Orientation Lightmaps for Photon Radiosity in Complex Environments

A. Wilkie, R. F. Tobler and W. Purgathofer
Institute of Computer Graphics, Vienna University of Technology, Vienna, Austria
{wilkie, rft, wp}@cg.tuwien.ac.at

Abstract. We present a method that makes the use of photon radiosity methods feasible for complex scenes when a totally accurate solution is not essential. This is accomplished by using *orientation lightmaps*, which average the illumination of complex objects depending on the surface normal. Through this averaging, they considerably reduce the variance of the stochastic solution. For the use of these specialised lightmaps, which consume comparatively small amounts of memory, no changes have to be made to the actual photon tracing algorithm. Also, they can be freely mixed with normal lightmaps and inserted at any point in the scene description graph. This gives the user good control over the amount of inaccuracy he introduces by their application. The area computations necessary for their insertion are performed using a stochastic sampling method that performs well for highly complex objects.

1 Introduction

In the field of photorealistic computer graphics a particular problem has been evident for quite some time: with rising scene complexity sophisticated rendering algorithms (*i.e.* methods that take global illumination into account), which would yield better results, run into difficulties earlier than more primitive ones like *e.g.* "plain" raytracing. The main difficulties usually are that a given method would use far too much memory when applied to a complex scene, and/or that its execution time would be inacceptable.

The *modelling* of highly complex scenes is a problem where several useful proposals have been made; however, producing high–quality *renderings* of such scenes can pose a considerable problem.

In this paper, we present a new extension to a certain kind of stochastic photon radiosity simulation that blends seamlessly with the original approach, and which enables it to be conducted on scenes that contain objects of a complexity for which the direct application of the unmodified algorithm (and any other global illumination method) would not be feasible.

2 Previous work

In this section we give a brief overview of previous approaches to solving the problems associated with high scene complexity, both in the areas of modelling and of rendering.

2.1 Modelling

Unwieldy complexity is usually the hallmark of scenes containing natural objects auch as plants, terrain and generally outdoor environments. Work such as that of Prusin-

kiewicz and Lindenmayer [13] or the plant ecosystem effort by Deussen *et al.* [3] goes a long way to provide rendering systems with convincing models of scenes with huge geometric complexity. The technical ideas behind these efforts, such as rule–based object construction and instancing, have become commonplace in the computer graphics community, especially since they are usually also useful for scenes of normal complexity.

However, rendering such models can very often only be done using systems that were specifically written or modified for the purpose of proving one particular method. For example, Deussen *et al.* [3] describe a hybrid method that renders portions of the scene at one time; these α-Z images have to be combined in a later stage.

2.2 Cyclic CSG Graphs

Gervautz and Traxler [5] proposed a method which allows raytracing of scenes that contain complex objects defined by cyclic CSG graphs; the main focus of their effort was on rule–defined plants. During raycasting, objects defined in this way are created "on the fly" for intersection testing; when they are no longer needed, the objects are discarded again. It is not necessary to maintain an explicit representation of all the objects in memory; this allows for the treatment of huge scenes. Unfortunately, this method is so far restricted to "plain" raytracing, since the non–explicit nature of the scene graph prohibits the attachment of lightmaps or similar global illumination data structures to the objects; doing this would require unrolling of the cyclic graphs. Due to bad convergence properties, distribution raytracing has also turned out to be infeasible for natural scenes of this type.

2.3 Radiosity

Rushmeier *et al.* [14] suggested the use of geometric simplification (GSII) in order to reduce computation time for form–factor based radiosity. They eliminate small, isolated patches and replace clusters of objects with simple, optically equivalent boxes, which are used for the radiosity calculations. Their approach is an extension of a progressive multi–pass rendering method proposed by Chen *et al.* [1]. The authors develop a theoretical basis in order to determine when the use of GSII is appropriate in a scene.

