Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Graphics 29 (2005) 203–208
COMPUTERS & GRAPHICS
www.elsevier.com/locate/cag

A polar-plane-based method for natural illumination of plants and trees

María J. Vicent, Vicente Rosell, Roberto Vivó*

Dpto. SIC, Universidad Politécnica de Valencia, Camino de Vera s/n, 46022 Valencia, Spain

Abstract

We introduce an illumination method for outdoor scenes containing plants and trees. Our method supports multiresolution plant and tree models based on random L-systems. The method includes modeling and rendering of the sky at different times for different locations on the earth. We propose an efficient algorithm to illuminate the leaves of a tree using form factors pre-computed with a polar-plane-based method. We also present two solutions to the visibility problem, a heuristic solution and a solution integrated with the polar-plane method.

Keywords: Natural lighting; Form factors; Plant and tree rendering

1. Introduction

Simulating light interaction with plants supports both generating their appearance attributes, like color, brightness and transparency, and simulating their growth and their interaction with the environment. Rendering realistic outdoor images with plants and trees requires using lighting models that take into account natural light sources, like the sky and the sun. Also, the complexity of plant and tree models prevents the use of traditional global illumination algorithms.

Most plant and tree rendering algorithms are based on texture mapping. Jakulin [1] proposes an interactive tree rendering algorithm that defines boxes or *slicings* that contain the tree's branches and leaves. Slicings are defined for certain directions parallel to the ground. They are divided into *slices* texture-mapped with those branches and leaves that are closest to each slice. To render the slices from an arbitrary direction, Jakulin blends the two slicings closest to the desired view. Texture maps are rendered using ray-tracing software.

The method proposed by Meyer et al. [2] generates a hierarchy of bidirectional textures (HBTs). For each pair illumination direction-viewing direction, the method pre-computes a self-oriented impostor. At rendering time, a new image is rendered by interpolating the impostors closest to the desired pair illumination direction-viewing direction.

Other plant and tree rendering algorithms use local illumination models. For example, Qin et al. [3] propose an algorithm that uses a set of 2D buffers containing surface shading information.

Complex radiosity algorithms can also be used for plant and tree rendering. Sillion [4], for instance, uses hierarchical radiosity with grouping. Max et al. [5] employ a radiance transport method. Their assumption is that radiance depends only on the direction of the light flow and the height with respect to the ground. This makes the method applicable to dense vegetation, instead of single isolated trees. Hondermarck and Chelle [6] use a nested radiosity method that distinguishes between close and far radiosity, depending

*Corresponding author. Tel.: +34 9638 77795; fax: +34 9638 77359.
E-mail address: rvivo@dsic.upv.es (R. Vivó).

0097-8493/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2004.12.005

on the distance between the polygons involved in the computation. Recently, Soler and Sillion [7] proposed a method based on hierarchical instantiation that identifies objects that share a common behavior in the presence of light.

We want to use natural light sources for rendering plants and trees. The most common approach divides the sky into areas, each represented by a light source located at infinity, exactly like the sun. Müller et al. [8] and Daubert et al. [9] integrate natural light sources into radiosity algorithms. The sky's complex lighting features have been modeled using specific methods like those described in [10] and [11].

Summarizing, most of these plant and tree rendering algorithms are good for dense vegetation. They even achieve interactive rates using texture mapping without natural lighting. Most global illumination methods use hierarchical radiosity with clustering. They are highly sensitive to the hierarchy's quality.

We propose a lighting method for outdoor scenes with plants and trees. We model the plants and trees using a multiresolution representation based on random L-systems. We describe how we model and render the sky at different times of day and different places on the earth. With that model we light the leaves of plant and tree models.

2. Polar-plane illumination

2.1. Sky model

Using any illumination method requires three steps: (i) modeling the light sources, (ii) defining a method to determine how much light reaches each point in the scene, and (iii) computing a color for each relevant point in the scene.

We use a sky model like the one described by Preetham et al. [11]. The model is a triangle mesh that approximates a hemisphere. We triangulate the hemisphere one level at a time. Triangle vertices at the finest level are the point samples we use to compute the radiance. For each sky sample point we compute and store its color as a function of time of day, day and position on the earth (longitude and latitude).

This model represents the sky as a set of point light sources located at infinity. These sources are later used by an illumination algorithm to obtain the color of certain surfaces with pre-determined orientations. Both steps, computing the lighting from the sky and coloring the surfaces, can be accomplished during pre-processing. That way, at rendering time, we can illuminate a surface with an arbitrary orientation by interpolating between surfaces already illuminated.

