An Efficient Instantiation Algorithm for Simulating Radiant Energy Transfer in Plant Models

Cyril Soler, François X. Sillion, Frédéric Blaise, Philippe de Reffye

An Efficient Instantiation Algorithm for Simulating Radiant Energy Transfer in Plant Models

Cyril SOLER and François X. SILLION
iMAGIS-GRAVIR/IMAG-INRIA
and
Frédéric BLAISE and Philippe DEREFFYE
CIRAD/INRIA

We describe a complete lighting simulation system tailored for the difficult case of vegetation scenes. Our algorithm is based on hierarchical instantiation for radiosity and precise phase function modeling. It allows efficient calculations both in terms of computation and memory resources. We provide an in-depth description and study of the instantiation-based radiosity technique and we address the problems related to generating and managing phase functions of plant structures, as needed by the instantiation process. We present results demonstrating the high performance of the hierarchical instantiation algorithm and we describe two examples of applications : rendering of large vegetation scenes and plant growth simulation. Other applications of our system range from landscape simulation to agronomical and agricultural studies, and to the design of virtual plants responding to their environment.

Categories and Subject Descriptors: I.3.7 [**Three-Dimensional Graphics and Realism**]: Radiosity; I.3.2 [**Methodology and Techniques**]: Graphics data structures and data types; I.6.3 [**Simulation and Modeling**]: Applications—*Plant Growth Simulation*

General Terms: Algorithms, Measurement, Performance

Additional Key Words and Phrases: Plant growth simulation, lighting simulation, radiosity, instantiation, landscape simulation, calibrated physiological simulation

1. INTRODUCTION

Three-dimensional scenes containing plants and vegetation elements are usually of tremendous complexity, typically consisting of millions of elements. Therefore they constitute extremely challenging cases for rendering and simulation techniques, and have indeed been used extensively as test scenes to push all sorts of algorithms to their limits.

Still, the ubiquitous presence of vegetation around us, even in artificial spaces such as office buildings, makes it necessary to be able to render and model plants efficiently. Although rendering can be performed in many ways, including non-photorealistic algorithms [Deussen and Strothotte 2000], we observe that accurate lighting simulation in plant models has a number of applications:

First, photorealistic rendering of vegetation scenes can be achieved by a correct simulation of light energy exchange inside plants models. This is obviously useful for visual applications in various fields like cinema, urban and architectural design.

Plant growth simulation is an other challenging application of lighting simulation in vegetation scenes for at least two reasons : (1) The creation of realistic plant models, is required by many applications in very diverse fields. In computer graphics, we have seen numerous examples of beautiful renderings of trees, flowers and other plants, which tremendously add to the realism of virtual scenes. Plant models however are very complex, making it tedious to create them by hand. Conversely, *simulating* plant growth offers the perspective of being able to build, study and render virtual plants in their specific environment while controlling their development. (2) Research in agronomy concerning the influence of light over the production of a cultivated crop under different conditions of illumination requires complex settings and long term experiments. Instead, properly simulating the growth of plants readily gives results while allowing control over many growth parameters simultaneously.

In this paper we present a complete lighting simulation algorithm tailored for vegetation scenes. This algorithm addresses the two principal difficulties encountered with vegetation, which usually result in unacceptably high computation or memory costs: first, the intrinsic geometric complexity of plant models, consisting of millions of disconnected elements, including small, elongated structures. Second, the photometric complexity of light transfer within vegetation models, with diffusion inside foliage, complex BRDFs and leaf transparency effects.

Though many solutions have been proposed in the past, we shall see that they each have a limited range of applicability in terms of scene sizes. The solution we propose has the double advantage of being applicable to a broad range of scene complexities, and offering a continuous trade-off between accuracy and computation/memory cost.

Hierarchical radiosity algorithms, especially those using clustering, try to avoid considering the inherent complexity of energy exchanges by computing transfers at fairly high levels of a scene hierarchy. However a complete traversal of the scene is needed to estimate the energy emitted or received by a cluster with any accuracy [Smits et al. 1994]. The use of meta-objects, or impostors, has been proposed to avoid this descent in the hierarchy, and instead perform the computation with fairly large (and simple) objects [Rushmeier et al. 1993; Ouhyoung et al. 1996].

We show in this paper however, that in the context of hierarchical radiosity, using such meta-objects poses additional problems, which we solve by pre-computing high level *phase functions* (reflectance) and *transmittance functions* for those meta objects. Obviously, these characteristics are quite costly to handle, both in terms of computation time and storage. Meta-objects are therefore especially useful when a sufficient number of similar objects are present in the scene, *i.e* when these objects can share the same intrinsic characteristics.

In summary, the process of using meta-objects, which we call *instantiation*, is a key element in making the accurate characterization of simplified objects viable. It is realized by identifying elements in the scene (in fact, clusters) that share a similar behavior in terms of light emission, reflection and transmission. An important requirement is thus to identify the degree of similarity between the light properties (reflectance and transmittance) of plant structures. We discuss this issue and propose solutions in Section 4.1. Note that the radiometric behavior of an instance can therefore be an approximation of reality, just as the geometry of a classical impostor is an approximation to that of the original. A flexible trade-off is therefore possible between the accuracy of the representation and its compactness, largely controlled by the degree of self-similarity in the scene.

The algorithm we are presenting is easily controllable: a computation time *vs.* accuracy trade-off can be readily performed by acting on the refinement threshold of the radiosity links. Another aspect of the controllability is that we can limit the in-depth traversal of the instance hierarchy to a given size of structures and thus obtain a high level solution at a very low computation cost. This will be discussed in section 6. However, the accuracy of this solution is still much better than that of a clustering algorithm that we would limit to large scale clusters, because of the precise reflectance and transmittance information used at the instance level. This proves very interesting, for instance, when using lighting results in a plant growth simulator, which may be satisfied by the knowledge of the illumination received at the level of entire branches, with no need to perform the computation down to the level of individual leaves.

The present paper builds on the hierarchical instantiation work presented in [Soler and Sillion 2000], with a particular emphasis on the following issues : (1) automatic construction of instances in plants models; (2) pre-computation and storage of radiometric information; (3) instantiation policy; (4) application to plant growth simulation; (5) application to rendering of large vegetation scenes.

2. PREVIOUS WORK

We successively review in this section the various methods and geometry representations that serve the computation of the distribution of light energy in plants, and their possible applications.

Illumination models

Computing the distribution of light energy in vegetation constitutes a very challenging task. To achieve it, many methods have been derived, most of them coming from the field of agronomic research. We have sorted them in increasing order of complexity.

A number of methods estimate direct illumination in the plant possibly using attenuation factors, but without accounting for light scattering inside the vegetation. The simplest binary ray-casting approach can be done very efficiently by projecting the geometry of organs along sampled hemisphere directions, as proposed by [Fournier and Andrieu 1999; Chen et al. 1993; Pearcy and Sims 1998; Planchais and Sinoquet 1996]. A more complete approach consists in casting rays toward the sky and through the geometric model [Perttunen et al. 1996; Takenaka 1994] or a voxel representation thereof [Greene 1989]. Měch et al. [Měch and Prusinkiewicz 1996] extended this technique by accumulating the opacity of voxels successively encountered by a ray to account for the translucency of the foliage.

Global illumination techniques [Kajiya 1986] have also been used to compute the distribution of light energy in plants. Among these, we distinguish radiosity-like methods and Monte-Carlo methods, from methods based on differential radiance transfer equations. Whereas the former stay arbitrarily close to the very geometry of the scene and the solution, the latter act on an equivalent *turbid medium* and depend on various approximations concerning its isotropy, homogeneity or periodicity [Verhoef 1984; Gastellu-Etchegorry et al. 1996]. As an example, Max [Max et al. 1997] proposes a simplification of the radiant transfer equations in order to compute the density of light for each altitude in an infinite canopy, provided that it is horizontally isotropic.

Although radiosity techniques are often quite costly, they produce faithful results and many approximations of the radiosity method have been used up to now: Goel et al. obtain a radiosity solution in a corn field using periodicity assumptions [Goel et al. 1991], which reduces the number of form factors to compute with the neighboring polygons of each plant. The use of standard radiosity on a pure geometric model also limits this approach to scenes with a small number of polygons. Borel et al. [Borel et al. 1991] propose to set form factors of distant objects to 0. This introduces a bias in the solution, but enables to handle large scenes. Chelle et al.'s *nested radiosity* algorithm [Chelle et al. 1998] uses a geometric model for the local neighborhood of a polygon and a volumetric model using scattering equations for distant geometry. This requires isotropy assumptions on distant parts of the canopy and periodicity of the model. Such work is consequently applicable only to large-scale scenes (such as plant canopies).

Monte Carlo methods have been used by Ross and Marshak [J.K. and A.L 1988] and Govaerts [Govaerts 1995] to estimate the canopy bi-directional reflectance function. These techniques work well for BRDF computation because they do not need to save the distribution of light inside the model. Dauzat and Eroy [Dauzat and M.N. 1987] use it to estimate the light received by the leaves of plants taking into account of internal light scattering.

Of course several intermediate methods have been used. One example is given by Gastellu et al. [Gastellu-Etchegorry et al. 1996] who add a direct illumination component due to the sun, to a multiple-scattering solution obtained using a spherical harmonic representation.

Representation of vegetation

We review the various models used for representing the vegetation in the lighting simulation algorithms. The most precise models in term of scene geometry are based on the very geometry of the plants. Radiosity based techniques intrinsically employ this representation, although some of them, combine it with a simpler model to compute distant interactions [Chelle et al. 1998]. However, working on a geometry-based description of the scene is the cause of a very high memory cost, which is one common drawback of radiosity-based methods.

Simplified, shape-preserving representations such as ellipsoids and cylinders have been used by Balandier and Norman [Balandier et al. 2000; Norman and Jarvis 1975]. Here, the topology of the plants is partially preserved but not the geometry. As pointed in [Gastellu-Etchegorry et al. 1996], approaches that transform the very geometry of the plants are not suitable for precise computation of parameters of the canopy models like reflectance functions, due to the strong anisotropic nature of the models with respect to light reflection.

Less faithful to plant geometry is a voxel-based representation of the scene. This is used for instance by Castro [Castro and Fetcher 1998] and Whitehead [Whitehead et al. 1990]. Voxels can be used to store elements of the scene (walls, water, soil) in addition to leaf density coefficients for the vegetation.

Finally, some approaches consider the vegetation as a turbid medium [Ross 1981]. They totally ignore the topology and geometry of plants as well as the very local variations of their light properties, and produce an adequate solution at larger scales. The medium properties are represented as density functions over which isotropy assumptions are usually made in order to limit the number of equations.

Applications

The first application of lighting simulation in plant models (which is also the most familiar to computer graphics) is rendering. For this, the scene geometry has to be accessed at least once, but does not necessarily serve the computation of multiple-scattering light in the model, as shown by Max [Max et al. 1997].

Remote sensing of the environment is a very important application to light simulation in plants. Light computation serves the interpretation of remote sensing data like satellite images for the computation of biophysical parameters [Goel 1988]. Many references on this subject can also be found in the survey by Myneni et al. [Myneni et al. 1989].

Plant growth simulation is a direct application to lighting simulation in plants in the sense that the growth of a plant depends on the presence of light for photosynthesis. From an agronomic point of view, growth simulation under various lighting conditions permits to study optimal culture configurations. Fournier and Andrieu give an example of such simulation on corn [Fournier and Andrieu 1999]. Other physiological plant growth simulation models exist [Rauscher et al. 1990; Takenaka 1994; Blaise et al. 1998; (de) Reffye et al. 1999], all accounting quantitatively for the light received by leaves to compute the growth rate.

Plant growth simulation may finally be used to study the reaction of plant growth to light environment. The causal effect of light on plant morphology and growth has been demonstrated on real plants but requires tedious in-the-field experiments [Beaudet and Messier 1998; Beaudet et al. 2000]. It is therefore much more interesting to perform the same experiments on virtual plants using a calibrated growth model. Gautier et al., for instance dedicate their work to the study of the influence of self-shadowing on plant organs morphogenesis [Gautier et al. 2000].

Discussion

Direct lighting approaches do not account for the contribution of light scattering inside the model due to the essentially diffuse translucency of the leaves [Govaerts 1995]. However light scattering in plant foliage represents a definite part of the illumination and thus on the growth and architecture of plants (see Section 6).

Like all stochastic methods, Monte Carlo approaches have two drawbacks: they converge very slowly and the accuracy of the result is not easily controllable. Vegetation indeed contains very uncorrelated polygons, and thus induces a large dispersion of rays hence an especially noisy and slow convergence. This is particularly true when performing the computation near to the infra-red wavelength, where transmittance and reflectance of plant leaves can both approach 50%. This boosts up the number of necessary reflections/transmissions to consider along each ray. This make stochastic methods poorly suited to an interactive work in hand with a plant growth engine.

