Generating Spatial Distributions for Multilevel Models of Plant Communities

Brendan Lane Przemyslaw Prusinkiewicz

Department of Computer Science
University of Calgary
laneb | pwp @ cpsc.ucalgary.ca

Abstract

The simulation and visualization of large groups of plants has many applications. The extreme visual complexity of the resulting scenes can be captured using multilevel models. For example, in two-level models, plant distributions may be determined using coarse plant representations, and realistic visualizations may be obtained by substituting detailed plant models for the coarse ones. In this paper, we focus on the coarse aspect of modeling, the specification of plant distribution. We consider two classes of models: local-to-global models, rooted in the individual-based ecosystem simulations, and inverse, global-to-local models, in which positions of individual plants are inferred from a given distribution of plant densities. We extend previous results obtained using both classes of models with additional phenomena, including clustering and succession of plants. We also introduce the formalism of multiset L-systems to formalize the individual-based simulation models.

Key words: realistic image synthesis, multilevel modeling, plant ecosystem, spatial distribution, clustering, succession, multiset L-system

1 Introduction

The simulation and visualization of plant ecosystems has many theoretical and practical applications. They include fundamental research in ecology, visual impact analysis of forestry practices, and synthesis of complex scenery for computer animations, among others. The inherent complexity of the scenes resulting from the ecosystem simulations can be managed using the multilevel approach to modeling [3]. Rather than model the entire ecosystem at the detailed level of plant organs, such as leaves, flowers, apices, and internodes [13], the multilevel approach employs a hierarchy of models. For example, in the simplest, two-level case, a high-level model determines the distribution of the plants, and lower-level models determine the plants' shapes. The models are coupled so that information created at a higher level can affect the outcome of the model at the lower level.

In this paper, we focus on the generation of the spatial distribution of plants. Specifically, we extend the methods reported in [3] with the ecologically and visually important phenomena of clustering and succession of plants. We also introduce the formalism of *multiset L-systems* to formalize some of these models.

Previous work on multilevel modeling of plant ecosystems is summarized in Section 2. Following the approach introduced there, we distinguish the *local-to-global* approach, in which the distribution of plant densities is determined by a simulation of interactions between the individual plants, and the *global-to-local* approach, in which positions of individual plants are inferred from given large-scale density distributions. In Section 3.1, we introduce multiset L-systems as an extension of the L-system modeling framework. This extension allows us to use L-systems, long an individual plant modeling paradigm, to express local-to-global algorithms for generating plant distributions as well. Sample applications of multiset L-systems are given in Sections 3.2 to 3.4. The concept and examples of the global-to-local modeling of plant distribution are presented in Section 4, which extends preliminary results reported in [7]. Conclusions are presented in Section 5.

2 Previous work

Multilevel modeling of plant communities for image synthesis purposes was introduced in [3], although related techniques had been used earlier, *e.g.* [1]. The main concept was to consider the generation of a plant ecosystem as a hierarchy of tasks: specification of the terrain, generation of plant distribution using coarse plant models, synthesis of detailed plant models as needed to populate the scene, and the rendering of the final scene using instances of these detailed models.

Two different methods were used in [3] to create plant distributions. The first one was an individual-based ecosystem simulation, based on a model of Firbank and Watkinson [4]. Following that model, simulated plants were placed in the field at random, then iteratively 'grown', and 'killed' when dominated by larger plants. The resulting distribution fit the *self-thinning curve* of plant ecology [8], a relationship between the average mass and average density in a monoculture of plants of the same age. The individual-based approach was also used in [3] to produce a hierarchy (distribution) of plant sizes (*c.f.* [8]), similar to that observed in nature.

The second method was intended to allow more user control in defining the local density of plants. The input was a greyscale image representing a map of the density of plants throughout the field. The Floyd-Steinberg error diffusion algorithm [5] was used to create the positions of individual plants conforming to these densities. The points produced by this algorithm were slightly jittered to make the distribution appear more random.

These two methods exemplify two different approaches to ecological modeling. The individual-based simulation is representative of the local-to-global approach. It is characterized by the emergence of global features from the local interactions of individual plants. In contrast, the error diffusion method is an example of the inverse, global-to-local approach, in which local characteristics are derived from global properties of the distribution. This distinction is similar to the distinction between local-to-global and global-to-local methods used to model individual plants [14].

The methods described in [3] tend to create uniform plant distributions. In reality, however, plants often are clustered. *Clustering*, also known as *clumping* or *underdispersion* [2], is a common phenomenon, caused by environmental factors (plants of the same type tend to cluster in the areas favorable to their growth), propagation (seeds fall close to their parent plants, or plants propagate by runners), as well as other mechanisms. It has a significant impact on the appearance of plant distributions, which is why we are seeking to model it.

The effect of clustering can be quantified using several statistical measures. We use the *Hopkins index* [6], which is defined as the average distance from a randomly chosen point to its nearest plant within a given region, divided by the average distance from a randomly chosen plant to its nearest plant:

$$H = \frac{\langle min_i(\|x - p_i\|)\rangle_x}{\langle min_i(\|p_j - p_i\|)\rangle_j}.$$

Distributions that are completely uncorrelated ('random') have an H value of 1. Distributions that are more dispersed than random ('regular') have an H value less than 1, and distributions that are clustered have an H value greater than 1. For example, Figure 1 compares an overdispersed distribution with a Hopkins index of 0.4 and a clustered distribution with a Hopkins index of 2.4.

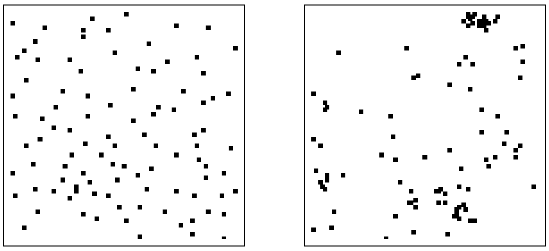

Figure 1: The effect of clustering on plant distribution. Left: an overdispersed distribution with H = 0.4. Right: a clustered distribution with H = 2.4.

3 Local-to-global modeling of plant distribution

3.1 Multiset L-systems

We model the individual plants using L-systems [13] and a related technique based on Chomsky grammars [14]. In addition, we extend the L-system formalism to generate plant distributions using the local-to-global approach. To this end, we introduce the notion of *multiset L-systems*.