2.2. Polar plane

The intensity arriving at any polygon can be computed assuming that the polygon is located at the center of the hemisphere representing the sky. This is because we assume that the light arriving from the sky does not depend on the height of the polygon above ground.

Computing the illumination for any polygon is an expensive task. Hence, we propose pre-computing the illumination for a fixed set of orientations and storing it in a look-up table. We determine the illumination of a polygon with a non-pre-computed orientation by interpolating between the illumination information stored for the four closest directions to the polygon's orientation.

We use the polar-plane method to obtain the form factors needed to determine the pre-computed illuminations and the illumination of any given polygon, including its visibility in the scene. In our illumination model, the form factor is used to model the amount of radiance emitted by a surface, the sky, which reaches another surface, the leaves or the ground.

The polar-plane method computes form factors using finite elements [12–14]. It uses an infinite plane parallel to a differential surface area dA_i containing the point of interest x, whose form factor we want to calculate. The polar plane can be divided into concentric rings, so that each ring has the same form factor [15]. If we divide the plane into n rings, then each ring has a form factor of $\Delta FF = 1/n$. We also discretize each ring into m sectors with the same angle. The form factor of each sector is $\Delta ff = 1/(nm)$. We can then compute the form factor of a given polygon by adding the form factors of the sectors covered by the polygon.

2.3. Polar plane for pre-determined orientations

We select a set of pre-determined orientations using spherical coordinates defined on a hemisphere of radius 1. We identify each orientation by two angles (θ, ϕ) that represent elevation and azimuth.

The set of directions we obtain is (θ_i,ϕ_j) for $i \in [0\ldots,\text{stacks}]$ and $j \in [0\ldots,\text{slices}-1]$ where each θ_{i+1} is obtained from $\theta_{i+1} = \theta_i + \Delta\theta$ with $\theta_0 = 0$ and $\Delta\theta = p/(2\cdot\text{stacks})$, and each ϕ_{j+1} is obtained likewise by discretizing an angle of 2π radians.

We assign to each sector the luminance of the portion of the sky that projects onto that sector. The sum of the luminances for all the sectors covered by a sky polygon gives the maximum luminance associated to that orientation for each hour. This is assuming that there are no occlusion issues due to the presence of other objects.

To assign a luminance to each sector we first determine the sky triangle that the sector projects onto (see Fig. 1). A sector projects onto a triangle if its center (cx,cy,cz) projects onto the interior of the triangle. We compute the luminance by interpolating between the luminances at the three vertices of the sky triangle. The

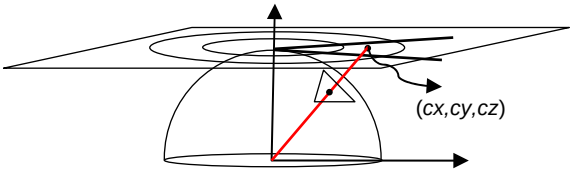

Fig. 1. Projecting the center point of a sector onto the hemisphere.

total number of operations needed is minimal, since every time we process a new sector we start with the level-0 sky triangles and then we traverse the hierarchy of subdivision triangles until a bottom-level triangle is reached.

The center point of a sector is the intersection point between the radius that halves the sector and a special circumference. That circumference is defined to divide the sector into two sub-sectors, each with the same form factor.

For each pre-computed orientation we store the normal to its associated polar plane, the maximum luminance (color) for each hour, and the actual polar plane. For each polar plane we store the radii of the rings, the radii of the special circumferences described above, and the center point of each sector. The radii of the rings are the same for each orientation, since we use the same plane subdivision scheme. For each sector we also store the luminance associated to that orientation for each hour.

2.4. Illumination for arbitrary orientations

The lighting of each leaf of a tree depends on its orientation and its position, in that order. Each leaf is modeled using a single polygon. If we know the leaf's orientation we can easily obtain the maximum luminance arriving at it. That luminance corresponds to the visible portion of the sky at that time (hour of day). That is only if the leaf is isolated, that is, there are no occluders between the leaf and the sky.

The maximum illumination of an arbitrarily oriented polygon, for a given time of day, is computed by interpolating between those four pre-computed orientations closest to polygon's orientation. Given an arbitrary orientation (θ_a, ϕ_a) such that $i\Delta\theta \leqslant \theta_a \leqslant (i+1)\Delta\theta$ and $j\Delta\phi \leqslant \phi_a \leqslant (j+1)\Delta\phi$, the new orientation is located between the orientations: (i,j), $(i,j+1)$, $(i+1,j)$, and $(i+1,j+1)$.