Finally, none of the methods reviewed above can adapt to a wide range of scene scales. Turbid medium and voxel-based methods are limited to large scale scenes because of their statistical description of the plants. Conversely, geometry-based methods are restricted to small-scaled scenes mainly because of their high memory cost. The consequence of this is that none of these methods can be used for long term plant growth simulation. Indeed, when growing a plant from a seed up to a tree, a varying number of scales must be considered.

In this paper, we propose a global illumination approach for computing light energy balance in plant models that attempts to fill these gaps. Our method is based on hierarchical radiosity with clustering, hence inheriting the controllability and multi-resolution facilities of this method. In order to get rid of the traditionally high computation and memory costs of hierarchical radiosity methods, we develop the idea of instantiation, *e.g* the ability to share geometric information between parts of the scene in order to gain memory.

3. HIERARCHICAL INSTANTIATION FOR RADIOSITY

3.1 Hierarchical radiosity with clustering

This section intends to provide the relevant background to readers who are not familiar with hierarchical radiosity with clustering. Experienced readers can directly jump to section 3.2.

We denote by $b(x)$ the *radiosity* at any point x in a scene, *i.e* the total light power per unit area out-coming from x on a surface. We call $e(x)$ the emittance at x, which denotes the corresponding quantity directly emitted at x, *e.g* $e(x)$ is not null only when x is on a light source. Let us finally call $\rho(x)$ the reflectance at x, *i.e* the proportion of the incident energy flux at x which is reradiated. These quantities are linked together by the *radiosity equilibrium equation* [Goral et al. 1984] :

$$b(x) = e(x) + \rho(x) \int b(y)G(x,y)dy$$

where $G(x,y)$ denotes a kernel function accounting for the geometric configuration which characterizes the energy exchange between points x and y. This equation signifies that the light energy at a point x on a surface is the sum of the emitted energy at x (the $e(x)$ term of the sum) and the energy coming from all other points y of the scene which reflects at x (the integral term of the sum).

Radiosity methods in general constitute an approach for solving this equation using finite elements [Goral et al. 1984; Ashdown 1994; Sillion and Puech 1994]. The most common approach consists in looking for a piecewise constant approximation of the solution of the above equation. A discretization of all surfaces is performed in order to produce a linear system :

$$B_i = E_i + \rho_i \sum_j F_{ij} B_j$$

where B_i, E_i and R_i respectively denotes the uniform radiosity, emittance and reflectance value over surface element i. In this linear system, also appear the F_{ij} terms, called *form factors*, which express the contribution of each element j to the radiosity on element i.

Solving this system using classical methods such as *Gauss-Seidel* iteration becomes very costly for even simple configurations, which has stimulated the development of hierarchical approaches. The idea of hierarchical radiosity is to hierarchically group together surface elements into larger surface elements and scene objects into *clusters* [Hanrahan et al. 1991; Goldsmith and Salmon 1987; Smits et al. 1994; Sillion 1995] (see Figure 1). The energy exchanges between all pairs of surface elements in the scene can then be factored (and approximated) by energy exchanges (or *links*) between pairs of these structures, thus making the economy of a large number of form factor computations. The level at which links are established is a trade-off between accuracy and speed. The linear system is then solved by summing energy contributions along these links (See Figure 2).

3.2 Overview of the algorithm

Hierarchical radiosity with clustering [Smits et al. 1994; Sillion 1995] is usually well adapted to treating scenes of varying orders of magnitude, thanks to its automated adaptability and to the continuous trade-off it offers between computation time and accuracy. However, such methods are limited in scene size because of the super linear amount of memory they require in terms of the number of input polygons. The main difference between our method and traditional lighting simulation methods is the use of instantia-

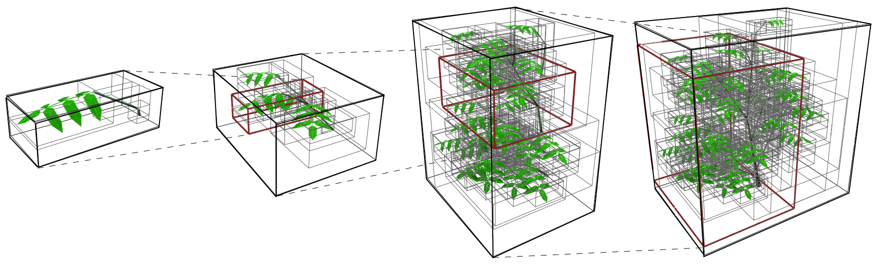


Fig. 1. Hierarchy of clusters for representing a small plant.

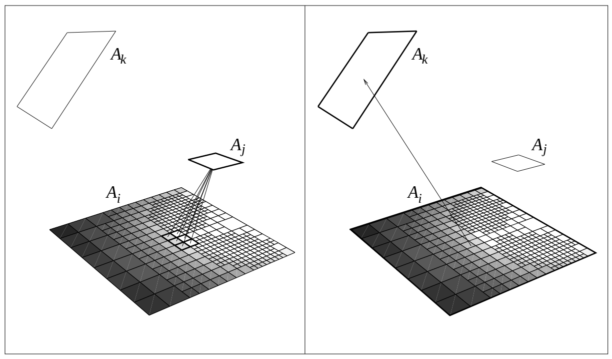


Fig. 2. Example of a speed/-accuracy trade-off. A single link is sufficient for representing the average energy transfer from A_i to A_k while several links are necessary to account for variations in the energy transfer from A_i to A_j. surface discretization and energy transfer links established at adequate level to minimize the number of computations while trying to account for local variation of the energy contribution between concerned elements.

tion [Soler and Sillion 2000]. Instantiation, more commonly used in ray tracing methods, allows to treat arbitrary large scenes by storing in memory only the necessary geometry for current calculations. Applying this paradigm to hierarchical radiosity with clustering, we thus combine the low memory cost of instantiation ray tracing methods and the stability and controllability of hierarchical radiosity with clustering.

One very eye-striking characteristic of plant models is self-similarity: leaves in a plant are very similar to each other and, up to a large extent, branches look like other branches as well as an entire plant looks like any entire plant of the same species and age. It is thus possible to approximately represent the geometry of a plant model using a small number of representative elements (branches, leaves, etc) that we can instance in order to build an efficient representation of the plant as shown on figure 3.

However, radiosity algorithms compute an explicit representation of illumination, typically associated with the geometry in the form of a mesh. Copies of a given object each have their own, unique illumination. Instantiation for radiosity is therefore more elaborate than for simpler rendering techniques (like ray tracing), since it should differentiate between the geometry (that is easily shared) and the illumination (that varies from one instance to another). Moreover, self-similarity in plants is never exact, and similar structures show more and more differences with age: unlike leaves, that really look like each other, larger structures only look similar in their overall shape.

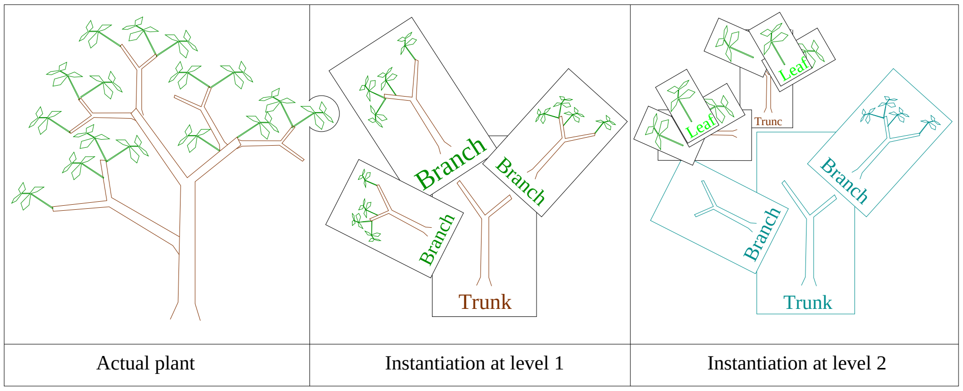


Fig. 3. Using approximate self-similarity between structures to instantiate a plant model. Note that although a common model is used to instantiate branches at level 1, each branch is in turn detailed using appropriate instances for substructures in level 2.

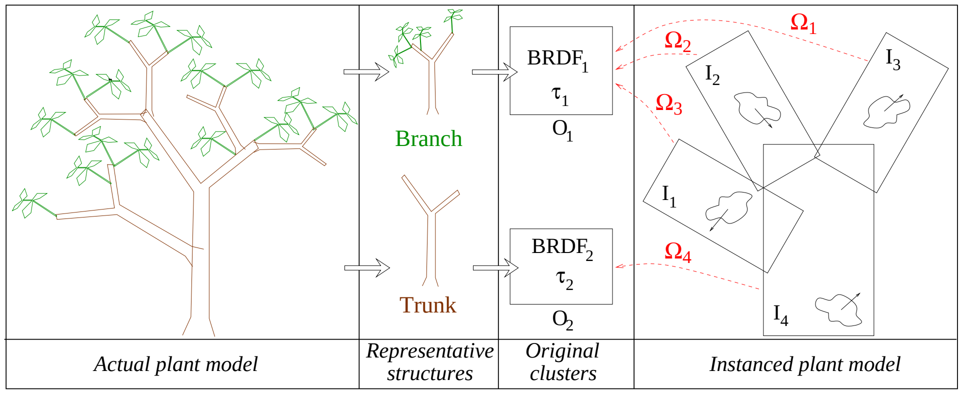


Fig. 4. Instantiation of a plant model for hierarchical radiosity computation. (1) Each structure of the plant (branch, leaf, trunk) is independently assigned a representative structure that sufficiently resembles the current structure. The phase function (or BRDF) and transmittance properties (see text and definitions in section 4.2) of representative structures are packed into *original clusters*. (2) the plant is represented as a collection of instances (I_1,...,I_4), each one pointing to the adequate original cluster (O_1 or O_2 in this example) and equipped with a geometric transformation (Ω_1,...,Ω_4 here) that permits visibility and reflectance computation on the instance using the information of the original cluster.

Because of this, since we still need to know the energy distribution on the real geometry of every part of a tree, the computation can not be performed on a unique geometry shared by the instances. Instead, the algorithm for computing the equilibrium of light using instantiation works as explained by the two following key points :

(1) The core idea of the algorithm is to share between sibling instances macroscopic functions (instead of geometry) in a single object called an *original cluster*, and use these functions for propagating light inside the scene. As will be shown in section 4.2, these functions are designed to simulate the energy resulting from the interaction of the real

```
main()
    OpenOutputFile()
    OpenInstance(root)
    CloseOutputFile()

HierarchicalInstantiation(cluster H)
    if IsAnInstance(H)
        OpenInstance(H)
    if IsACluster(H)
        ForAllChildren(g,G)
            HierarchicalInstantiation(g)

    if IsAPolygon(H)
        Render(H)

OpenInstance(instance H)
    cluster G = LoadNextLevel(H)
    ReplaceInHierarchy(H,G)
    TransferLinks(H,G)

    ComputeLocalSolution(G)

    HierarchicalInstantiation(G)

    DeleteLinks(G)
    ReplaceInHierarchy(G,H)
    Delete G
```

Fig. 5. Pseudo code for the radiosity instantiation algorithm. The generic **Render()** procedure replaces any output of the information as rendering the polygon to an off-screen buffer, or saving its radiosity to a file previously opened by **OpenOutputFile()**. The call **LoadNextLevel(H)** loads from the disc the actual geometry of the opened instance **H**, possibly creating instances at lower levels and returns the result as a cluster. This cluster is temporarily put in the hierarchy in the place of **H** and links pointing to and from **H** are also changed to act on it. Functions **IsAnInstance(), ForAllChildren(), IsAPolygon(), DeleteLinks()** are simply named according to what they exactly do.

geometry with light coming from any given direction, concerning out-scattering (or *phase function*, as defined in [Siegel and Howell 1992]) and mean attenuation along rays crossing the instance. A geometric transformation inside each instance properly links the replaced geometry to the original clusters (See Figure 4). We explain in section 3.3 that this makes it possible to compute the equilibrium of light energy at any hierarchical level above the instances while limiting the number of objects in the scene hierarchy.

(2) After a radiosity solution is obtained in a scene containing instances, the illumination of objects hidden *inside* each instance must still be determined. This involves a *local hierarchical radiosity solution* in which the contained geometry is temporarily loaded into memory and subjected to the incident illumination already computed for the considered instance. This is a *local* pass because only links that bring energy inside the instance are now considered and refined.

A major potential difficulty is that the contents of the instance might still be too complex to allow a memory-efficient hierarchical radiosity calculation. The *hierarchical* instantiation algorithm provides an elegant and efficient solution to this problem: because plants have self similarity at multiple hierarchical levels (between leaves, branches, whole plants), loading the geometry of an instance may include the temporary creation of new instances at lower levels to which the algorithm can be applied recursively.

We detail this operation named *opening an instance* in section 3.4.

The pseudo-code in Figure 5 summarizes the algorithm.

