumont (Compositae). *New Phytologist*, 105:191–220, 1987.

[15] W. Kreutzer. *System simulation: Programming styles and languages*. Addison-Wesley, Sydney, 1986.

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

[17] A. Lindenmayer and H. Jürgensen. Grammars of development: Discrete-state models for growth, differentiation and gene expression in modular organisms. In G. Rozenberg and A. Salomaa, editors, *Lindenmayer systems: Impacts on theoretical computer science, computer graphics, and developmental biology*, pages 3–21. Springer-Verlag, Berlin, 1992.

[18] B. B. Mandelbrot. *The fractal geometry of nature*. W. H. Freeman, San Francisco, 1982.

[19] G. S. P. Miller. Natural phenomena: My first tree. Siggraph 1988 Film and Video Show.

[20] G. J. Mitchison and M. Wilcox. Rules governing cell division in *Anabaena*. *Nature*, 239:110–111, 1972.

[21] E. Mjolsness, D. H. Sharp, and J. Reinitz. A connectionist model of development. *Journal of Theoretical Biology*, 152(4):429–454, 1991.

[22] H. Noser, D. Thalmann, and R. Turner. Animation based on the interaction of L-systems with vector force fields. In T. L. Kunii, editor, *Visual computing – integrating computer graphics with computer vision*, pages 747–761. Springer-Verlag, Tokyo, 1992.

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

[24] P. Prusinkiewicz, M. Hammel, and J. Hanan. Lychnis coronaria. QuickTime movie included in the Virtual Museum CD-ROM, Apple Computer, Cupertino, 1992.

[25] P. Prusinkiewicz and J. Hanan. Visualization of botanical structures and processes using parametric L-systems. In D. Thalmann, editor, *Scientific Visualization and Graphics Simulation*, pages 183–201. J. Wiley & Sons, Chichester, 1990.

[26] P. Prusinkiewicz and J. Hanan. L-systems: From formalism to programming languages. In G. Rozenberg and A. Salomaa, editors, *Lindenmayer systems: Impacts on theoretical computer science, computer graphics, and developmental biology*, pages 193–211. Springer-Verlag, Berlin, 1992.

[27] 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.

[28] P. Prusinkiewicz, A. Lindenmayer, and J. Hanan. Developmental models of herbaceous plants for computer imagery purposes. Proceedings of SIGGRAPH '88 (Atlanta, Georgia, August 1–5, 1988), in *Computer Graphics* 22, 4 (August 1988), pages 141–150, ACM SIGGRAPH, New York, 1988.

[29] L. F. Shampine, I. Gladwell, and R. W. Brankin. Reliable solution of special event location problems for ODEs. *ACM Transactions on Mathematical Software*, 17, No. 1:11–25, March 1991.

[30] K. Sims. Panspermia. SIGGRAPH Video Review, ACM SIGGRAPH, New York, 1990.

[31] A. R. Smith. Plants, fractals, and formal languages. Proceedings of SIGGRAPH '84 (Minneapolis, Minnesota, July 22–27, 1984) in *Computer Graphics*, 18, 3 (July 1984), pages 1–10, ACM SIGGRAPH, New York, 1984.