An L-system model generates plants represented as strings of symbols [9] with optional parameters [13]. These strings define both the topology and the geometry of the resulting structures. An L-system specification consists of three components: the *alphabet*, which is the set of symbols that represent distinct components of the plant; the *axiom*, which represents the initial state of the modeled structure; and a list of *productions*, which define the development of the plant's components over steps of time. The alphabet may be defined implicitly, as the set of symbols that appear in the productions. The development of a plant is simulated in a sequence of *derivation steps*. In any step, each symbol is rewritten using the first applicable production on the list (or rewritten into itself if no production applies), yielding a new string. An extension of L-systems called *pseudo-L-systems* [11] makes it possible to rewrite two or more symbols using a single production. Another extension, called *open L-systems* [10], makes is possible to capture the interactions between the modeled plants and their environment.

Multiset L-systems unify and extend to branching structures two previously defined notions of the L-system theory: developmental systems with finite axiom sets [16] and L-systems with fragmentation [17]. In multiset L-systems, the set of productions operates on a multiset of strings that represent many plants, rather than a single string that represents an individual plant. New strings can be dynamically added to or removed from this multiset, representing organisms that are added to or removed from the population.

Formally, a context-free non-parametric multiset L-system is a four-tuple $G = \langle V, \%, \Omega, P\rangle$ where V is the *alphabet* (a finite set of symbols), $\% \notin V$ is a reserved *fragmentation symbol*, $\Omega \subset V^\star$ is a finite set of words over V called the *axiom*, and $P \subset V \times (V \cup \{\%\})^\star$ is a finite set of *productions*. The alphabet V may contain, in particular, a pair of brackets, [and], which are used to delimit branches in the *bracketed string* notation of tree structures [13].

A derivation step in a multiset L-system consists of two sub-steps. First, all words x_i in the predecessor multiset are replaced by the intermediate successor words y_i using productions in P. The individual derivations $x_i \rightarrow y_i$ are performed as in an ordinary L-system. Second, the words y_i that contain one or more fragmentation symbols % are subdivided. In this process, symbol % acts as the marker of positions at which branches y_{ik} are cut off the tree y_i. The remaining part of the tree y_i and the cut off branches y_{ik} become the members of the successor multiset.

For example, let us consider the multiset L-system specified below.

Alphabet:	{A, B, I, [,] }
Axiom:	{ A, B }
Productions:	1. A → I[B]A
	2. B → B%A

Starting with the axiom, the first two derivation steps yield the multisets listed in the Table 1.

step	intermediate multiset	final multiset
0	{A, B}	{ A, B }
1	{ I[B]A, B%A }	{I[B]A, B, A }
2	{I[B%A]I[B]A, B%A, I[B]A }	{ I[B]I[B]A, A, B, A, I[B]A }

Table 1: Operation of a sample multiset L-system

Extensions of L-systems, such as pseudo-L-systems and open L-systems, also apply to the multiset L-systems. In particular, in the simulations of ecosystems we rely extensively on the *communication symbols* ?E, introduced in [10] as a part of the open L-system formalism. The communication symbol is a vehicle for information exchange between plant models and their environment. It can be associated with one or more parameters, which are set by the environmental program interfaced with the L-system-based simulator.

The plant models used in the ecosystem simulations are extremely simplified, in order to accommodate a large number of plants. We have used the L-system-based plant modeling software `L-studio/cpfg` [12], extended with multiset capabilities, to both generate plant distributions and model the individual plants.

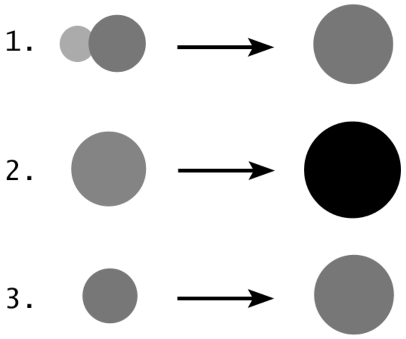

Figure 2: Diagrammatic representation of a model of self-thinning. Dark grey circles represent growing plants, the light grey circle represents a dominated plant, and the black circle represents a mature plant that no longer grows.

3.2 Self-thinning

As the first illustration of the concepts described above, let us consider a multiset L-system implementation of the individual-based self-thinning model outlined in [3]. Self-thinning takes place among a group of plants of the same species and age. As the plants grow and compete with each other for resources, smaller and weaker plants become *dominated* by larger, stronger plants, and eventually die. The essence of this process can be captured using the set of rules shown in Figure 2. The corresponding L-systems is given below:

Axiom: { T($\vec{x}_1$,r_1)?E(1) ,
T($\vec{x}_2$,r_2)?E(1) ,
... ,
T($\vec{x}_n$,r_n)?E(1) }

1. T($\vec{x}$,r)?E(c) : c == 0 → ϵ
2. T($\vec{x}$,r) : $r \geq R$ → T($\vec{x}$,R)
3. T($\vec{x}$,r)?E(c) → T($\vec{x}$,r + grow($r, \Delta t$))

Each plant is described by module T($\vec{x}$,r) followed by the communication module ?E(c). Vector $\vec{x}$ and number r represent position and size (shoot radius) of the plant. Parameter c is used for communication with the environmental process, which sets c to 1 if the plant is not dominated and to 0 if it is dominated. The environmental process considers each plant as a circle of a radius r, and determines which circles are intersecting. The smaller of any pair of intersecting circles is considered dominated.

The axiom introduces n plants with random positions and sizes (the initial distribution of plants could also be generated algorithmically). The first production, guarded

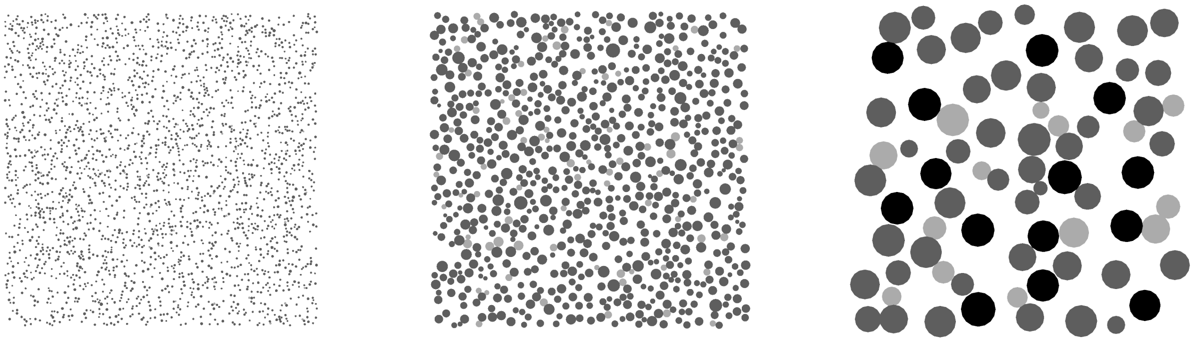

Figure 3: Three stages of simulation of the self-thinning process. Dark grey circles are growing plants, light grey circles are dominated plants, and black circles are mature plants, as in Figure 2.

by the condition c == 0, removes any dominated plant and its associated communication module from the population. Production 2 stops the growth of a plant that has reached its the maximum size R. Finally, production 3 increases the size of a plant that is neither dominated nor mature. The user-defined function grow($r, \Delta t$) captures growth of a plant of radius r over time interval Δt.