3.3 Local Hierarchical Radiosity solutions

Each time a portion of the scene is loaded (which concerns the whole scene when starting the algorithm, or a smaller part of it when "opening" an instance), the corresponding local hierarchy is loaded into memory with a depth limited to the next possible level of instantiation. As a result, the entire scene can be described as a hierarchy of clusters, in which instantiable clusters appear at various levels (possibly one included in the other). However, during any call to the computation of a *local* solution using hierarchical radiosity, the part of the hierarchy that is considered always consists of a cluster hierarchy whose leaves are either non-open instances or polygons.

The local hierarchy is processed by the hierarchical radiosity solver, which involves iteratively establishing (refining) links between clusters and propagating energy until convergence. Refinement of the links is limited to the level of instances, since their geometry is not available at this time. However the resulting solution is still much more accurate than if we had performed a hierarchical radiosity solution on the entire scene while limiting the link refinement to the level of the corresponding clusters. The fairly precise representation of each instance "phase function" or BRDF, which is precomputed, embodies the effect of light propagation and scattering inside the instance. In addition, it should be noted that, unlike normal clusters [Sillion 1995], no self-links are established on instances, because their phase function already accounts for internal light scattering.

During these temporary hierarchical radiosity solutions, elements in the hierarchy that previously exchanged light with the parent (now opened) instance are treated as fixed light sources. Indeed, thanks to the use of the precomputed instance BRDF functions, the internal solution among instance contents is not supposed to act on energy exchanges external to the instance. However, this is not perfectly true because of the approximate instantiation and translates into an approximation in the solution finally obtained (See section 6).

Once the local solution is obtained, we traverse the local hierarchy, and focus on each instance encountered recursively calling the local Hierarchical radiosity algorithm on it. When we reach a level with no instances below, the local solution is equivalent to hierarchical radiosity with clustering, and a complete solution is available for the current branch of the scene hierarchy, taking into account contributions from the entire scene.

Finally, the local geometry is destroyed and replaced back by its parent instance. Consequently, the solution for the current portion of the hierarchy is accessible at the current stage only, because its supporting geometry will be deleted when closing the parent instance. We thus render the corresponding polygons into an off-screen buffer (or output the results to a file), thereby progressively forming the image during the traversal of the scene.

3.4 Opening instances

We detail here the operations involved in the opening of instances during the recursive traversal of the instance hierarchy. This process is illustrated in Figure 6. On the left, we see a solution computed at a given level. Oval shapes represent objects or clusters, while rectangles represent instances. Links are indicated by arrows, and have been created at varying levels of the cluster hierarchy.

When the lower-right instance is opened, we build a hierarchy with its contents, as shown on the right-hand side of the figure. In order to properly account for all incoming light, we create copies of all links that previously arrived on the instance (marked using dashed lines on the figure) and attach them to the root of the new hierarchy. We also add a self-link to the

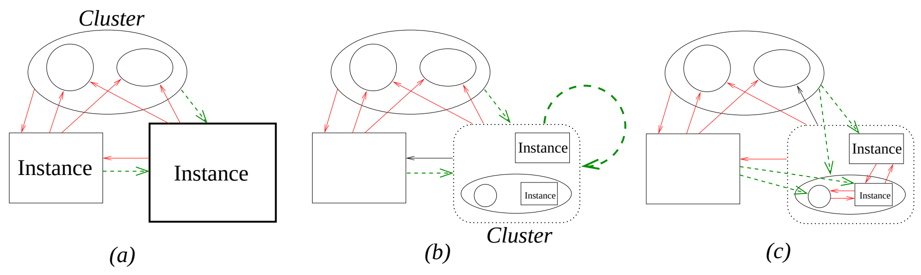


Fig. 6. Closer view on the computation of the solution. *(a)* a solution as been computed using the reflection and transmission properties of instances; links are indicated by arrows. *(b)* the bottom right instance is "opened", *e.g* its geometry is loaded into memory. *(c)* incoming links (in green) are locally refined as well as the self link that was added on the replacing cluster.

root if no self-link exist on any parent levels, to account for all internal exchanges [Sillion 1995]. This newly created cluster needs this self-link indeed, as any regular cluster of the hierarchy.

We can then apply the solution procedure outlined above, that is first solve for radiosity, then traverse the hierarchy to open instances and recurse. The right side of Figure 6 illustrates the radiosity solution, in the opened level: Dashed links correspond to links that previously arrived at the instance level, and have been refined. Internal links issued from the refinement of the added self link are also represented. The recursion would then continue into the smaller instances before returning to the left-hand situation and opening the other instance.

Note that refinement is constrained such that, only elements belonging to the considered hierarchy may be subdivided (either as emitters or receivers). Gathering and push/pull operations are also applied to the local hierarchy only, essentially treating all elements external to this hierarchy as fixed light sources.

3.5 Cost considerations

Simple recursion arguments allow us to evaluate the cost of our algorithm in terms of storage and computation cost. Let us denote the number of instantiation levels by k, the number of elements (polygons plus instances) at each instantiation level by N, and the number of these elements that are instances by p. This model is very simple because it assumes a uniform branching factor among all levels of the hierarchy of instances, and a uniform proportion of instances and polygons at each level of the scene hierarchy.

By definition, the number of polygons in the root of the hierarchy (as well as inside each instance) is $N-p$. This is repeated p times at the next level, and $p^2 ... p^{k-1}$ times at subsequent levels until level $k-1$. At level k, there are no instances below and thus N polygons per instances. The total number of polygons in the scene is consequently :

$$n = \underbrace{(N-p)(1+p+...+p^{k-1})}_{\text{Levels 0 (root) to } k-1} + \underbrace{p^k N}_{\text{level } k} = O(p^k N) \quad (1)$$

Gain in memory. Let ε, I and o pectively denote the size of a polygon, an instance and an original (instanced) object in memory.

Assuming r original objects are used to create the p instances at each level, the memory footprint of the scene at the top level of the calculation is :

$$M_{inst}(1) = (N-p)\varepsilon + pI + ro$$

Since the algorithm only loads the geometry of the branch of the hierarchy it is descending into, the maximum memory requirement is reached at the bottom of the hierarchy, where it is :

$$M_{inst}(k) = k\left[(N-p)\varepsilon + pI + ro\right] \tag{2}$$

For the ideal case of a well balanced hierarchy of instances, the memory cost is thus logarithmic in terms of the total number of polygons in the scene. In any case, it is much less than the $O(n\varepsilon)$ memory size of the model itself. In scenes with limited instantiation depth (*e.g* k is small) the logarithmic equivalent does not hold anymore. In the worst case, the gain in memory is the number of instances times the ratio between the memory cost of an instance and the actual geometry.

As an example taken from our implementation and real data, consider $o = 1,100$ bytes (an original object holds two sampled directional functions at 528 bytes each), $\varepsilon = 150$ bytes (this rather large size accounts for geometry, radiometric and subdivision information) and $I = 200$ bytes (in our implementation, instances are also clusters and thus contain inherited information). For the tree presented in Figure 14, we have $n = 119,000$, $k = 4$, $r = 5$, $N \approx 30$ and $p \approx 8$. The expected memory size given by (2) is 48 Kb, which is much less than $n\,\varepsilon = 15,085$Kb, the expected size of the entire scene.

Although these numbers do not translate directly into required memory sizes, because of the missing constants and various fixed costs, we will see in the result section that a large memory reduction is observed, the gain increasing with scene complexity. It actually becomes feasible to simulate very large scenes that simply could not be treated by previous methods.

Since the accuracy threshold does not change when recursively computing the local solutions, the maximum number of links in memory can be estimated by the number of links that contribute to the illumination of a leaf element in a classical hierarchical radiosity solution on the entire scene, multiplied by the number of leaf elements at the lowest level, *e.g* $O(N\log n)$. This is much less than the $O(n\log n)$ links of the normal clustering radiosity method.

Computation cost. We consider that a hierarchical radiosity solution in a scene of n elements equipped with a well balanced hierarchy can be performed in $O(n\log n)$ time.

Let $C(i)$ denote the cost of our algorithm for solving level i and its sub-levels. To get an expression for this cost, we add the cost of a local radiosity solution between N elements to the cost of recursively calling the algorithm on the next level for the p instances :

$$C(i) = N\log(N) + pC(i+1) \qquad \text{and} \qquad C(k) = N\log(N)$$

The cost for the entire scene is thus :

$$\begin{aligned} C(0) &= N\log(N)\left(1 + p + ... + p^k\right) \\ &= O(p^k N\log(N)) \end{aligned}$$

Considering that $n = O(p^k N)$, the value $C(0)$ appears to be equivalent to $O(n\log(N))$, which is close to the cost of the classical hierarchical radiosity algorithm. Practical experiments show that, whereas the gain in memory is a little over estimated due to some fixed costs, the $\log n / \log N$ gain in computation time accurately reflects reality.

3.6 Discussion

The Hierarchical Instantiation algorithm essentially gains by neglecting the correlation between objects lying in different instances at the same hierarchical level. For two such sibling instances, no link can ever be created between one object from each, because the contents of both instances are never simultaneously present in memory. This ensures that every local solution only involves a small number of objects, at the expense of a small approximation. We will illustrate this approximation on practical examples in section 6.

For the same reason, a complete solution is never present in memory, although every part of the global solution is available at some stage of the calculation. This explains why any results such as images or radiosity values written to a file must be output during the calculation as mentioned earlier.

A similar behavior could be achieved in a normal radiosity algorithm, by preventing the refinement of an emitter if it is an “instantiable” object different from the receiver. However the global accuracy would be lower unless the emitter is already refined enough to obtain a high-quality representation of its internal light distribution. Since the phase functions of the original instances are pre-computed and stored, more computation time can be invested in this process than typically done in a hierarchical radiosity computation. For instance, the effects of internal visibility in emitting clusters, which are usually not computed for cost reasons [Sillion and Drettakis 1995], are intrinsically accounted for in the phase functions.

4. LIGHT PROPERTIES OF PLANTS

Instantiation in plant models is based on plant self-similarity. In section 4.1 we first discuss how to determine potential instances and sibling structures in plant models. For an instance to be able to participate in radiosity calculations without accessing its geometric content we require the knowledge of (a) an outgoing radiance distribution [Sillion et al. 1995], (b) a bidirectional scattering phase function to convert incoming energy into outgoing radiance, and (c) a transmittance function. We discuss in 4.2 how to represent and compute these functions. Finally, at the lowest level in the geometric hierarchy, the leaves of the plant are responsible for light interaction with the model. In 4.3 we present the model we use for local leaf-light interaction.

4.1 Identifying instantiable structures

We want to reveal the redundancy present at different scales in plant models. The question is thus : how to characterize similar structures in a plant ? For this we can distinguish two approaches :

Formally, two structures can be replaced by a common instance as soon as they have sufficiently similar phase and transmittance functions. However, a complete investigation of these functions over all structures in a plant is a very expensive calculation, which makes any brute-force approach impracticable.

The second possibility is to rely on additional information related to the plant models. In our case, plants are defined as hierarchies of botanical structures, each one being assorted with an orientation and a collection of botanical parameters, such as the number of leaves contained in the structure, its physiological age [(de) Reffye et al. 1996], the type of the structure (*Branch, leaf, whole plant, flower,* etc.). It makes sense that structures of the same number of leaves and same type have very similar geometry and thus very similar phase and transmittance functions. We will verify it through an example :

Figure 7 shows structures of a poplar tree and a locust tree. The curves in Figure 8 represent the cut of their phase function for fixed input θ and φ and output θ, and varying output φ[1] We indicate for each structure its type (Branch, Plant, ...) and its number of leaves. On the left side (poplar) we see that structures of similar number of leaves (by *similar* the mean the same order of magnitude) have very similar phase function values. This phenomenon is all the more verified that the number of leaves is large. The number of leaves in a structure can thus in this case be used as an efficient way of detecting instantiable structures throughout the model.

Looking at the same curves for the locust tree, it appears that the similarity is not respected between structures of number of leaves of the same order (see for instance $Branch-248$ and $Plant-367$) unless they are of the same type (for instance $Branch(c)-248$ and $Branch(c)-131$ are very similar, as well as $Plant-367$ and $Plant-1615$). A pertinent parameter set is thus the number of leaves plus the type of the structure for this particular case. We have found that these parameters work fine for all other species of trees we have tested.

As we will see later, the memory cost of the phase function of an instance can be quite large, and when using our lighting simulation algorithm in cooperation with a plant growth simulation program, larger and larger structures may appear in the scene. Computing the phase function of these structures on the fly would be very costly (even more than not using instantiation at all !). Fortunately, we observe that, as structures get more complicated, they tend to have their phase function and transmittance converge to a fixed value. This can be observed in Figure 8 for the phase function. Our policy is therefore to use a fixed phase function and transmittance for structures larger than a certain size.

In our implementation, the information needed to know which parameters are relevant for instantiation is stored into an *instantiation policy* file, as well as the differentiation intervals for these parameters and the maximum size of differentiable structures for each type. A specific *instantiation policy* file has been constructed for each kind of plant. This also means that we are performing *approximate instantiation*, *e.g* we only use a small number of representative structures to provide phase function and transmittance for all possible instances. The policy for sharing the phase functions and transmittances is for the moment designed by hand for each plant, but it could be automated based on the computation of differences between phase functions of various structures.