3. Visibility

Tree leaves do not typically receive all the light corresponding to their orientation, since the leaves are not isolated and other neighboring leaves may occlude some of the incident light. To obtain a more realistic illumination we incorporate visibility into our lighting algorithm.

We implement different visibility techniques that trade off between rendering quality and rendering time. We use two techniques, a heuristic technique that obtains a visibility term V and other techniques based on using the polar plane.

3.1. Heuristic visibility

Our first heuristic technique is called Visibility_1 (see Fig. 2). Here the visibility factor depends on two values: the cosine of the angle between the position vectors of the leaf and the sun, and the distance from the leaf to the center of the tree. This distance value takes into account the fact that leaves inside the tree receive less light than those on the outside.

This technique has the following problem: it is good for trees with a compact set of leaves, but it does not work well when the leaves are spread out. So, we propose a second algorithm called Visibility_2 (see Fig. 3). This algorithm computes an estimate of the percentage of the sky occluded for any given leaf i. The estimate considers the number of leaves outside of leaf i and approximates the luminance that those leaves occlude. It takes into account the number of leaves outside leaf i, the amount of light each leaf occludes and

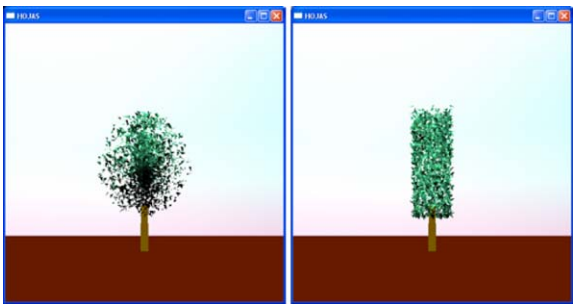

Fig. 2. Visibility_1. At 12 noon. Rendering time: 1 s.

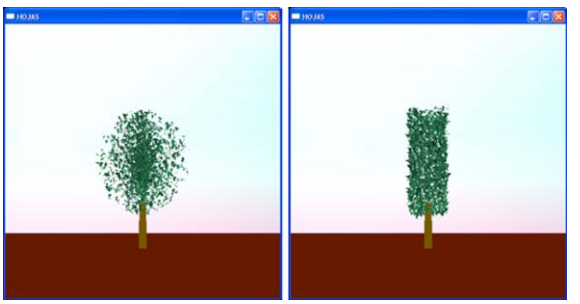

Fig. 3. Visibility_2. Estimating occlusion terms. At 12 noon. Rendering time: 41 s.

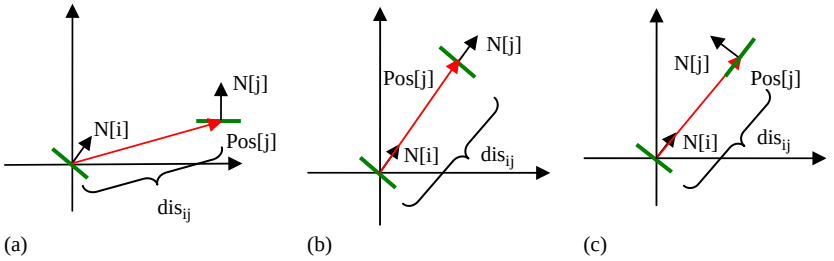

Fig. 4. Parameters used in the estimation of the occlusion terms.

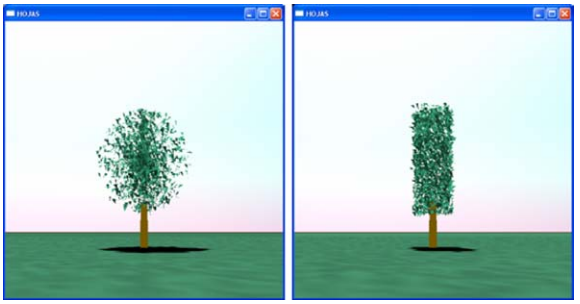

Fig. 5. Visibility_3. Estimating occluded polar-plane sectors. At 12 noon. Rendering time: 73 s.