Figure 3 shows three stages of a self-thinning process simulated using this L-system. As the plant community develops over time, dominated plants gradually disappear and thin out the distribution. Sample visualizations obtained by substituting realistic plant models for the individual circles are shown in [3].

3.3 Plant succession

An extension to the previous L-system transforms it into a model of interaction between two plant species:

```
Axiom: { X }

1.  X → T($\vec{x}_1$,$r_1$,1)?E(1) %
        ...
        T($\vec{x}_n$,$r_n$,1)?E(1) %
        T($\vec{x}_{n+1}$,$r_{n+1}$,2)?E(1) %
        ...
        T($\vec{x}_{n+m}$,$r_{n+m}$,2)?E(1) % X

2.  T($\vec{x}$,$r$,$sp$) > ?E(c) : c == 0 &&
        random(1) < shaded[$sp$] → T($\vec{x}$,$r$,$sp$)
3.  T($\vec{x}$,$r$,$sp$) ?E(c) : c == 0 → $\epsilon$

4.  T($\vec{x}$,$r$,$sp$) : $r \geq R$ && random(1) < oldage[$sp$]
        → T($\vec{x}$,$R$,$sp$)
5.  T($\vec{x}$,$r$,$sp$) : $r \geq R \rightarrow \epsilon$

6.  T($\vec{x}$,$r$,$sp$) → T($\vec{x}$,$r$ + grow($r$,$sp$, $\Delta t$),$sp$)
```

In this model a plant is represented by the module T($\vec{x}$,r,sp). Parameters $\vec{x}$ and r denote the plant's position and radius, as in the previous model. Parameter sp is the plant's species identifier, either 1 or 2. Production 1 adds n new plants of species 1 and m new plants of species 2 to the population. The production predecessor X reappears in the successor multiset, thus new plants are added in every simulation step. Productions 2 and 3 remove a dominated plant with probability $1 - \mathrm{shaded}[sp]$[1]. The value shaded[sp], called the *shade tolerance* of the plant, is a measure of how likely it is to survive in shadow.

Productions 4 and 5 model the senescence of plants. Once a plant has reached the radius R, it survives with the probability oldage[sp]; a plant that does not survive dies and is removed from the community. Production 6 uses the growth function grow(r,sp,Δt) to simulate the growth of plants that are neither dominated nor old, according to their size and species.

With the right parameterization, this model captures the phenomenon of *succession* [8]. If species 1 has a higher growth rate but lower shade tolerance and old-age survivorship than species 2 (grow(r,1Δt) $>$ grow(r,2,Δt), shaded[1] $<$ shaded[2], oldage[1] $<$ oldage[2]), then an initially empty field will be populated in stages. First, the field will be dominated by species 1. As the largest members of species 1 die, smaller members of species 2, which have survived due to their greater shade tolerance and now have a size advantage over young seedlings of species 1, will fill in the gaps. Eventually, the field will be dominated by members of species 2. A straightforward extension of this model to three plant species is illustrated in Figure 4.

[1] Recall that the production list is ordered, thus the string rewriting mechanism will first attempt to apply production 2, and only use production 3 if the condition of production 2 is not satisfied.