2.4 View–dependent methods

There are a number of high–quality rendering methods available that generate viewpoint–dependent results. These sophisticated Monte Carlo global illumination methods like *e.g.* distribution raytracing, bidirectional path tracing, or metropolis light transport share one common disadvantage with respect to usage in complex scenes: namely that, due to the high number of sampling rays or paths cast, they are very dependent on the efficiency of raycasting, which gets significantly more expensive for large numbers of objects.

Although well–accelerated raycasting in practice scales sublinearly with the number of objects involved, the sheer size of *e.g.* outdoor scenes with vegetation poses a significant problem for these methods. For still images the effort required is, up to fairly large scenes, still tractable and the results satisfactory; however, if one wishes to render successive views of such a scene (*e.g.* for animation purposes), view–dependent methods start to warrant further investigation.

3 Monte Carlo Radiosity

The viewpoint–independent calculation and storage of the indirect illumination in scenes has so far been almost exclusively restricted to polygonal scenes. This is due to inherent restrictions of the finite–element approaches that are used; they are referred to as *form–factor based* approaches in the radiosity context. Most commercial radiosity software nowadays is form–factor based; these methods are (within their limits) proven and well–researched (*e.g.* by Goral [6], Cohen [2] or Shirley [15]).

Stochastic photon tracing algorithms[12, 16, 11, 4] for indirect illumination are still an area of ongoing research. Since they are more akin to a simulation of light transport than conventional radiosity, they can directly work on curved objects and easily accommodate phenomena such as refraction or specular surfaces, which are almost intractable by form–factor based approaches. Some approaches (*e.g.* Wilkie *et al.* [18]) also offer the possibility to directly operate on untesselated CSG models.

The main idea is to shoot packets of light energy (referred to as "photons") through a scene according to the physics of light transport, and record their interaction with the scene in some way. The reconstruction of the radiosity function over a surface is based on the random samples across that surface generated by the shooting pass.

Stochastic photon tracing algorithms can be classified according to what kind of data structure is used to store the information gathered during the shooting pass. We will present two main types and briefly discuss their behaviour for complex scenes.

3.1 Photon Maps

Jensen introduced the concept of *photon maps*[9, 10], where the incidences of the individual photons in the scene are stored in a global data structure with all their context such as incoming direction.

In order to compute the radiance L_r at a given point x in the scene, one basically has to locate the n photons nearest to x and add up their influences. The search for these n photons can be done efficiently if the photons are stored in a balanced kD–tree.

In practice, photon maps are not used alone during the rendering pass; for specular and highly glossy surfaces Monte Carlo sampling is used, and direct illumination is calculated using shadow rays.

Behaviour for complex scenes. Jensen[10] points out that his method, while it is (unlike most other photon tracing methods) not particularly prone to variance problems due to undersampling, has the disadvantage that it tends to use large amounts of memory even for comparatively simple scenes (*cf.* the memory usage statistics given in the paper), making it less than optimal for scenes with *e.g.* hundreds of thousands of objects.

3.2 Photon Radiosity

The second major approach to storing the samples gathered during the shooting pass is to use *photon lightmaps* that cover the faces of the geometric primitives (see figure 1) and store the energy deposited by photon hits. In this paper we refer to this version of stochastic photon tracing as *photon radiosity*.

We use the term "photon lightmaps" to reflect the usage of these data structures in a photon tracing environment; the concept of such radiosity textures was introduced in a bidirectional raytracing context under the name of *Rexes* by Heckbert in [7]. While

the data structures that we use for storing irradiance are similar, we employ a different, more efficient method of determining the actual indirect illumination.

Any kind of functional representation can be used for the recording of photon hits on these lightmaps; although higher–order bases have been used (*e.g.* by Zatz in [19]), the most commonly implemented solutions use constant and bilinear representations. This is mainly because the number of hits needed for a stable estimate rises sharply with the order of representation.