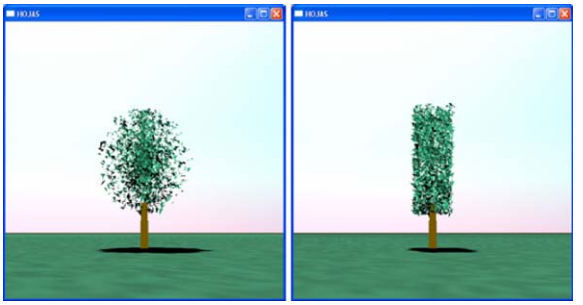

Fig. 6. Visibility_PolarPlane. At 12 noon. Rendering time: 3260 s.

the dispersion of the portions of the sky that are occluded. The occlusion term for leaf i partially occluded by leaf j is determined using the distance between both leaves and the cosine of the angle between their normals $N[i]$ and $N[j]$ (see Fig. 4(a)). Figs. 4(b) and (c) show the two extreme cases, when the occlusion is maximized and when it is minimized.

We implement a third visibility algorithm, Visibility_3 (see Fig. 5). We approximate the visibility of a leaf by estimating the number of occluded polar-plane sectors. The occlusion term O_j depends on the distance dis_{ij} between the leaves i and j, the angle between both leaves' normals, and the angle between the normal to leaf i and the position vector of leaf j (see Fig. 4). Also, this algorithm takes into account the orientation of the leaf with respect to the sun. If the leaf is oriented towards the sun, the luminance of the visible sectors is increased.

3.2. Polar-plane-based visibility

We present a visibility algorithm based on the polar plane, Visibility_PolarPlane (see Fig. 6). To compute visibility for leaf i, we obtain its polar plane by interpolating between the planes oriented closest to the leaf's orientation. Each sector of the leaf's polar plane contains the luminance of the part of the sky that projects onto it. The sum of the sectors' luminances is the maximum possible radiance arriving at the leaf given its orientation and the time of day.

The rendering algorithm uses an auxiliary occlusion matrix that contains one entry per sector. Each entry has a toggle value that says whether the sector is occluded or not. To light leaf i we project onto its polar plane all the leaves j located in i's visible subspace. For each leaf j we convert its projected vertices to polar coordinates and we determine which sectors are covered by the projection. A sector is considered covered when its central point, as defined above, is inside the projected polygon.

To speed up this algorithm we only process those sectors located between two rings: the largest ring and the smallest ring covered by leaf j's projection. The largest ring is given by the projected vertex located farthest from the rings' center. The smallest ring is given by either the rings' center or the projected vertex closest to the rings' center.

The following algorithm computes the luminance at leaf i, for any given time of day h:

```
Visibility (i,h)
For each ring c
    For each sector s
      Occlusion[c][s]← false
Lum [I][h]←Maximum_Luminance [i][h]
For each leaf j
    Project_Onto _Polar_Plane (j)
    Cmin←Compute smallest ring covered by projection
```

Fig. 7. Left, at 7:00am; center, at 12:00noon; and right, at 6:00pm.

```
Cmax ← Compute larges ring covered by projection
For each ring c between Cmin and Cmax
  For each sector s
    If (Interior(c,s))
      If (Occlusion [c][s] = false)
        Occlusion [c][s] ← true;
        Luminance[i][h] ← Lum[i][h]-Ppolar_i
[c][s][h]
```

3.3. Conclusions

Figs. 2–6 show two trees with 3000 leaves rendered using the algorithms described in this section. Each caption shows the time of day and the rendering time, which includes pre-processing time. The fastest method is heuristic visibility; the slowest method is polar-plane-based visibility. We conclude that fast low-precision algorithms are better for trees located far from the viewer, since the increase in quality is not noticeable when using high-precision algorithms.

4. Rendering an example tree

In this section we apply our polar-plane-based algorithm to the rendering of an example tree. The tree is represented using a multiresolution model based on random L-systems [16]. The model groups leaves into clusters associated to branches, and stores a bounding box for each cluster of leaves.

The ground is a quadrilateral mesh defined on the XY plane. The vertices have been randomly perturbed, so that we can simulate certain reflection conditions that affect both its color and the shadow cast by the tree.

For each tree branch we store a data structure that contains:

- an identifier,
- a pointer to the graphics primitive representing the branch: we use truncated cones, and store length and cap radii for each cone,
- a transformation matrix to locate the branch within the tree, and
- a toggle that tells whether the branch has leaves or not.

For each leaf we store the following information: its position, its normal, the coordinates of its vertices and the illumination for each time of day. For each random orientation related to a leaf, we store the polar plane in the same way as we do for pre-calculated orientations. But we add an extra flag that tells whether a sector is visible or not for that orientation. Finally, we compute the luminance of each sector by interpolating the luminances of those sectors corresponding to the closest pre-computed orientations.