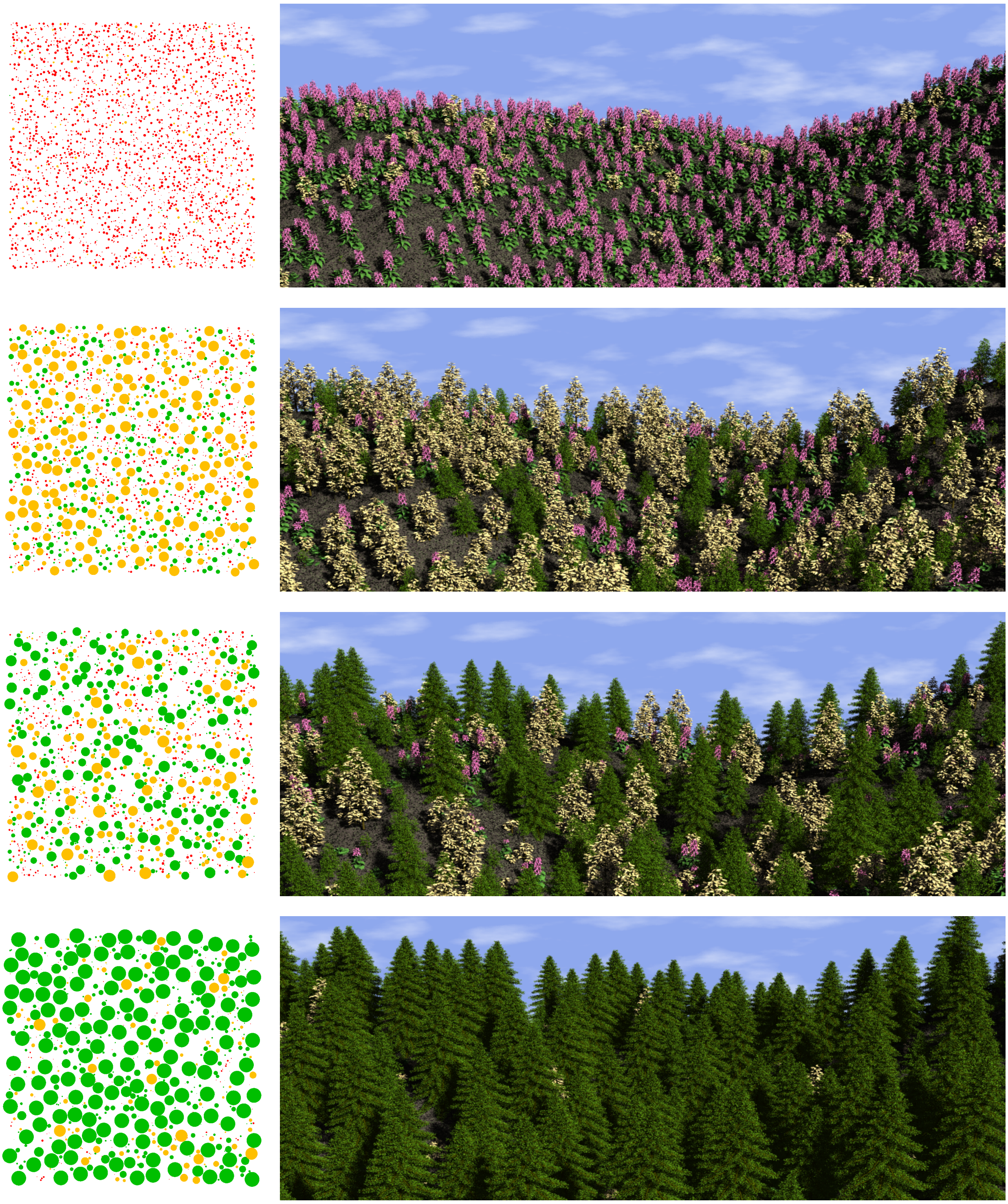

Figure 4: Four stages of ecosystem simulation using the plant succession model. Left: results of coarse-level simulation, using pink circles to indicate position of herbaceous plants (fireweed), orange circles to indicate positions of the early-succession deciduous trees, and green circles to indicate positions of the late-succession coniferous trees. Right: synthetic images obtained by placing tree models at the locations generated by the coarse-level simulation.

3.4 Plant propagation

The evaluation of the Hopkins index for the distributions shown in Figure 3 yields values of H equal to 0.8, 0.4, and 0.4, respectively. Similarly, the evaluation of the Hopkins index for the distributions in Figure 4 results in H values of 0.6, 0.7, 0.7, and 0.6. This shows that the competition for space leads to overdispersed plant distributions.

We can see why this is the case. If any two plants so much as touch each other, one of them will become dominated. In our self-thinning model, the dominated plant immediately dies; in the succession model, it dies with some probability per derivation step. In either case, the competition for space drives the plants apart, and there is no opposite mechanism encouraging plants to cluster.

One clustering mechanism observed in nature is local propagation. We can capture it, for instance, by 'sowing' new plants near the parent plants of the same species, instead of making them appear at random throughout the field. The resulting alteration of the succession model is given below.

```
Axiom: { T($\vec{x}_1$,$r_1$,1)?E(1) ,
             ... ,
             T($\vec{x}_n$,$r_n$,1)?E(1) ,
             T($\vec{x}_{n+1}$,$r_{n+1}$,2)?E(1) ,
             ... ,
             T($\vec{x}_{n+m}$,$r_{n+m}$,2)?E(1) }

1.   T($\vec{x}$,$r$,$sp$) > ?E(c) : c == 0 &&
         random(1) < shaded[$sp$] $\rightarrow$ T($\vec{x}$,$r$,$sp$)
2.   T($\vec{x}$,$r$,$sp$) ?E(c) : c == 0 $\rightarrow \epsilon$

3.   T($\vec{x}$,$r$,$sp$) : $r \geq R$ && random(1) > oldage[$sp$]
         $\rightarrow$ T($\vec{x}$,$R$,$sp$)
4.   T($\vec{x}$,$r$,$sp$) : $r \geq R \rightarrow \epsilon$

5.   T($\vec{x}$,$r$,$sp$) ?E(c) $\rightarrow$ T($\vec{x}$,$r$ + grow($sp$,$r$,$\Delta t$),$sp$) ?E(c)
         % T($\vec{x} + \Delta\vec{x}$,$r_0$,$sp$) ?E(c)
```

The axiom defines the initial state of the model by placing n plants of species 1 and m plants of species 2 at random in the field. The subsequent productions are the same as in the succession model, except for production 5. According to it, a plant that is not dominated creates a new plant at position $\vec{x} + \Delta\vec{x}$, where $\Delta\vec{x}$ is a small random vector. Since the new plant is in close proximity to its parent, this propagation mechanism encourages clustering in the distribution.

Figure 5 illustrates the operation of this model. At the beginning, plants are randomly distributed. As the ecosystem develops, the two species become spatially segregated, creating large clusters of plants of each species. For example, the Hopkins indices of species 1 at the three stages shown are equal to 1.1, 4.2, and 11, respectively.

4 Global-to-local modeling of plant communities

4.1 The deformation-kernel method

The effect one plant in the self-thinning model (Section 3.2) has on the probability of finding another plant nearby is shown diagramatically in Figure 6. Within the reference plant radius r_t, the probability of finding another plant is very small; outside that radius, the probability is not affected by the reference plant.

The function K shown in Figure 6 is an example of a *deformation kernel*. If we suppose there is a field of values that characterizes the probability of placing a new plant at various locations, the deformation kernel captures the impact of an existing plant on this field. Various interactions between plants can be described using deformation kernels of different shapes, as suggested in Figure 7.

A simple plant placement algorithm can now be developed using this deformation kernel idea. We maintain a *joint probability density function* [15] $f(x, y)$, which characterizes the probability $f(x, y)dxdy$ of placing a new plant in the area $dxdy$ centered at point (x, y). The plants are placed one at a time; as each is placed, its deformation kernel modifies the probability function f that will be used to determine the position of the next plant. In this manner, a distribution of plants will eventually be formed.

Formally, the joint density function f defines a probability field, where the probability of a new plant growing in the rectangle $[0, x_s] \times [0, y_s]$, with $0 \leq x_s \leq x_{max}$ and $0 \leq y_s \leq y_{max}$, is given by the cumulative probability distribution function

$$\begin{aligned} F(x_s, y_s) &= P\{x_t \leq x_s, y_t \leq y_s\} \\ &= \int_0^{x_s}\int_0^{y_s} f(x, y)\, dx\, dy. \end{aligned}$$

Obviously, the probability that the plant will be found in the whole field is one, thus the density function must satisfy the normalizing equation

$$\int_0^{x_{max}}\int_0^{y_{max}} f(x, y)\, dx\, dy = 1. \qquad (1)$$

We find the position (x_t, y_t) of the plant to be added by calculating first its y, then its x coordinate. To this end, given the two-dimensional density function $f(x, y)$, we create the *marginal distribution function* $F_Y(y_s)$. That