Finally, turning each structure definition into a cluster, we obtain a cluster hierarchy that only contains *instantiable* clusters but still may have a very large branching factor. Its efficiency toward hierarchical radiosity is then improved by inserting new levels of (non instantiable) clusters, using a constrained clusterizer [Hasenfratz et al. 1999]. Besides, self-similarity occurs in vegetation scenes at multiple scales including groups of plants of various sizes, and there is no reason to limit the instantiable hierarchy to the level of the plant themselves, as soon as we manage to compute (or predict) their phase functions.

[1]The angles $\theta \in [0..\pi]$ and $\varphi \in [0..2\pi]$ are the angular coordinates of a direction, in the coordinate system local to the plant structure: the main trunk of the structure is aligned with the z axis ($\theta = 0$) and the x axis ($\theta = \frac{\pi}{2}, \varphi = 0$) is orthogonal to the z axis of the parent structure.

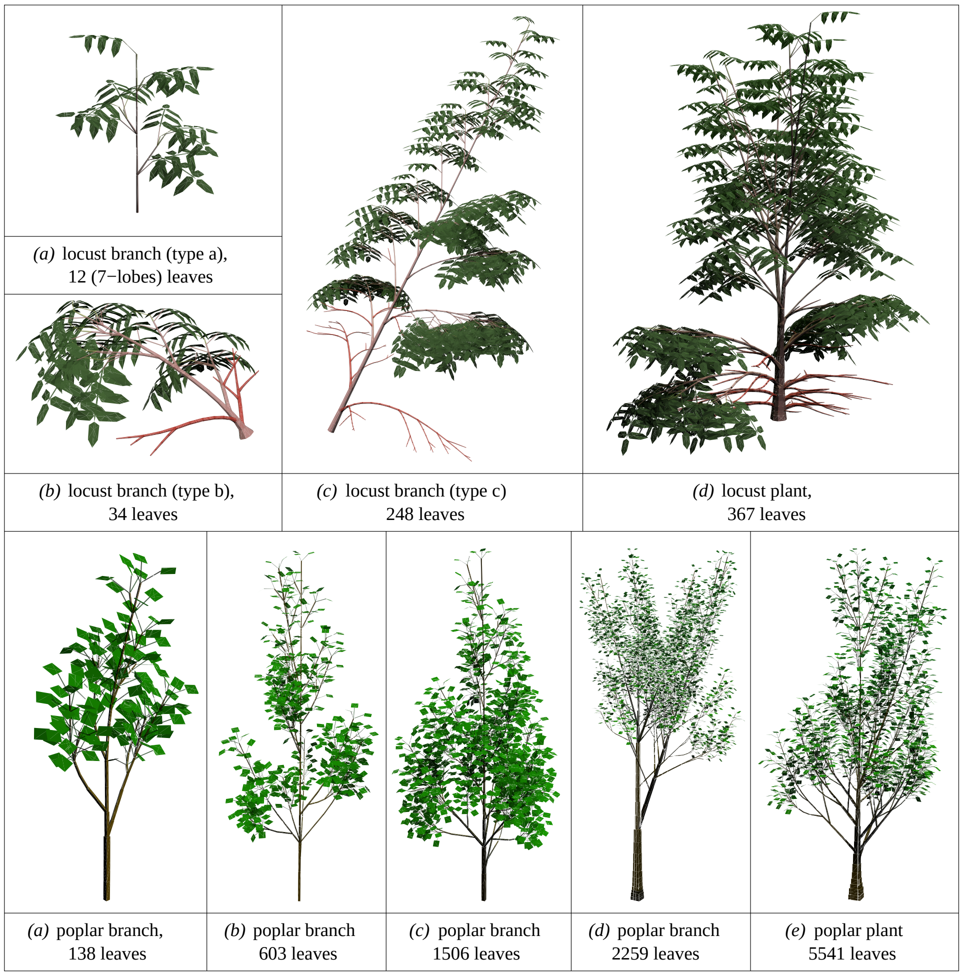

(a) locust branch (type a), 12 (7−lobes) leaves
(b) locust branch (type b), 34 leaves
(c) locust branch (type c) 248 leaves
(d) locust plant, 367 leaves
(a) poplar branch, 138 leaves
(b) poplar branch 603 leaves
(c) poplar branch 1506 leaves
(d) poplar branch 2259 leaves
(e) poplar plant 5541 leaves

Fig. 7. Some of the structures used to compute the illumination in the poplar and locust trees in this paper. In the case of the poplar tree, the relative random orientation of the leaves and the common shape of branches and whole plants makes the number of leaves a sufficiently pertinent factor for instantiation. This is not the case for locust structures, as can be seen on the phase function values of Figure 8. Locust structures are represented in the global coordinate system, poplar ones are in their local coordinate system given by their botanical orientation.

4.2 Reflection and transmission properties of representative structures

4.2.1 *Definitions.* To represent reflectance properties, we can choose between at least three kinds of functions of various memory costs and accuracy: (1) constant values, *e.g* a single spectra, like in diffuse radiosity algorithms, (2) mono-directional functions corresponding to the average radiosity of the replaced geometry when stimulated from a given input direction [Soler and Sillion 2000], and (3) bidirectional phase functions. Representations 1 and 2 are easily obtained from the phase function itself.

The bidirectional phase function f is defined in general as the contribution of input light energy $I_i(\theta_i, \varphi_i)$ in the incoming direction (θ_i, φ_0) to the energy $I_s(\theta_s, \varphi_s)$ that is scattered

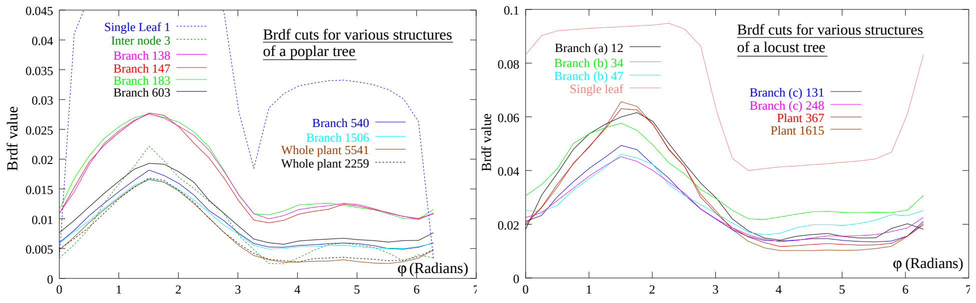


Fig. 8. These different curves represent slices of the phase function of various structures extracted from a model of poplar tree (*left*) and a locust tree (*right*), computed using a hierarchical radiosity algorithm (without instantiation) on the plant models. See Figure 7 for images of some of the corresponding structures. The slices correspond to a lateral input direction of $\theta_{in} = \frac{\pi}{2}$ and $\varphi_{in} = \frac{\pi}{2}$ and a lateral output direction of $\theta_{out} = \frac{\pi}{2}$ and varying φ_{out}.

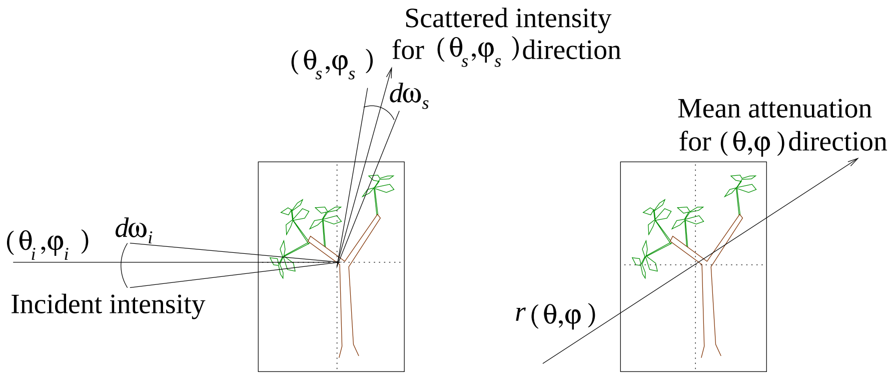

Scattered intensity for (θ_s, φ_s) direction
(θ_s, φ_s)
$d\omega_s$
(θ_i, φ_i)
$d\omega_i$
Incident intensity
Mean attenuation for (θ, φ) direction
$r(\theta, \varphi)$

Fig. 9. Notations for the definition of the phase and transmittance functions of an instance.

toward direction (θ_s, φ_s) per unit solid angle in both directions [Siegel and Howell 1992] (see also Figure 9 for notations) :

$$I_s(\theta_s, \varphi_s) d\omega_s = f(\theta_i, \varphi_i, \theta_s, \varphi_s) I_i(\theta_i, \varphi_i) d\omega_i$$

By extension, and with respect to its usual definition in computer graphics, we also define f to be the *BRDF* of the instance.

The transmittance function $\tau(\theta, \varphi)$ of an instance is defined as the mean value of the binary visibility $v(\theta, \varphi)$ along all rays of a given direction (θ, φ) that cross the bounding box of the instance :

$$\tau(\theta, \varphi) = \frac{1}{A_{\theta,\varphi}} \int_{r//(\theta,\varphi)} v(\theta, \varphi) dr$$

In this expression, $A_{\theta,\varphi}$ denotes the area of the projection of this bounding box in the given direction. Obviously 4-dimensional per ray transmittance functions would be overkill because they would faithfully represent a geometry that is not the one the instance really replaces, due to the approximate instantiation. Using a single transmittance value [Sillion 1995] (for instance based on the density of clusters) would amount to considering plant structures to be isotropic. This assumption is not valid for a number of

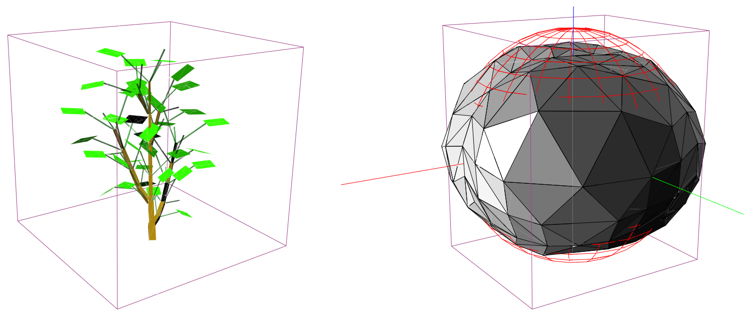


Fig. 10. A small structure from a poplar tree and its directional transmittance function. The red sphere represents the mean value of the function over all directions. The flatness of the leaves make the transmittance higher in near-horizontal directions and lower in near-vertical directions.

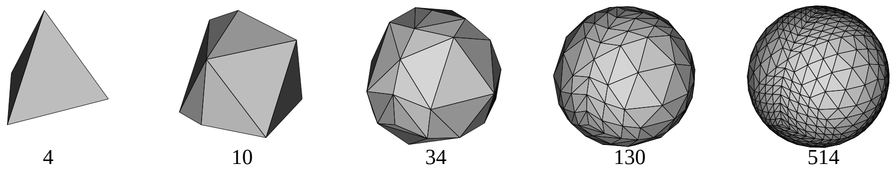


Fig. 11. Recursive subdivision of a tetrahedron to obtain a sampling of directions.

structures such as plagiotropic branches, in which leaves share a common orientation. We consequently have chosen a directional approximation of the transmittance. An example of such a function can be seen in Figure 10.

4.2.2 *Representation.* The key point in choosing a good representation of directional functions is the compromise between fast access to the values and low memory cost of the storage. Some authors for instance used spherical harmonics [Gastellu-Etchegorry et al. 1996; Sillion et al. 1995]. These are inherently smooth and therefore not very suitable for representing highly varying functions.

We represent directional functions as an array of values they take for a set of pre-sampled directions distributed on the unit sphere. These directions are obtained by recursive subdivision of a tetrahedron, yielding to successive sets of 4, 10,34,130,514... directions associated to the vertices of the resulting polyhedron, as shown on Figure 11.

Fast access to the values of the functions is provided using linear interpolation on each face of the polyhedra. Any useful information that does not depend on the function values but serves some computation (solid angle attributed to each direction, adjacency relationship between the directions, etc) is shared. Bi-directional functions are also sampled in the same way, each input direction producing a directional distribution of outgoing light.

The table below sums up the memory cost inherent to various choices for representing reflectance function of the poplar branch of Figure 7.b (Bottom row) using constant values, mono-directional functions sampled in 130 directions, and bidirectional functions sampled in respectively 34^2,130^2 and 514^2 directions. The first two columns indicate the maximum and average difference between the reflectance function and the most accurate 514^2 bidirectional phase functions. The last column shows the measured memory cost for propagating light in a poplar tree of 5541 leaves (Figure 7.e) using 4 levels of instances, and an average of 8 instances and 30 polygons per newly opened instance. Four original clusters (and thus four distinct reflectance functions) were used.