Fig. 7 shows our example tree rendered using our illumination method. We have used luminance and chromaticity values for the sky at 14 different times of the day ranging from 6:00am to 7:00pm. The images were generated for Julian day 175. The location of the tree is in Valencia at 0.26917° longitude west and 39.28583° latitude north.

5. Conclusions and future work

We introduce a natural illumination algorithm that models the sky, the ground and the vegetation of an outdoor scene. We show how our algorithm illuminates leaves with random orientations and supports changes in illumination conditions.

The main part of our algorithm is the visibility method it implements to shade the leaves. The method approximates the amount of light arriving at a leaf, given the leaf's position and orientation and the set of leaves that occlude the light arriving at the leaf from the sky. We propose heuristic visibility methods that are faster than its physically based counterparts.

We can easily apply our algorithm to scenes containing multiresolution tree models generated with random L-systems. The models support changes in the leaf distribution to adapt it to the tree's topology. The

algorithm renders the scene including the ground and the shadows cast by the tree.

We can improve both the speed and the quality of the results of our illumination algorithm. Here is a list of suggested improvements:

- Apply multiresolution techniques to the structure of the polar planes. Illumination could then be computed with more or less precision depending on viewing distance. For long distances computations could be faster using *box–leaf* approach instead of a *leaf–leaf* approach.
- Increase realism by improving the geometric model of the leaves and by including parameters to characterize the leaves' tissue.
- Implement a multiresolution representation for the branches of the tree. Such a representation can be used to generate branch and leaf textures and adapt them to different illumination conditions.

Acknowledgments

This work was partially supported by Grant TIC2002-04166-C03-01 of the Spanish Ministry of Science and Technology.

References

[1] Jakulin A. Interactive vegetation rendering with slicing and blending. In: Proceedings of the short papers EUROGRAPHICS 2000.
[2] Meyer A, Neyret F, Poulin P. Interactive rendering of trees with shading and shadows. In: Proceedings of the Eurographics rendering workshop 2001.
[3] Qin X, Nakamae E, Tadamura K, Nagai Y. Fast photo-realistic rendering of trees in daylight. Computer Graphics Forum 2003;22(3):243–52.
[4] Sillion F. Clustering and volume scattering for hierarchical radiosity calculations. In: Proceedings of rendering techniques, 1994.
[5] Max N, Mobley C, Keating B, En-Hua W. Plane-parallel radiance transport for global illumination in vegetation. In: Proceedings of rendering techniques'97, Springer Computer Science 1997; p. 239–50.
[6] Hondermarck J, Chelle M, Renaud C, Andrieu B. Parallel form factors computation for radiative transfers in vegetation. In: Proceedings of the second eurographics workshop on parallel graphics and visualization, 1998.
[7] Soler C, Sillion F. An efficient instatiation algorithm for simulating radiant energy transfer in plant models. ACM Transactions on Graphics 2003;22(2): 204–33.
[8] Müller S, Kresse W, Schoeffel F. A radiosity approach for the simulation for the simulation of daylight. In: Proceedings of eurographics rendering workshop, 1995.
[9] Daubert K, Schirmacher H, Sillion F, Drettakis G. Hierarchical lighting simulation for outdoor scenes. In: Proceedings of rendering techniques'97, Springer Computer Science, Berlin: Springer; 1997. p. 229–38.
[10] Perez R, Seals R, Michalski J. All-weather model for sky luminace distribution—Preliminary configuration and validation. Solar Energy 1993;50(3):235–45.
[11] Preetham J, Shirley P, Smits B. A practical analytic model for daylight. In: Computer Graphics Proceedings 1999, SIGGRAPH'99, p. 91–100.
[12] Cohen MF, Greenberg DP. The hemi-cube: a radiosity solution for complex environments. Computer Graphics 1985;19(3):31–40.
[13] Sillion F, Puech C. A general two-pass method integrating specular and diffuse reflection. Computer Graphics 1989;23(3):335–44.
[14] Baum DR, Rushmeier HE, Winget JM. Improving radiosity solutions through the use of analytically determined form-factors. Computer Graphics 1989;23(3): 325–34.
[15] Vivó R, Vicent MJ, Lluch J, Molla R, Jorquera P. Study of the form factor calculation by single polar plane. In: Proceedings of the IASTED international conference on visualitation imaging and image processing, 2001.
[16] Lluch J, Camahort E, Vivó R. Procedural multirresolution for plant and tree rendering. In: Proceedings of second international conference on virtual reality, Computer Graphics, Visualization and Interaction in Africa. AFRIGRAPH, 2003, p. 31–8.