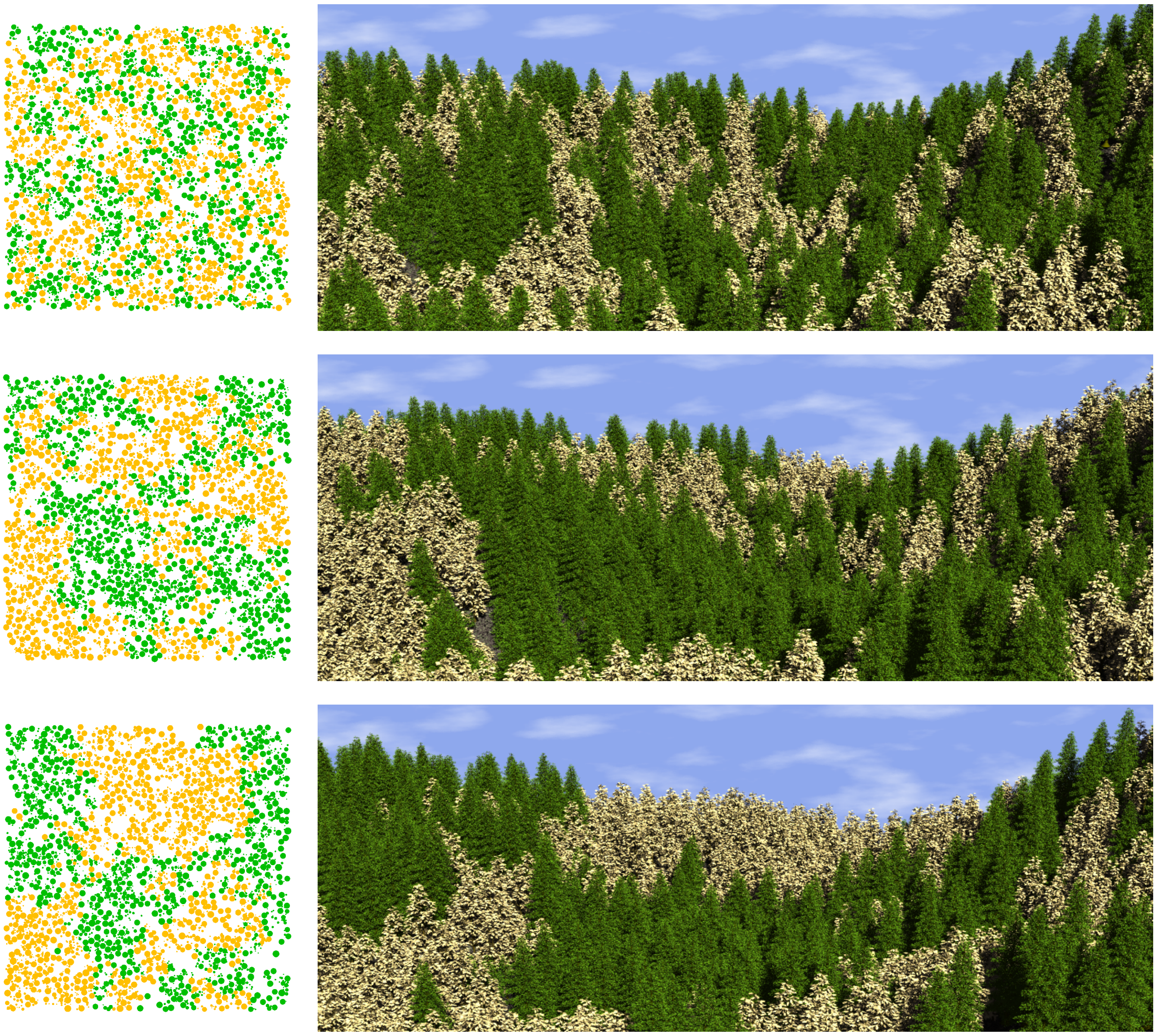

Figure 5: Three stages of ecosystem development simulated using the plant propagation model. Left: results of coarse-level simulation, using orange circles to indicate positions of poplar trees and green circles to indicate positions of spruce trees. Right: synthetic images obtained by placing tree models at the locations generated by the coarse-level simulation.

distribution function describes the probability that $y_t \leq y_s$ independently of the choice of x_t [2] :

$$F_Y(y_s) = P\{x_t \leq x_{max}, y_t \leq y_s\} = F(x_{max}, y_s).$$

We choose the y_t coordinate for the plant using the *inverse transformation method* [15]. To this end, we generate a random number u from the uniform distribution on $[0, 1]$. We then perform a binary search on $F_Y(y)$ to find the value y_t such that $F_Y(y_t) = u$. As F_Y is monotone and continuous, y_t exists and is unique. This is our plant's y coordinate.

Once we have chosen y_t, we calculate the *conditional distribution* $F_{X|Y}(x_s \mid y_t)$, which describes the probability that $x_t \leq x_s$ for a given a y value y_t:

$$F_{X|Y}(x_s \mid y_t) = P\{x_t \leq x_s \mid y_t\} = \frac{\int_0^{x_s} f(x, y_t)\, dx}{\int_0^{x_{max}} f(x, y_t)\, dx}.$$

We then apply the inverse transformation method to find coordinate x_t, given $F_{X|Y}(x_s \mid y_t)$.

[2] There is a corresponding marginal distribution function $F_X(x_s)$, which describes the probability that $x_t \leq x_s$, independent of what is chosen for y_t.

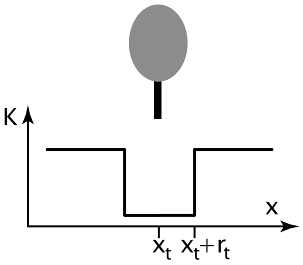

Figure 6: The effect of a reference plant on the probability of finding neighboring plants

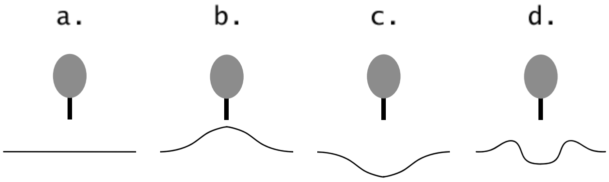

Figure 7: Examples of deformation kernels: a) kernel that has no effect on the neighboring plants, b) kernel that has a promotional effect, c) kernel that has an inhibitory effect, d) kernel that has an inhibitory short-range effect and promotional longer-range effect.

Having placed a plant of size r_t at position (x_t, y_t), we now deform the probability density function $f(x, y)$ in order to simulate the effects of that plant on the placement of nearby plants. To this end, we first multiply the probability density function $f(x, y)$ by the plant's deformation kernel $K(x, y)$,

$$f_{temp}(x, y) = f(x, y)K(x, y),$$

then renormalize the function $f_{temp}(x, y)$ to satisfy Equation 1. The deformation kernel typically is a function of the form

$$K(x, y) = \kappa\left(\frac{\sqrt{(x - x_t)^2 + (y - y_t)^2}}{r_t}\right),$$

where the function $\kappa(r)$ measures the effect a unit-sized plant has on the formation of plants at a distance r from it.