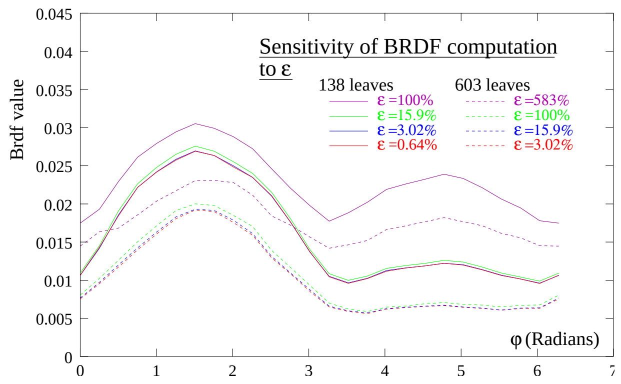


Fig. 12. This figure shows cut sections of the BRDF of two structures (see Figure 7.a and 7.b, bottom row) computed with a hierarchical radiosity algorithm with different accuracy threshold values. For both structures there exists a value of ε under which the functions are nearly indistinguishable from the actual BRDF of the geometric model.

Reflectance function type	Error		Function size in kB	Memory in kB
	$\|\cdot\|_\infty$	$\|\cdot\|_2$		
Constant	70 %	15 %	0.012	1
Mono-directional	52 %	11 %	6.168	26
Bi-directional 34^2	24 %	13 %	13.872	993
Bi-directional 130^2	11 %	3 %	202.800	1 748
Bi-directional 514^2	0 %	0 %	3 170.352	13 618

When using bi-directional reflectance functions, mono-directional radiosity and irradiance distributions are necessary for every instance. Bidirectional reflectance functions can be quite costly, but these functions are shared by many instances and only a few distinct bidirectional phase functions are usually necessary (hardly more than 10) for a given plant.

In the above experiment, the memory cost is essentially driven by the size of the phase functions we use. Indeed, thanks to the small branching factor (38 in this case) of the hierarchy, the number of instances simultaneously present in memory is small (Less than 8 here) and their cost is over-weighted by that of the phase functions for 514^2 directions.

Looking at the associated accuracy, our conclusion is that mono-directional reflectance functions are usually a good compromise but still can advantageously be replaced by phase functions of 130^2 directions when very accurate solutions are needed.

4.2.3 *Computation.* To compute the phase function of a structure, we stimulate the geometry with a light flux of constant intensity and controllable direction, compute the equilibrium of light energy inside the geometry using a hierarchical radiosity algorithm and measure the outgoing light intensity and spectrum. Although sampling is performed using graphics hardware, the computation itself is a costly operation that mainly depends on the accuracy threshold of the hierarchical radiosity used [Sillion 1995].

Figure 12 shows an example result of such a calculation. It appears clearly that the accuracy threshold must be sufficiently small to reach convergence, but not too small to avoid wasting time without any improvement of the phase function. In practice, threshold values of about 20% (relative to the impulse stimulation) provide excellent results. The hierarchical radiosity algorithm indeed proceeds an in-depth refinement of light interactions between clusters in the structure. An interaction between two clusters is considered satisfactory when the estimated variance of energy transfers between pairs of polygons of these clusters is smaller than ε.

To compute with the phase function the outgoing distribution of light of an instance receiving incoming energy along a link, we multiply its phase function for the incident direction by the irradiance along the link. For that reason, the phase function can also be called the *reflectance* function of the instance, as an extension to its usual definition on simple surfaces.

Transmittance functions are computed for each direction, by rendering the corresponding plant structure into an image. Three colors are used during this operation : one for the geometry, one for the background and one for the bounding box of the geometry. The directional transmittance of the object is then obtained as the proportion of pixels of the bounding box still visible among all pixels of a different color than the background. Computing the transmittance is much faster than the phase function since it only requires a few successive off-screen renderings.

4.3 Leaf reflectance and transmittance model

Examining real plants, it clearly appears that the reflectance of leaves is not necessarily the same on both sides, and is also partially specular mainly on the side facing the sun. According to [Baranoski and Rokne 1997] and [Govaerts 1995], the translucency of plant leaves is also almost purely diffuse. Note that transmission through a leaf can amount to up to 40 % of the received energy, depending on the wavelength considered.

For the sake of simplicity, we currently assume diffuse reflection and transmission on leaves. But, although it would require to store directional distributions of out-coming energy and multi-dimensional reflectance functions, using a directionally-dependent reflectance (or transmittance) model for light transfers on leaves does not cost much more for two reasons : (1) only a small number of leaves are simultaneously present in memory thanks to the instantiation algorithm and thus only a few out-coming light distributions are simultaneously required. (2) the reflectance/transmittance functions being the same for all leaves in the plant, they can be shared in memory.

In summary, our model for light exchange at the level of leaves simply consists in transferring the irradiance on each side of a leaf to the radiosity on the other side, after multiplication by a diffuse translucency factor τ for each side. Calling B_i the radiosity of side i of a leaf and I_i its irradiance and R_i its reflectance, we use:

$$B_1 = \rho_1 I_1 + \tau_{2\to 1} I_2$$
$$B_2 = \rho_2 I_2 + \tau_{1\to 2} I_1$$

This is computed during the hierarchical push/pull operation [Sillion 1995]. The light energy pulled up in the hierarchy is the average value of the energies on both sides of the leaf.

Transmittance through standard geometry is computed using ray-tracing and the total attenuation along a ray that crosses several instances is computed by multiplying the transmittances of the instances together.

5. IMPLEMENTATION ISSUES

We have grouped into this section the details concerning our implementation of the hierarchical instantiation algorithm for vegetation scenes.

Instantiation database

Reflectance and transmittance functions for the representative structures are stored into a database. The database contains a module that is responsible for choosing which representative structure can approximate a given part of the plant. This module is also in charge of updating the database on the fly, in the case where it cannot find a suitable representative structure among already computed ones. A buffer of plant structures is also used in order to keep trace of already loaded plant geometry and phase functions.

Light sources

Our system currently handles directional light sources, *e.g* that illuminate all objects from the same direction (thus producing hard shadows), infinite light sources (*e.g* sources that are composed of an independent set of incoming directions packed into a fixed solid angle scope, thus producing soft shadows), and local diffuse light sources as in standard radiosity packages.

In order to simulate a sky dome illumination during a long period (which is the more realistic situation for a tree), we combine an infinite light source covering the sky hemisphere and a number of infinite light sources with small solid angles to represent sampled positions of the sun during the period.

Radiosity-like diffuse light sources are not common in nature, but still can be used to simulate the illumination in urban environments and to simulate the illumination coming from the painted windows of a greenhouse.

Representation of light

Light is represented by components along different wavelengths. Red/Green/Blue is used to directly obtain images, but infrared simulations are more useful for applications in agronomy, such as computing the temperature and also studying geometrical response of the plant (positioning towards light) to the illumination [Gautier et al. 2000].

Refinement and visibility

The refinement is a very important stage of each iteration of any hierarchical radiosity-based algorithm, since it decides at which level in the hierarchy energy exchanges will take place. From the choice of the minimum size of refined elements depends the accuracy of the shadows projected by objects in the scene. During refinement, a conservative visibility information is ensured using shaft-culling and inherited by refined links. When the scene contains instances, the geometry that the instances replace is not present. Thus it is not necessary to refine elements that receive shadows from instances more than the size of the shadow itself.

This highlights one of the potential drawbacks of the hierarchical instantiation algorithm in its present form : in order to compute nice shadows, more geometry must be loaded into memory at the expense of the memory cost. More precisely, we can act on the minimum depth at which we allow geometry to be replaced by instances so as to get more and more accurate shadows.

Another method is to use *geometric instantiation*, *e.g* replacing the transmittance function of an instance by a copy of its geometry, both being shared by all instances of a given structure. This geometry is then used to trace rays trough the instance at a larger memory cost, but still much lower than the cost of an explicit entire scene. In addition to plants,

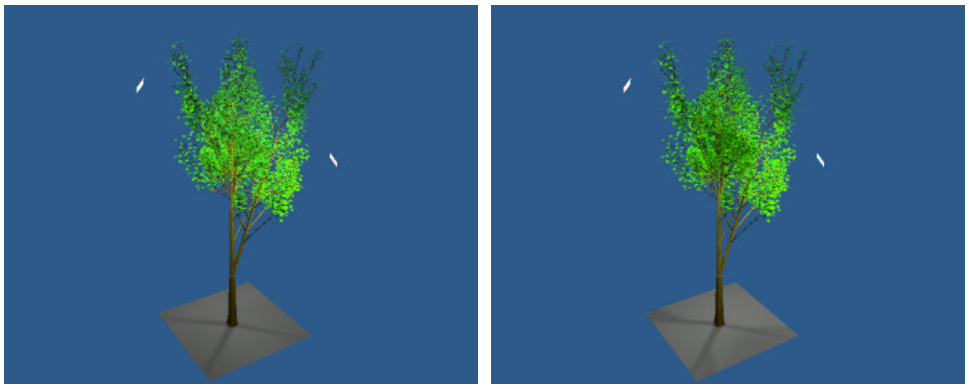
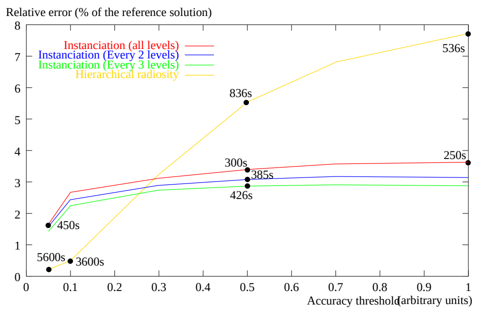


Fig. 13. Comparison between instantiation (*left*, $14min/8MB$) and classical hierarchical radiosity (*right* $119min/123MB$). Instantiation starts at depth 1 (*e.g* branches. Level 0 is the entire plant.)



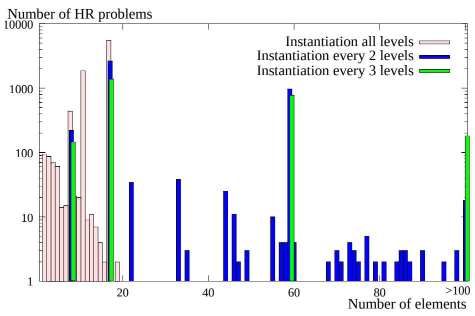


Fig. 14. *Left:* L^2 error in percentage of the maximum value of the reference solution (computed using hierarchical radiosity) for various values of the accuracy threshold used in link refinement. The numbers in black on the curves indicate computation times in seconds. *Right:* histogram of the number of hierarchical radiosity solutions on local hierarchies with variable numbers of leaf elements. When instancing at fewer levels, hierarchical radiosity computations tend to involve more elements.

we have successfully tested this method for repetitive objects in architectural scenes [Soler and Sillion 2000].

6. PERFORMANCE

In Figures 13 and 14 we compare the results of our instantiation algorithm to that of a classical hierarchical radiosity simulation with clustering. The same set of parameters has been used except the instantiation flag. The tree is a 30-year old poplar tree consisting of 119 000 polygons. Experiments have been conducted on a *SGI Origin2000* computer.

Looking at the images on Figure 13 the results seem identical at first sight. Some subtle differences can be found however, which mainly concern the variation of energy in some parts of the plant where instantiation has been used. We attribute it to the fact that in our implementation, the radiosity stored in the instances is not directional but represented as a single value (As explained in section 4.2). On the *top right* image for instance, the light arriving from the light source on the center of the tree has been distributed behind the instance because of this approximation.

As expected by our estimations, the gain is very important (15 times lesser memory in this particular case), especially considering that there are fixed memory costs (Our radiosity program requires a fixed amount of 5MB of RAM).

An important gain in computation time is also apparent (Hierarchical Instantiation is 8 times faster). Indeed, because the instantiation algorithm does not allow bidirectional refinement of energy links between instances, it refines fewer links and computes fewer form factors. The ratio between the two numbers of links is the average number of elements an emitter is subdivided into. This of course depends on the relative positions of the elements and on the refinement algorithm. If the instances are not too close to each other, we can consider that our algorithm is equivalent to normal refinement; for really close instances it is more approximate. Visibility calculations are also faster with instances, since they do not require geometric operations, instead using the stored mean transmittance value in the relevant directions.