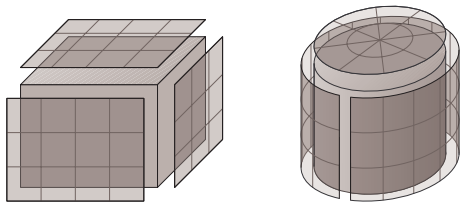

Fig. 1. Photon lightmaps applied to the faces of basic geometric primitives. While the "texels" on these lightmaps are separate (patch–like) "buckets" into which photons fall, the real object geometry is used during the simulation and the final rendering pass. Also, the whole lightmap is considered a coherent entity (as opposed to a sum of independent patches): a fact that is highly useful for interpolation purposes during the final rendering pass.

It has to be noted that the regular subdivision of the lightmaps shown in figure 1 is in no way mandatory; one can use any meshing and/or an adaptive hierarchical approach, as originally demonstrated by Heckbert [7] or (adapted for photon radiosity) by Tobler *et al.* [17] for this task.

The radiosity of a lightmap texel j is (at any time during the simulation)

$$B_j = \frac{1}{A_j} \cdot m_j \cdot \Phi \tag{1}$$

where m_j is the number of particles received by texel j, A_j its surface area and Φ the energy carried by one photon.

This shows the necessity that during the setup phase of the simulation the surface area has to be computed for each lightmap texel; a requirement that incurs a considerable execution time penalty if curved and arbitrarily transformed objects are used, especially in the presence of CSG intersections that may clip off part of the texels as discussed in Wilkie *et al.* [18].

Similar to Jensen's Photon Maps, photon lightmaps are normally not used alone — their advantages are best put to use in the context of a multipass renderer that *e.g.* uses area light source sampling to determine the direct illumination at a surface point, and the information in the lightmaps for all other contributions. The only modification one has to make to the photon radiosity algorithm to accommodate for this is, that in this case the lightmaps start recording photon hits only after their first bounce from a surface.

Behaviour for complex scenes. Photon radiosity in its original form takes badly to complex scenes for both the reasons stated in the introduction. The data structures that record photon hits are maintained locally for each primitve in the scene, leading to more or less fixed memory requirements per geometric primitive. While this is practicable for Cornell box environments, it does not work for *e.g.* a forest with a huge number

of needles on one tree alone. Also, in order to perform a meaningful reconstruction of the illumination on a surface a certain number of photon hits has to be recorded on each photon lightmap. This is clearly impractical for complex scenes, especially since the shooting of photons becomes more expensive as the number of participating objects increases. Finally, the cost of performing exact surface area calculations is prohibitive for large scenes.

It has to be noted that one has to address all these three issues (area estimation, memory consumption and convergence speed) simultaneously if one wants to make photon radiosity suitable for use in complex environments; every single one of these problems would render it impractical for the purpose.

4 Our Proposal — Orientation Lightmaps

We propose a lightmap–like data structure that averages all incoming irradiance for complex objects based on the *surface normal* of the photon hit point. We call this an *orientation lightmap* (OL for short) due to the surface normal dependent way it stores the illumination of the object it "covers". The proposal is in a way akin to the work of Rushmeier[14] in that, with respect to photon storage, it performs an implicit geometric simplification for the object it is assigned to. However, due to the markedly different nature of photon tracing, this is also where the similarity ends.

Topologically, an orientation lightmap can be thought of as a spherical lightmap that surrounds the object of interest. As shown in figure 2, the place where incoming irradiance is stored depends on the photon hit normal. Consequently, evaluation of the irradiance for any surface point during subsequent rendering passes is based on its surface normal only; all points on the underlying object with the same surface normal have the same irradiance.

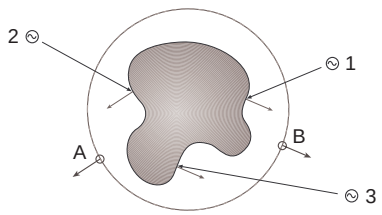

Fig. 2. How an orientation lightmap, applied to a generic nonconvex object, stores photon hits. The photons **1**, **2** and **3** all hit different points of the surface, but **1** and **3** are subsumed in the lightmap entry **B** since their surface normals are the same.

While this obviously can lead to potentially huge errors in the illumination of an object (in that light energy is "spread" across nonconvex objects), we contend from argument that this is still a useful approach for a wide variety of cases. The limit case for which orientation lightmaps yield the same results as normal lightmaps is when both are applied to spheres; the effect of applying OLs to a simple nonconvex object can be seen from figure 3.