An obvious way to implement the above concepts is to represent values of the probability density function $f(x, y)$ using an n by n array of samples f_{ij}. To calculate position of a new plant, we create a vector **R** of partial sums of the rows, where

$$R_k = \sum_{i=0}^{k}\sum_{j=0}^{n-1} f_{ij}, \quad k = 0, 1, \dots, n - 1.$$

We determine the coordinate y_t of the newly placed plant using the inverse transformation method. To this end, we pick a random number u from the uniform distribution on the interval $[0, R_{n-1}]$, then perform a binary search to locate R_i such that $R_i \le u < R_{i+1}$. We then linearly interpolate between (i, R_i) and $(i+1, R_{i+1})$ to find (y_t, u).

Now a vector **C** of values representing the conditional distribution $F_{X|Y}(x_s \mid y_t)$ is computed by interpolating rows i and $i + 1$ of the array f_{ij}.

$$C_k = (y_0 - i)\sum_{j=0}^{k} f_{(i+1)j} + ((i + 1) - y_0)\sum_{j=0}^{k} f_{ij},$$

Given the values C_k, $k = 0, 1, \dots, n - 1$, we choose a value from the uniform random distribution on the interval $[0, C_{n-1}]$, and use the inverse transformation method to find x_t.

The kernel is applied by simply calculating the distance d of every sampling point (i, j) from (x_t, y_t), then multiplying the value f_{ij} of the distribution function f at that point by $\kappa(\frac{d}{r_t})$.

If there are m plants to be placed and the function f is represented using n^2 values f_{ij}, the above algorithm will take $O(mn^2)$ time to run, since the sums R_k must be recalculated each time a new plant is placed. We improve on this result by updating the array **R** incrementally. When a kernel is applied to the distribution, the differences between f_{ij} and $\kappa(\frac{d}{r_t})f_{ij}$ are summed for each (i, j) within the range of the plant, and the differences are applied to the array **R**. Assuming that the kernel is only applied to a small fraction of the cells in the grid, this operation can be performed in $O(n)$ time per plant.

The operation of the kernel method is illustrated in Figure 8. The deformation kernel is that of Figure 7d. The initial distribution is uniform; $f(x, y) = c$. In the middle, a single plant has been added to the field; the density function has been altered in the plant's neighborhood. On the bottom, four more plants have been added; the density function has been modified near each of them.

Figure 9 shows point patterns generated using this algorithm with different deformation kernels. The Hopkins indices of these patterns are 0.4, 1.0, 1.2, and 2.4, confirming the visual observation that the kernel method is capable of creating a range of distributions, from overdispersed to random and clustered.

In Figure 10 points have been replaced by simple models of daisies, created using an L-system. The overdispersed pattern at the top of Figure 10 looks less realistic than the clustered pattern at the bottom, which justifies the introduction of clustering into the model.

The distributions shown in Figure 10 have been generated after initializing the probability density function

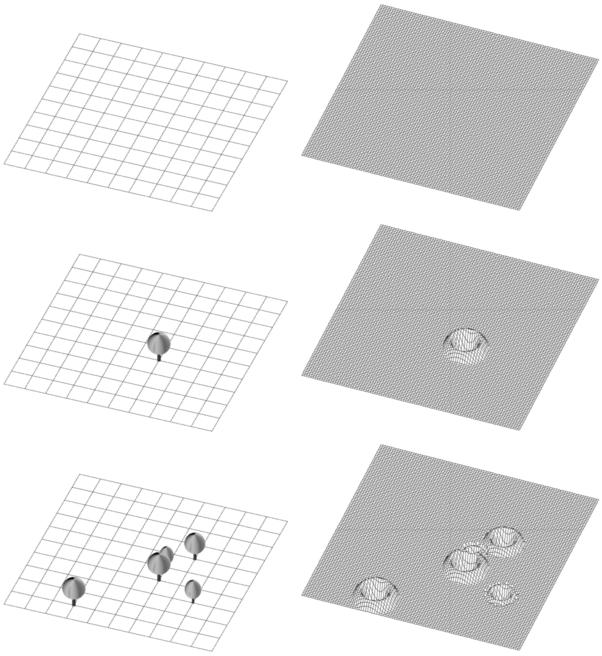

Figure 8: An example of the deformation kernel algorithm, using the kernel of Figure 7d. Left: the plant distribution. Right: the joint probability density function f. From top to bottom: the initial state, the state after placing the first plant, and the state after placing four more plants.

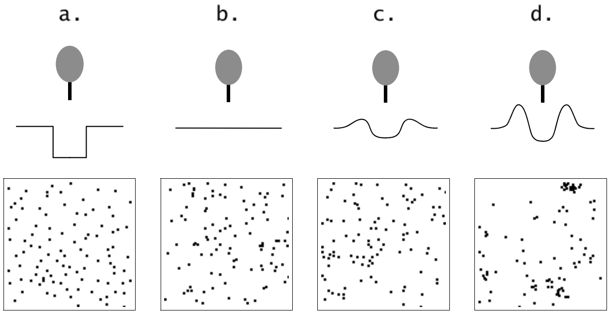

Figure 9: Some kernels and the point patterns they generate. The kernels are drawn at a larger scale than the point patterns.

f to a constant value. If, instead, we initialize f to a user-defined field (with a paint program, for example), we can generate spatial distributions of plants that conform to this field, as shown in Figure 11.

This result improves on the methodology described in [3], which allowed for the application of a user-specified density map, but did not make it possible to control the degree of plant clustering.

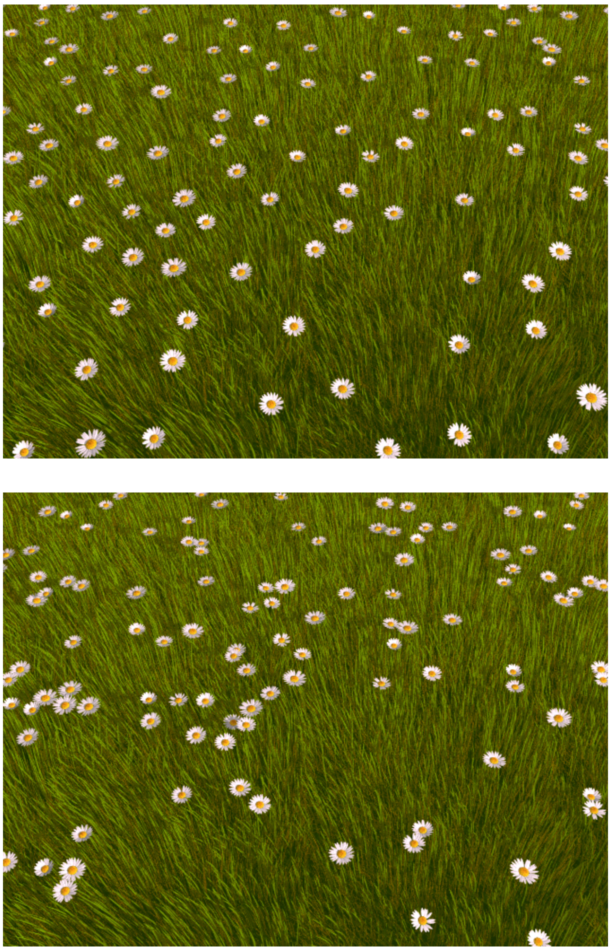

Figure 10: Point patterns corresponding to Figures 9a and 9c, rendered as fields of daisies.