On the left side of Figure 14, we show a comparison of the accuracy of hierarchical radiosity and hierarchical instantiation for values of the accuracy threshold used in the refinement of links. It appears that for larger values of the error threshold, hierarchical instantiation is much faster than hierarchical radiosity with a better accuracy. This confirms that using phase functions that accurately account for the internal scattering of light is more efficient than the traditional approximation used for clusters in hierarchical radiosity. For small error thresholds, hierarchical radiosity is still more accurate than hierarchical instantiation. We believe that this is caused by our choice of an omni-directional approximation of the reflectance functions in our current implementation, which is confirmed by the fact that instancing fewer levels in the hierarchy only marginally increases the accuracy. The repartition of the hierarchical radiosity solutions required by our algorithm in terms of local scene size is shown on the *right* side of Figure 14. It clearly appears that when instancing all levels, only hierarchical radiosity problems with small number of elements occur. When instancing every other level, the average number of elements per hierarchical radiosity problem increases. Finally, when instancing every three levels, the algorithm tends to mainly solve hierarchical radiosity problems with more than 100 leaf elements. The corresponding histogram for a classical radiosity solution would in turn be composed of only one bar of height 1 at n elements, where n is the total number of polygons in the scene.

Influence of instantiation on plant growth simulation

We just saw that the use of instantiation in our simulation of light distribution inside plants introduces some approximations. When the resulting light distribution is used to control growth, it is therefore legitimate to consider the possible impact of these approximations on the plant growth or architecture.

We therefore compare a growth simulation using instantiation[2] to what is obtained using standard hierarchical radiosity with clustering.

Since the plant growth simulator integrates the light received by a particular region of the plant over a certain period of time and of space, local variations of the solution in time and space may not affect the result of the simulation, i.e the shape of the plant. We will thus directly measure the impact of these approximations on the plant model itself.

Measuring the "difference" between two plants grown under slightly different conditions

[2]The plant growth simulator we use is briefly described in Section 7.2

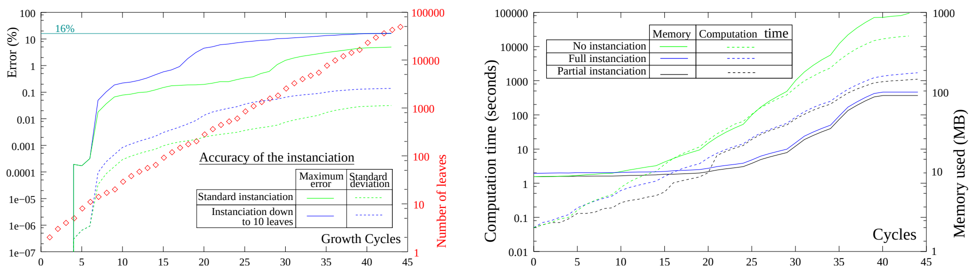


Fig. 15. *left :* Maximum value and standard deviation of the structural distance between a plant grown using instantiation and the same plant grown using standard HR with clustering, monitored during growth as a function of growth cycle number. In blue: instantiation was limited to structures of more than 10 leaves, after which a simple gather and pushpull was performed on the structures, thus inducing more error. In green: standard instantiation. *Right:* The computation times and memory cost corresponding to each growth cycle are displayed for these two experiments, as well as for the growth without instantiation. Note that when using instantiation, lighting simulation typically takes a fixed proportion of total growth simulation computation time (roughly 80%).

is not straightforward because their topology may not be the same. To cope with this, we have turned off the fall of dead structures and chosen a plant for which the branching factor is constant. In that case computing the difference between equivalent structures in two plants of the same age becomes easy.

Figure 15 shows the error associated with the use of instantiation when computing the light distribution during a plant growth simulation, as well as computation times and memory usage. This example can also serve as a reference for computation times and memory costs of our method.

The conclusion for this small experiment is that instantiation only perturbs the growth simulation in a marginal, very acceptable manner. Limiting the depth of the computation to instances of at least 10 elements only brings a small gain in computation time and saves the memory cost of light properties of small structures (*e.g* two BRDFs in that case).

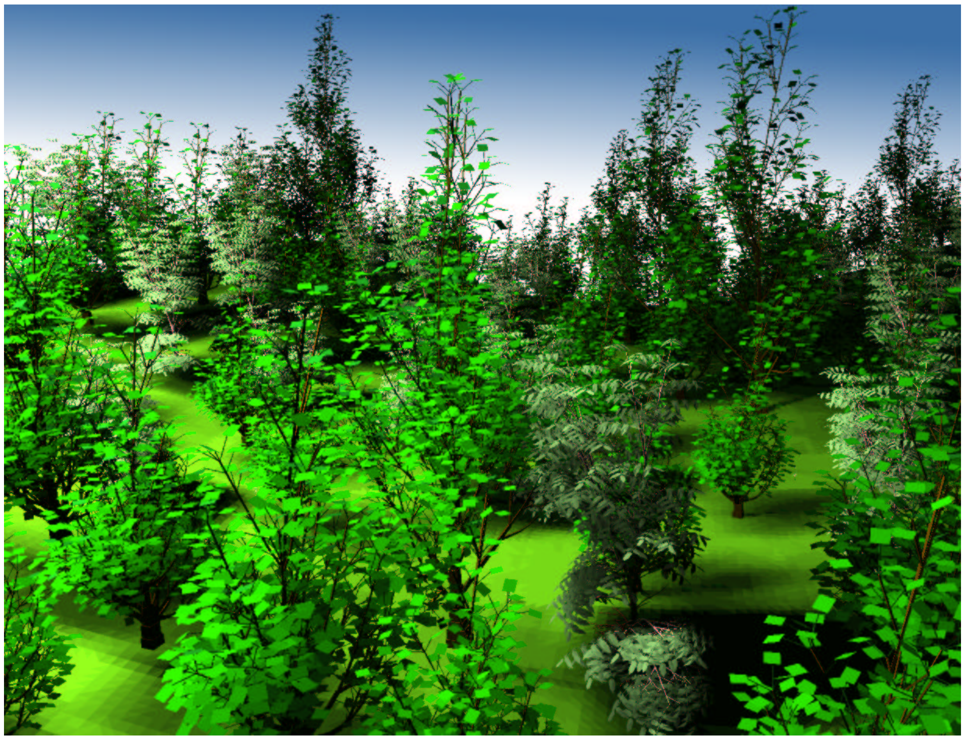
7. APPLICATIONS

As stressed in the introduction, lighting simulation in plant models has important applications in computer graphics (synthetic image generation) as well as in agronomy (plant growth simulation). Intermediate applications may also be considered, such as urban and architectural design. We discuss in this section the key points in using instantiation for the first two applications: rendering and plant growth simulation, and give practical examples for each.

7.1 Rendering

As previously discussed, the global illumination values on the geometry elements are only accessible during the calculation, when the corresponding parent hierarchy of instances are open. In order to obtain a view-independent solution, we have to save it in a file during the computation. To produce images, the user can load an instanced version of the scene (thus displaying bounding boxes instead of the complex geometry of instances) and load and display the result after having chosen the view point. We used this technique for the images in Figure 16.

When computing a single static image however, we simply render the geometry to an off-screen buffer. In this case, we are obviously not interested into the geometry that does



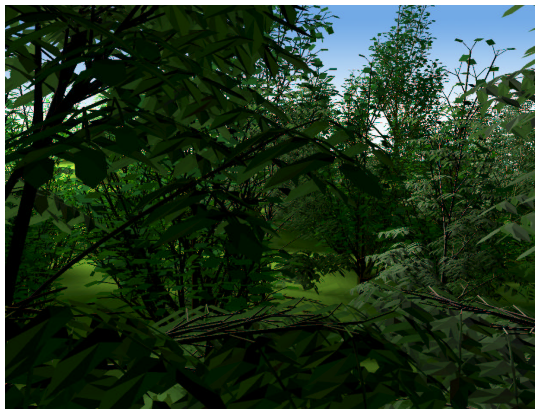


Fig. 16. A plantation of 233 trees (Poplar and locust trees) for a total of 2M leaves and 5M initial polygons. Computation time is four hours. Note that most the time is dedicated to the refinement of local energy transfers in order to obtain acceptable shadows. In other words, coarser solutions that approximate the global illumination solution very well are still obtained in a much shorter time (20 mn). The total memory cost for this scene is 80 MB.

not appear on the image itself. Because this geometry still participates into the global equilibrium of light energy in the scene, traditional radiosity algorithms have to consider it explicitly. Thanks to the instantiation algorithm, it is not necessary to open instances that do not appear in the viewing frustum during the calculation, which drastically increases efficiency while hardly altering the solution. Only instances that appear on the image are thus opened.

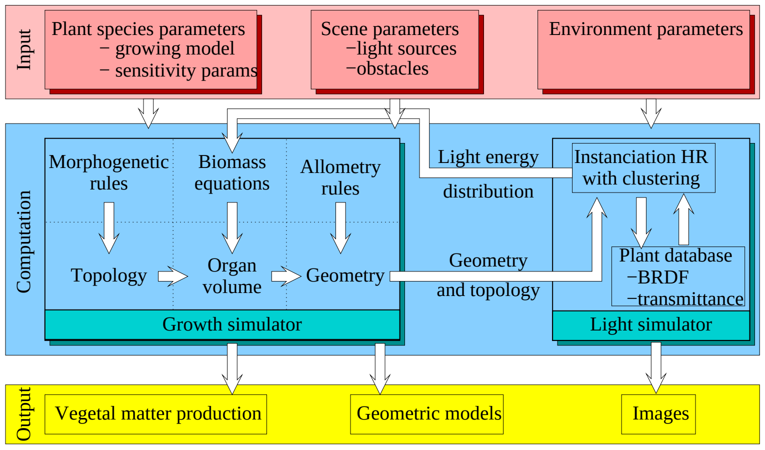


Fig. 17. Architecture overview of a plant growth simulation system. The plant growth simulator kernel on the left-hand side is in charge of three tasks: computing the plant topology using a morphogenetic description of the growth [de Reffye et al. 1988], computing the volume of organs due to the fabrication of fresh vegetable matter, and computing the geometry of the plant, using so called *allometry* rules. The second step (organ volume) uses the amount of light received by each leaf of the plant during the simulation step as an input variable.

7.2 Plant growth simulation

7.2.1 *Presentation.* We discuss here the application of our lighting simulation technique to plant growth simulation. A physiological plant growth simulator is coupled with our light simulation software. The former supplies, at each growing step, the topological and geometrical description of the plants and the latter responds with the amount of light received by the plants. Figure 17 shows an overview of this architecture.

Note that in the context of this paper, we are only considering the problem of light simulation, and treat the plant growth simulator as a “black box”. In our implementation we have used CIRAD’s AMAPhydro simulator, which computes the production of vegetal matter throughout the plant as well as the stimulation for the growth of each organ based on water transpiration, directly linked to the amount of received light energy. This allows to automatically simulate architectural and shape variations in plants, such as those shown in Figure 18 (*left*). In addition to its intensity, the directionality of light is used to have plant axes aim towards the light sources to maximize the amount of light energy received by their leaves (*e.g* Figure 18-*center*). Details about this simulator can be found elsewhere [de Reffye et al. 1988; (de) Reffye et al. 1996; Blaise et al. 1998; (de) Reffye et al. 1999] [3]. Note that other growth simulation modules could easily be used instead [Měch and Prusinkiewicz 1996].

The combination of eco-physiological laws in the plant growth simulator and proper light simulation allows precise modeling of the modification of the plant architecture, shape and size throughout the simulation, and accurate response to external factors. A key ingredient of the system is thus the lighting simulator. Unfortunately the distribution of light in a plant is highly non-uniform due to diffusion, self-shadowing, and indirect lighting. These

[3]In addition to the lighting simulation module, other modules exist but are not discussed here, such as a mechanical simulator to deal with internal mechanical constraints when inferring the geometry of branches.

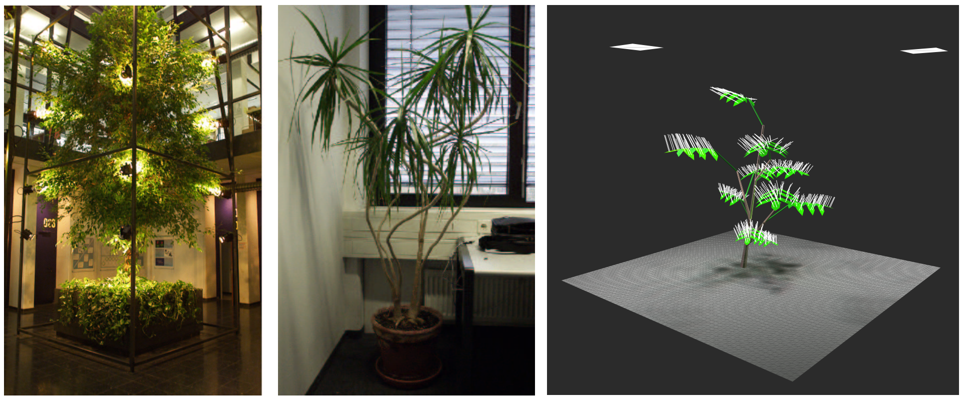


Fig. 18. *left* and *center* : Two examples of the light influence on plant growth: On the *left* the growth rate of the plant is not uniform throughout its body (higher in the regions close to the light bulbs); in the *center* image the plant has been rotated at regular intervals during growth, hence the twisted shape of its trunk due to its constant aiming toward the window. The image on the *right* shows *light directionality vectors* as computed using equation 3, for each polygon of the leaves of a simple plant, lit by four light sources. The directionality is easily perceptible on this picture.

phenomena must all be simulated to obtain meaningful intensity and lighting direction information for each growing structure of the plant. Thus we have two conflicting goals. We need a method that works at many scene scales: a single plant as well as a large number of trees; and we want accurate predictions at the finest scale, i.e. for each individual leaf. This makes the Hierarchical Instantiation algorithm a perfect candidate for this task.

Collecting information for the growth simulator. The physiological plant growth simulator needs to know for each growth cycle the intensity of light energy received by the leaves as well as the average direction of incoming light. Both quantities are obtained on the fly during the computation, when the concerned geometry is present in memory: the irradiance is obtained by summing the contribution of the links that end on all parents of the current leaf. The main irradiance direction v_i for leaf i is obtained by summing the irradiance vector contributions $I(E_j)$ from the emitters E_j of the links on the parents of i in the following way :

$$\vec{v}_i = \frac{1}{\sum_{E_j \in P(i)} I(E_j)} \sum_{E_j \in P(i)} I(E_j)\vec{v}(E_j) \quad \text{using} \quad \vec{v}(E_j) = \int_{E_j} \frac{y-x}{\|y-x\|^3} \cos\theta dy \qquad (3)$$

Using this formula, the more “directional” the incoming light is, the larger is the norm of $\vec{v}_i$. If incoming light is uniformly distributed around a leaf, $\vec{v}_i$ will be $\vec{0}$ (See example on Figure 18 *right*).

One important advantage of the instantiation algorithm over classical radiosity and other explicit methods is that we do not need to push the exploration of the scene down to the level of the leaves to simulate the equilibrium of light energy at higher levels of the hierarchy: before opening instances, the algorithm already provides an accurate global solution based on the information encoded into the phase and transmittance functions of the instances. As soon as the plant growth simulator can be satisfied with lighting information at the level of bigger structures than single leaves (branches for instance) is thus becomes

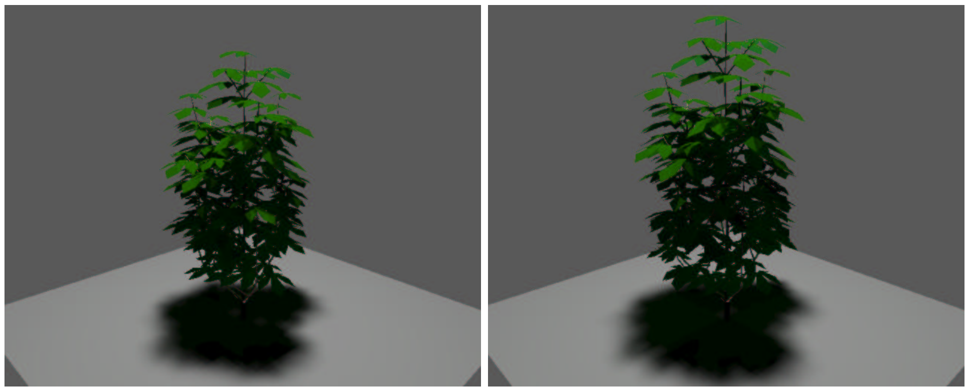


Fig. 19. Influence of indirect lighting on growth simulation. The left-hand plant was subjected only to direct lighting. Note the overall difference in size, as well as the subtle differences in the plant architectural balance.

possible to drastically accelerate the growth process by limiting the depth of the traversal of the hierarchy of instances during each lighting simulation step. When computing images however, the geometry still needs to be accessed and the instances have to be visited in depth. We limit the computation in such a case to a push-pull of the radiant energy computed at higher levels down to the level of the geometry.

7.2.2 *Experiments.*

7.2.2.1 *Influence of indirect lighting.* We present a test on indirect lighting to determine the importance of simulating this phenomenon during the growth process. This justifies the use of an algorithm capable of simulating global illumination rather than simply measuring direct lighting through the vegetation.

Figure 19 shows the same plant grown with direct lighting only on the left, and normal lighting on the right. Measuring the difference between the two results using the norm described above, we find (and it is visible on the pictures) that the error is more than 10% of the size of the plant (for comparison, the error due to instantiation approximations is less than 0.1% at that cycle).

Within the visible light spectrum range, diffusion of light inside plants foliage is limited by the small reflectance and transmittance values of the leaves (about 0.04: see [Govaerts 1995] for typical examples). In the IR domain however, internal diffusion of light becomes much more significant because reflectances and transmittances of plant leaves take values up to 40% [Govaerts 1995]. Simulating the IR distribution is useful to internal temperature profile in plants.

Indirect lighting may not only come from the vegetation itself but also from objects in the scene. A rather common example is that of a plant growing next to a wall (in front of a house for instance). Then a significant part of the light energy received by the plant comes from the surrounding objects.

7.2.2.2 *Virtual plants grown.* Figure 20 shows a plant that starts growing behind a wall. The production of vegetal matter and the growth rate of organs that receive light, as well as the seek for incoming light, favor the growth of vegetative axes trough the hole. Successive images correspond to cycles 2, 9, 13,17 and 25. These images were

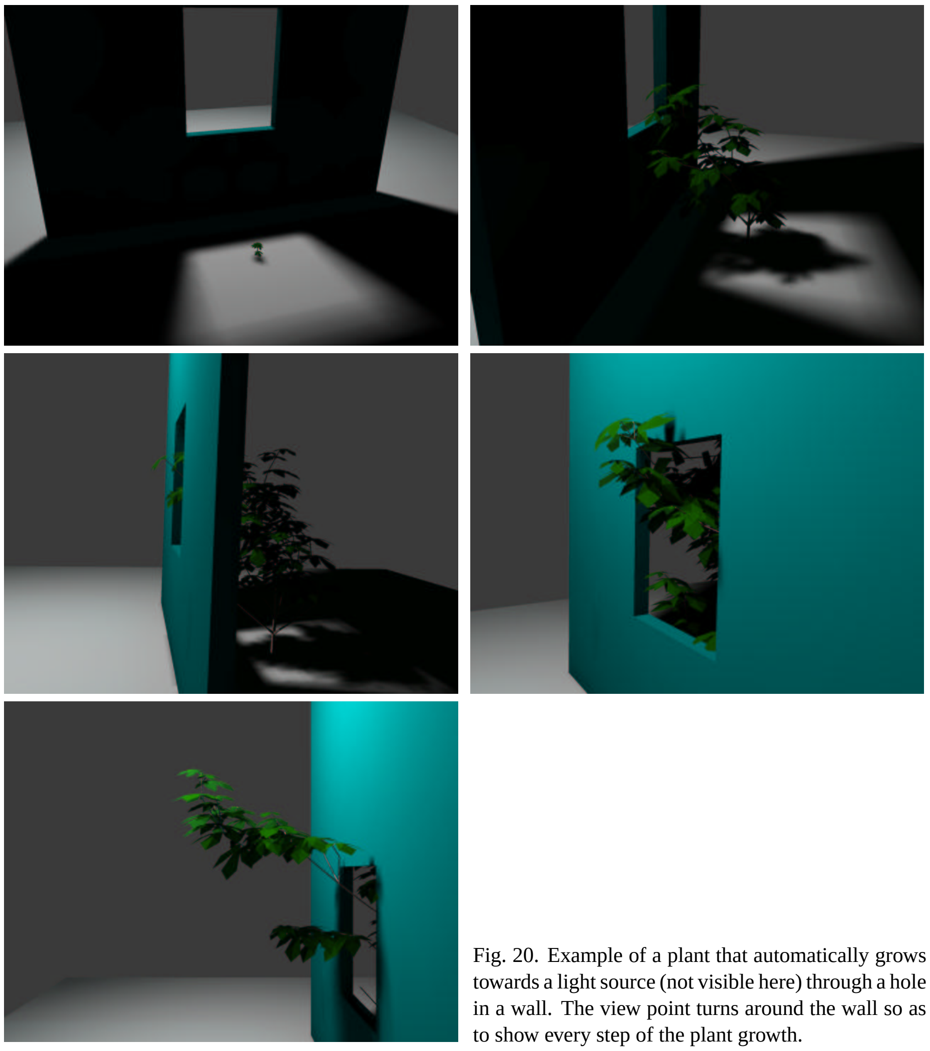


Fig. 20. Example of a plant that automatically grows towards a light source (not visible here) through a hole in a wall. The view point turns around the wall so as to show every step of the plant growth.

rendered in a second pass, using our radiosity simulator, from the geometric data produced during the simulation by the plant growth engine, but using a set of parameters that is more specifically adapted to visual results. Such an accuracy in the shadows, for instance, is not at all necessary for growing the plant itself. The total simulation time for growing this plant is less than 20 minutes on a workstation equipped with a 250 MHz *R*12000 Processor.

In Figure 21 we show an example of a plant growing under a light source. The images correspond to cycles 13,22,29 and 34. It appears clearly at cycle 22 that the overall balance of the plant is influenced by light. Indeed the part that is closer to the light source receives more energy and thus produces more vegetal matter, hence the higher density of the foliage. Directionality of the axis toward the light source is not as obvious here as it was on the previous example (although it can be seen on the second image) because of the larger rigidity of the branches for that species. The last image also shows the transparency effect of the leaves. Total simulation time here is 1 hour 30 minutes.

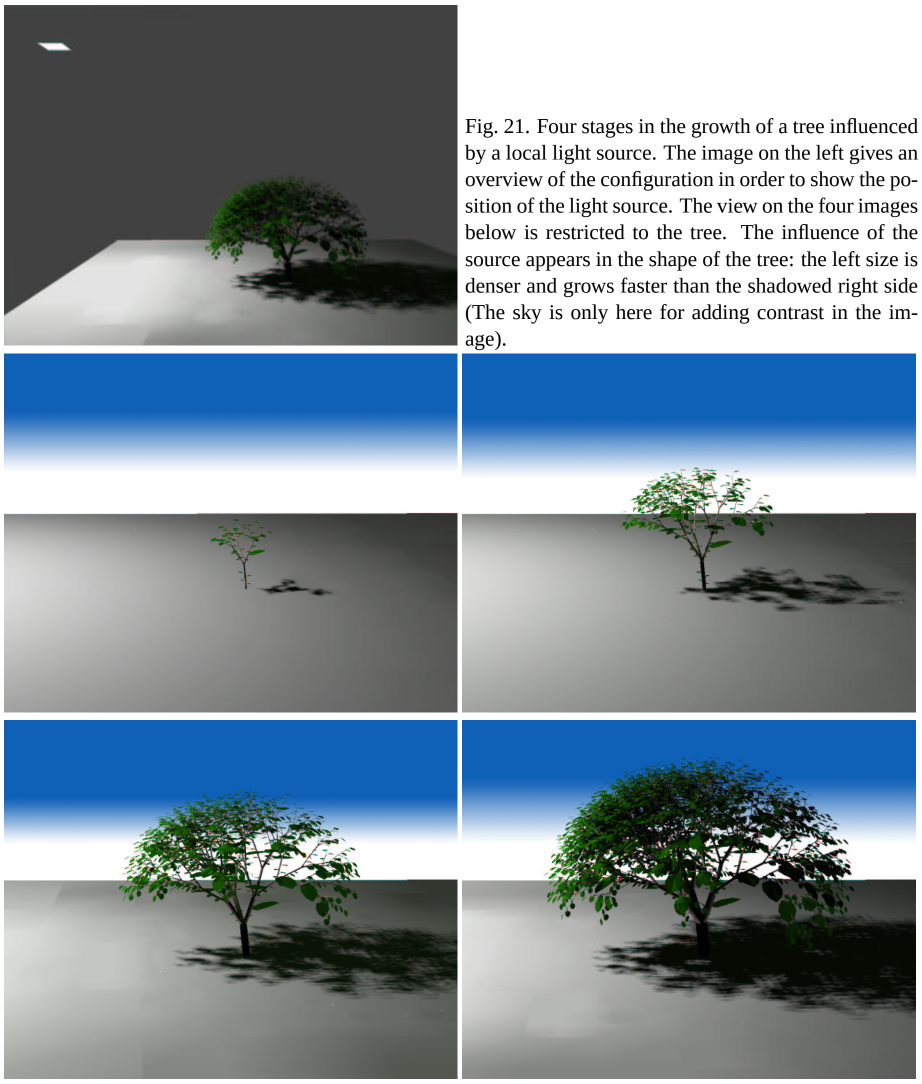


Fig. 21. Four stages in the growth of a tree influenced by a local light source. The image on the left gives an overview of the configuration in order to show the position of the light source. The view on the four images below is restricted to the tree. The influence of the source appears in the shape of the tree: the left size is denser and grows faster than the shadowed right side (The sky is only here for adding contrast in the image).