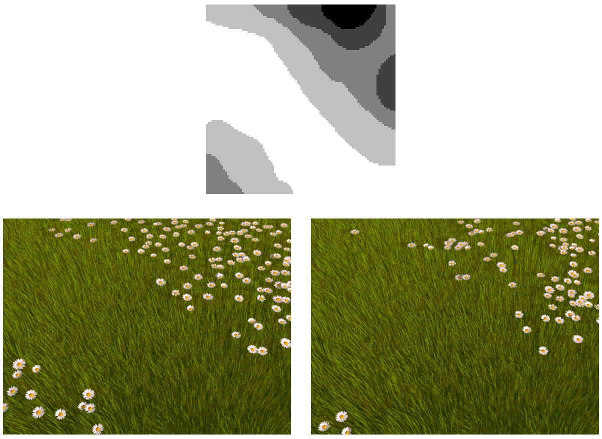

Figure 11: Plant distributions created from the user-defined density map, shown at the top. The Hopkins index values are $H \approx 1.2$ (left) and $H \approx 1.9$ (right).

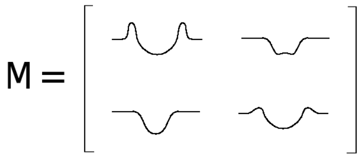

Figure 12: The concept of the kernel matrix M for two species

4.2 Extensions

The kernel-based method can be extended to include information about the plants' sizes, as well as to model the interaction of several species.

Size information is taken into account by placing plants in order of size, largest first. Larger, older plants then affect the positioning of smaller, younger plants, as it is to be expected. The actual sizes may be drawn from a distribution that gives few large plants, more plants of medium size, and still more small plants.

Plants of species 1 may have a different effect on other plants of species 1 than they do on plants of species 2. Different kernels are thus required to capture these effects. In fact, for two species, the total of four kernels is required: one for the effects species 1 has on itself, one for the effects species 1 has on species 2, one for the effects species 2 has on species 1, and one for the effects species 2 has on itself. This leads, in the general case, to a *kernel matrix* M (Figure 12), which defines the effects that each species has on itself and on each other species.

In an extension of the kernel placement method to n species, we keep track of n different probability density functions f_i. As plants are placed, each probability density function is deformed by the relevant kernel; thus if a plant of species j is placed, each function f_i is deformed by kernel $M_{i,j}$, where $i = 1, 2, \ldots, n$. This process is illustrated in Figure 13.

A sample application of the above method is presented in Figure 14. The modeling of trees incorporated in this scenes was discussed in [14].

5 Conclusions

We formalized and extended the methods for defining plant distributions that had originally been proposed in [3]. To this end, we introduced multiset L-systems, an extension of the L-system formalism, to model groups of plants, rather than single plants alone. We then applied these L-systems to simulate the essence of self-thinning, succession, and clustering of plants in an ecosystem.

We also improved the interactive techniques for generating plant distributions, making is possible to specify not only global densities, but also levels of clustering. The resulting kernel deformation method can produce plant distributions with a wide range of clustering values, from overdispersed to random to highly clustered. We have illustrated the use of this method using several models of plant communities.

Fig.	Number of			Time taken[a]		
	plants	different plants	primitives (millions)	distribution generation[b]	plants	rendering
4a	2688	20	34	9 sec	30 sec	8 min
b	1593	20	18	72 sec	27 sec	8 min
c	1409	20	30	2 min	30 sec	9 min
d	834	20	78	3 min	40 sec	11 min
5a	5584	24	141	2 min	30 sec	15 min
b	5526	24	147	22 min	30 sec	14 min
c	3427	24	100	44 min	30 sec	12 min
10	100	8	1.5	0.05 sec	2 sec	2 min
14	1599	36	147	0.5 sec	65 sec	11 min

[a] Times recorded on a 733 MHz Pentium III processor.

[b] For Figures 4 and 5, the times given show how long the simulation took to reach the given frame.

Table 2: Statistics pertinent to Figures 4, 5, 10, and 14

Both methods make it possible to generate plant ecosystems at the speeds required for their practical applications to realistic image synthesis. Statistics pertinent to the scenes included in this paper are given in Table 2.

We believe that the proposed methods can be applied to model diverse plant ecosystems and have various practical applications, such as scene dressing for computer animation purposes, and visual impact analysis of tree cutting and regrowth on the landscape. The realism of the resulting scenes can be further improved using more accurate models of the underlying biological processes, and more sophisticated rendering methods.

Acknowledgments

We thank Martin Fuhrer for the model of fireweed used in Figure 4, Lars Mündermann for the model of grass used in Figures 10 and 11, and the referees for their insightful comments. We acknowledge Radoslaw Karwowski and Radomir Měch for the development of the `L-studio/cpfg` plant modeling software, and Craig Kolb for the development of the `rayshade` ray tracer, which we used to render the scenes. The partial support of this research by grants from the Natural Sciences and Engineering Research Council of Canada is also gratefully acknowledged.