8. CONCLUSIONS AND FUTURE WORK

We have presented a complete system for lighting simulation in vegetation scenes, based on instantiation of similar structures in the plants. The classical hierarchical radiosity paradigm has been adapted in order to cope with the new representation of the scene, which is partially geometric and partially virtual (represented by phase functions) during the computation. We also have presented a complete study on the computation and storage of such data. The gain in memory and also computation time allows to treat scenes of thousands of trees and millions of polygons using a very reasonable amount of memory.

Our lighting simulation system has applications in both image synthesis and agronomy. We have presented an example of each application. More specific applications into the field of computer graphics include the computation of geometric plant models adapted to their environment, simulating plantation growth under various conditions for optimization of cultures in agronomy or for urban modeling and site planning. These applications only depend on the interaction of our lighting simulation system with a growth engine based on a detailed simulation of the internal functioning of the plant, accounting for incoming light energy.

We believe that a few improvements can be brought to the hierarchical instantiation algorithm. As discussed earlier, treating an instance as a directionally anisotropic medium is not very suitable for the production of accurate shadows. The solution we found to this problem was to instantiate plants at a lower level, at the expense of a larger memory cost. Another solution to this problem is to choose a different compromise between a coarse, but memory-cheap representation of visibility, and a memory-consuming but also more accurate model inside the instances. Billboards, the convex hull of the instance content, a quadtree of internal density would be such examples. Another complex issue that deserves further study is the effect of opening an instance on links that simply *cross* the instance: sometimes, the extra information available when opening the instance would probably be useful to decide further refinement of such links. This is however complicated by the fact that the emitting and/or receiving ends of such links may not be present in memory at that time.

Future work also includes the use of the instantiation algorithm in order to calibrate the reaction of plant growth simulators to lighting simulation: the growth engine we have used, for instance, is perfectly calibrated to statistical data measured on real plants (in terms of organic matter production and organ sizes). However we still need to infer the parameters involved into the reactions to light. This can be done using hidden-parameters identification techniques.

REFERENCES

ASHDOWN, I. 1994. *Radiosity: A Programmer's Perspective*. John Wiley & Sons, New York, NY.

BALANDIER, P., LACOINTE, A., ROUX, X. L., SINOQUET, H., CRUIZIAT, P., AND DIZÈS, S. L. 2000. Simwal: A structural-functional model simulating single walnut tree growth in response to climate and pruning. *Ann. For. Sci. 57*, 571–585.

BARANOSKI, G. V. G. AND ROKNE, J. G. 1997. An algorithmic reflectance and transmittance model for plant tissue. In *Computer Graphics Forum (Proc. Eurographics '97)*. Vol. 16(3).

BEAUDET, M. AND MESSIER, C. 1998. Growth and morphological responses of yellow birch, sugar maple, and beech seedlings growing under a natural light gradient. *Canadian Journal Forest Research 30*, 1007–1015.

BEAUDET, M., MESSIER, C., HILBERT, D. W., LO, E., WANG, Z. M., AND LECHOWICZ, M. J. 2000. Leaf- and plant-level carbon gain in yellow birch, sugar maple and beech seedlings from contrasting forest light environments. *Canadian Journal Forest Research 30*, 390–415.

BLAISE, F., BARCZI, J., JAEGER, M., DINOUARD, P., AND DE REFFYE, P. 1998. Simulation of the growth of plants, modeling of metamorphosis and spatial interactions in the architecture and development of plants. *Cyberworlds 6*, 81–109.

BOREL, C. C., GERSTL, S. A. W., AND POWERS, B. J. 1991. The Radiosity Method in Optical Remote Sensing of Structured 3-D Surfaces. *Remote Sensing of the Environment 36*, 13–44.

CASTRO, F. D. AND FETCHER, N. 1998. Three-dimensional model of the interception of light by a canopy. *Agricultural and Forest Meteorology 90*, 215–233.

CHELLE, M., ANDRIEU, B., AND BOUATOUCH, K. 1998. Nested radiosity for plant canopies. *The Visual Computer 14*, 3, 109–125.

CHEN, S., IMPENS, I., CEULEMANS, R., AND KOCKELBERGH, F. 1993. Measurement of gap fraction of fractal generated canopies using digitalized image analysis. *Agricultural and Forest Meteorology 65*, 245–259.

DAUZAT, J. AND M.N., E. 1987. Simulating light regime and intercrop yields in coconut based farming systems. *European Journal of Agronomy 7*, 63–74.

(DE) REFFYE, P., BLAISE, F., CHEMOUNY, S., JAFFUEL, S., FOURCAUD, T., AND HOULLIER, F. 1999. Calibration of a hydraulic architecture-based growth model of cotton plants. *Agronomie 19*, 265–280.

DE REFFYE, P., EDELIN, C., FRANCON, J., JAEGER, M., AND PUECH, C. 1988. Plant models faithful to botanical structure and development. In *Computer Graphics (SIGGRAPH '88 Proceedings)*, J. Dill, Ed. Vol. 22. 151–158.

(DE) REFFYE, P., FOURCAUD, T., BLAISE, F., BARTHÉLÉMY, D., AND HOULLIER, F. 1996. An ecophysiological model for tree growth and tree architecture. In *Workshop on Functional Structural Tree Models. Helsinki*. Silva Fennica eds.

DEUSSEN, O. AND STROTHOTTE, T. 2000. Computer-generated pen-and-ink illustration of trees. In *Siggraph 2000, Computer Graphics Proceedings*, K. Akeley, Ed. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 13–18.

FOURNIER, C. AND ANDRIEU, B. 1999. Adel-maize: an L-system based model for the integration of growth processes from the organ to the canopy. *Agronomie 19*, 313–327.

GASTELLU-ETCHEGORRY, J., DEMAREZ, V., PINEL, V., AND ZAGOLSKI, F. 1996. Modeling radiative transfer in heterogeneous 3-d vegetation canopies. *Remote Sensing of Environment 58*, 2, 131–156.

GASTELLU-ETCHEGORRY, J., ZAGOLSKI, F., AND ROMIER, J. 1996. A simple anisotropic reflectance model for homogeneous multilayer canopies. *Remote Sensing of Environment 57*, 22–38.

GAUTIER, H., MĚCH, R., PRUSINKIEWICZ, P., AND VARLET-GRANCHER, C. 2000. 3d architectural modeling of aerial photomorphogenesis in white clover (*trifolium repens* l.) using L-systems. *Annals of Botany 85*, 359–370.

GOEL, N. 1988. *Models of vegetation canopy reflectance and their use in estimation of biophysical parameters from reflectance data.* Gordon & Breach Publishing Group.

GOEL, N. S., ROZEHNAL, I., AND THOMPSON, R. L. 1991. A computer graphics based model for scattering from objects of arbitrary shapes in the optical region. *Remote Sensing of Environment 36*, 2, 73–104.

GOLDSMITH, J. AND SALMON, J. 1987. Automatic creation of object hierarchies for ray tracing. *IEEE Computer Graphics and Applications 7*, 5 (May), 14–20.

GORAL, C. M., TORRANCE, K. E., GREENBERG, D. P., AND BATTAILE, B. 1984. Modelling the interaction of light between diffuse surfaces. In *Computer Graphics (SIGGRAPH '84 Proceedings)*. Vol. 18. 212–22.

GOVAERTS, Y. M. 1995. A model of light scattering in three-dimensional plant canopies: A monte carlo ray tracing approach. Ph.D. thesis, Departement de Physique, Université Catholique de Louvain, Louvain, Belgium.

GREENE, N. 1989. Voxel space automata: Modeling with stochastic growth processes in voxel space. In *Computer Graphics (SIGGRAPH '89 Proceedings)*, J. Lane, Ed. Vol. 23. 175–184.

HANRAHAN, P., SALZMAN, D., AND AUPPERLE, L. 1991. A Rapid Hierarchical Radiosity Algorithm. In *Computer Graphics (ACM SIGGRAPH '91 Proceedings)*. Vol. 25. 197–206.

HASENFRATZ, J. M., DAMEZ, C., SILLION, F., AND DRETTAKIS, G. 1999. A practical analysis of clustering strategies for hierarchical radiosity. In *Computer Graphics Forum (Proc. Eurographics '99)*. Vol. 18. 221–232.

J.K., R. AND A.L, M. 1988. Calculation of canopy bidirectional reflectance using the monte carlo method. *Remote Sensing of the Environment 24*, 213–225.

KAJIYA, J. T. 1986. The Rendering Equation. In *Computer Graphics (ACM SIGGRAPH '86 Proceedings)*. Vol. 20. 143–150.

MAX, N., MOBLEY, C., KEATING, B., AND WU, E.-H. 1997. Plane-parallel radiance transport for global illumination in vegetation. In *Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering)*, J. Dorsey and P. Slusallek, Eds. Springer Wien, New York, NY, 239–250. ISBN 3-211-83001-4.

MĚCH, R. AND PRUSINKIEWICZ, P. 1996. Visual models of plants interacting with their environment. In *SIGGRAPH 96 Conference Proceedings*, H. Rushmeier, Ed. Annual Conference Series. ACM SIGGRAPH, Addison Wesley, 397–410. held in New Orleans, Louisiana, 04-09 August 1996.

MYNENI, R., ROSS, J., AND ASRAR, G. 1989. A review on the theory of photon transport in leaf canopies in slab geometry. *Agric. For. Meteorol. 45*, 1–153.

NORMAN, J. AND JARVIS, P. 1975. Photosynthesis in sitka spruce (*picea sitchensis*(bong) carr.). v. radiation penetration theory and a test case. *Journal of Applied Ecology 12*, 839–878.

OUHYOUNG, M., CHUANG, Y.-Y., AND LIANG, R.-H. 1996. Reusable Radiosity Object. In *Computer Graphics Forum*. Vol. 15. C348–C356.

PEARCY, R. AND SIMS, D. 1998. A three-dimensional shoot architecture model for assessment of light capture and carbon gain by understory plants. *Agricultural and Forest Meteorology 89*, 241–253.

PERTTUNEN, J., SIEVÄNEN, R., NIKINMAA, E., SALMINEN, H., SAARENMAA, H., AND VÄKEVÄ, J. 1996. Lignum: a tree model based on simple structural units. *Annals of Botany 77*, 87–98.

PLANCHAIS, I. AND SINOQUET, H. 1996. Foliage determinants of light interception in sunny and shaded branches of *fagus ylvatica*(l.). *œcologia 108*, 1–12.

RAUSCHER, H., ISEBRANDS, J., HOST, G. E., DICKSON, R. E., DICKMANN, D. I., CROW, T. R., AND MICHAEL, D. A. 1990. Ecophys: An ecophysiological growth process model for juvenile poplar. *Tree Physiology 7*, 255–281.

ROSS, J. 1981. *The radiation regime and architecture of plant stands*. Junk Pub., The Hague.

RUSHMEIER, H. E., PATTERSON, C., AND VEERASAMY, A. 1993. Geometric Simplification for Indirect Illumination Calculations. In *Proceedings of Graphics Interface '93*. Morgan Kaufmann, San Francisco, CA, 227–236.

SIEGEL, R. AND HOWELL, R. J. 1992. *Thermal radiation heat transfer*. Hemisphere Publishing Corporation.

SILLION, F. 1995. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Transactions on Visualization and Computer Graphics 1,* 3 (Sept.). (a preliminary version appeared in the fifth Eurographics workshop on rendering, Darmstadt, Germany, June 1994).

SILLION, F. AND DRETTAKIS, G. 1995. Feature-Based Control of Visibility Error: A Multiresolution Clustering Algorithm for Global Illumination. In *Computer Graphics Proceedings, Annual Conference Series, 1995 (ACM SIGGRAPH '95 Proceedings)*. 145–152.

SILLION, F., DRETTAKIS, G., AND SOLER, C. 1995. A Clustering Algorithm for Radiance Calculation in General Environments. In *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, P. M. Hanrahan and W. Purgathofer, Eds. Springer-Verlag, New York, NY, 196–205.

SILLION, F. AND PUECH, C. 1994. *Radiosity and Global Illumination*. Morgan Kaufmann publishers, San Francisco.

SMITS, B., ARVO, J., AND GREENBERG, D. 1994. A Clustering Algorithm for Radiosity in Complex Environments. In *Computer Graphics Proceedings, Annual Conference Series, 1994 (ACM SIGGRAPH '94 Proceedings)*. 435–442.

SOLER, C. AND SILLION, F. 2000. Hierarchical instantiation for radiosity. In *Rendering Techniques '00*, B. Peroche and H. Rushmeier, Eds. Springer Wien, New York, NY, 173–184.

TAKENAKA, A. 1994. A simulation model of tree architecture development based on growth response to local light environment. *Journal of Plant Research 107*, 321–330.

VERHOEF, W. 1984. Light scattering by leaf layers with application to canopy reflectance modeling: The sail model. *Remote Sensing of Environment 16*, 125–141.

WHITEHEAD, D., GRACE, J., AND GODFREY, M. 1990. Architectural distribution of foliage in individual *pinus radiata* (d. don) crowns and the effect of clumping on radiation interception. *Tree physiology 7*, 135–155.

Received Month Year; revised Month Year; accepted Month Year