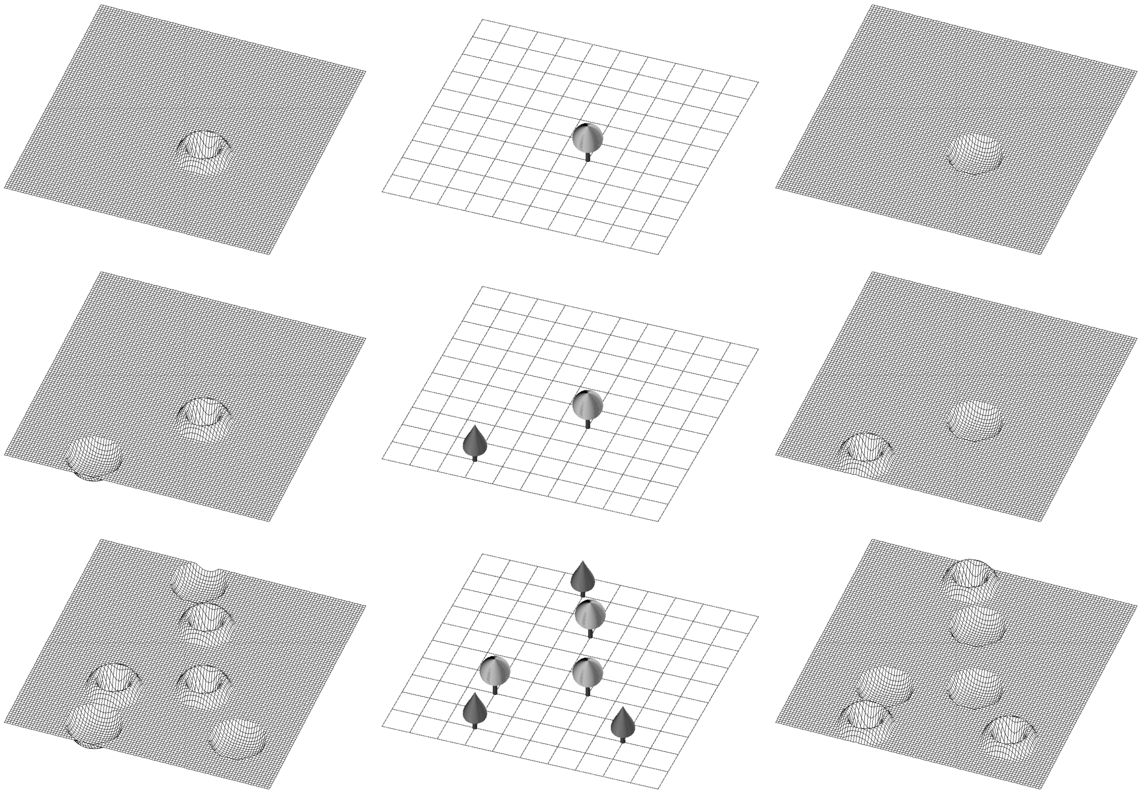

Figure 13: Generation of a two-species distribution using the deformation kernel algorithm with the kernel matrix shown in Figure 12. On the left, the probability density function of species 1; on the right, that of species 2. From top to bottom: the state of the system after placing the first plant of species 1, after placing the first plant of species 2, and after placing several plants of each species.

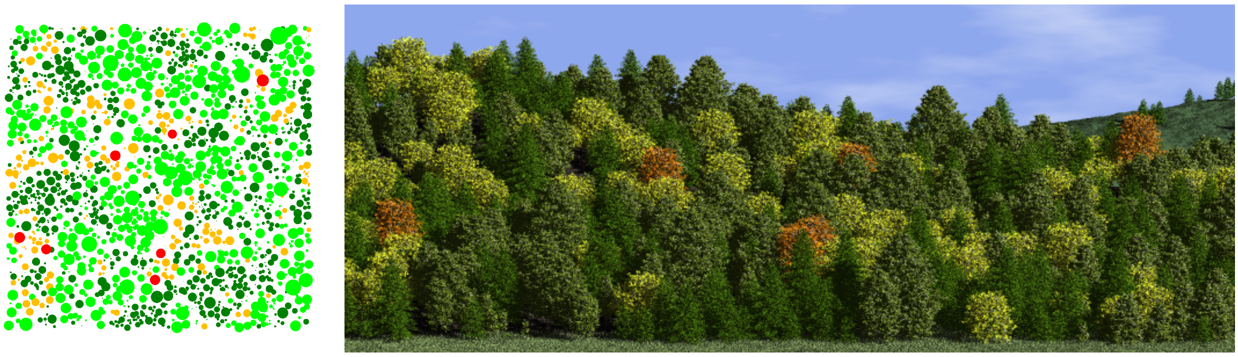

Figure 14: A forest model consisting of four species of trees. Left: distribution of plants. Right: realistic visualization of the model.

References

[1] N. Chiba, K. Muraoka, A. Doi, and J. Hosokawa. Rendering of forest scenery using 3D textures. *Journal of Visualization and Computer Animation*, 8(4):191–199, 1997.

[2] M. R. T. Dale. *Spatial Pattern Analysis in Plant Ecology*. Cambridge Studies in Ecology. Cambridge University Press, Cambridge, UK, 1999.

[3] O. Deussen, P. Hanrahan, B. Lintermann, R. Měch, M. Pharr, and P. Prusinkiewicz. Realistic modeling and rendering of plant ecosystems. *Proceeding of SIGGRAPH 98 (Orlando, Florida, July 19-24, 1998)*, pages 275–286, 1998.

[4] F. G. Firbank and A. R. Watkinson. A model of interference within plant monocultures. *Journal of Theoretical Biology*, 116:291–311, 1985.

[5] R. W. Floyd and L. Steinberg. An adaptive algorithm for spatial greyscale. *Proceedings of the Society for Information Display*, 17:75–77, 1975.

[6] B. Hopkins. A new method for determining the type of distribution of plant individuals. *Annals of Botany*, XVIII:213–226, 1954.

[7] B. Lane and P. Prusinkiewicz. Randomized generation of nonuniform plant distributions. *Proceedings of Western Computer Graphics Symposium 2000*, pages 61–66, 2000.

[8] L. Legendre and P. Legendre. *Numerical Ecology*. Elsevier, Amsterdam, 1983.

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

[10] R. Měch and P. Prusinkiewicz. Visual models of plants interacting with their environment. *Proceedings of SIGGRAPH 1996*, pages 397–410, August 1996.

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

[12] P. Prusinkiewicz, R. Karwowski, R. Měch, and J. Hanan. L-studio/cpfg: A software system for modeling plants. In M. Nagl, A. Schürr, and M. Münch, editors, *Applications of graph transformation with industrial relevance*, Lecture Notes in Computer Science 1779, pages 457–464. Springer-Verlag, Berlin, 2000.

[13] 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.

[14] P. Prusinkiewicz, L. Mündermann, R. Karwowski, and B. Lane. The use of positional information in the modeling of plants. *Proceeding of SIGGRAPH 2001 (Los Angeles, California, August 11-17, 2001)*, pages 289–300, 2001.

[15] S. M. Ross. *Introduction to Probability Models*. Academic Press, 1997.

[16] G. Rozenberg and K. P. Lee. Developmental systems with finite axiom sets. Part I. Systems without interactions. *International Journal of Computer Mathematics*, 4:43–68, 1974.

[17] G. Rozenberg, K. Ruohonen, and A. Salomaa. Developmental systems with fragmentation. *International Journal of Computer Mathematics*, 5:177–191, 1976.