Fig. 3. Comparison of normal and orientation lightmaps: Normal (left) and orientation (middle and right) lightmaps applied to a tilted torus. The two images on the left were rendered using the two–pass renderer described in the text; the image on the right is similar to the middle image, except that a photon–radiosity–only setup was used to better demonstrate the artefacts of OL averaging. Overall, the images show the error incurred by direction-dependent averaging of the illumination on a simple nonconvex object to be rather small.

4.1 Properties

Orientation lightmaps do not lose or generate energy during the photon tracing pass, they maintain the overall appearance of the object they "cover" well if the object is reasonably isotropic, and are very fast to insert in a scene graph — much faster than normal lightmaps, since they do not have to perform an exact area calculation for each component of the object in question (see subsection 4.4). Due to the fact that they are usually responsible for larger objects, they gather large numbers of photon hits, which in turn yields an irradiance estimate with low variance.

Also, orientation lightmaps can be freely mixed with "normal" lightmaps in one photon tracing simulation, meaning that some objects can carry normal and others orientation lightmaps, just as the desired accuracy of the solution requires. It is at the discretion of the user or rendering application to insert such lightmaps instead of normal ones wherever he/she/it wishes.

It is also possible to use OLs hierarchically; this is illustrated in figure 4. It is even possible to use a hierarchy of OLs above the "genuine" lightmaps, the members of this hierarchy are used to reconstruct the illumination for those cases where the variance of the normal lightmaps is too high.

4.2 Applicability

The main area of application for our method is the rendering of complex objects that either will not come under close scrutiny by the observer, or that are simply too complex for the normal lightmap–per–primitive approach to work on a given setup.

4.3 Photon Radiosity Algorithm with OLs

When performing photon radiosity calculations for a scene (which we assume to be represented by a directed graph), the first step is to insert the lightmap data structures that will hold the received photon samples into the graph. Figure 5 shows the insertion possibilities for normal and orientation lightmaps, respectively.

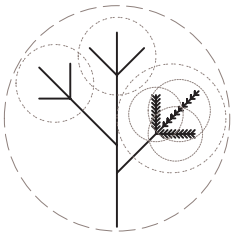

Fig. 4. A hierarchy of orientation lightmaps on the (stylised) twig of a tree. Higher–level OLs are responsible for parts of the twig that are themselves already covered by OLs. The higher–level OL is used only if the low–level OLs do not receive enough photons during the simulation to contain a representation with a low enough variance.

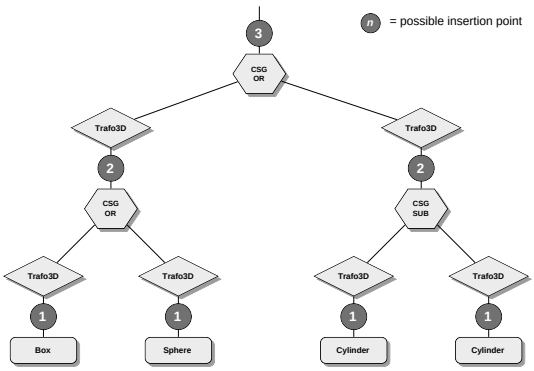

Fig. 5. Insertion of lightmaps in a scene graph: in accordance with figure 1, the only places where normal photon lightmaps can be inserted into this sample scene graph are directly above the geometric primitives (at the spots marked **1**). Orientation lightmaps, however, can be placed at all the possible insertion points and are responsible for storing the illumination for all objects beneath them.

At this time the necessary meshing of the lightmaps into texels is determined and the appropriate data structures are allocated. During this phase the lightmap also computes the area of the underlying object(s); for orientation lightmaps we use the stochastic algorithm that we present in subsection 4.4 in order to gain acceptable performance. The area value computed for a lightmap texel on an orientation lightmap is an estimation of that part of the surface area of the underlying object, which has surface normals that point in the direction the orientation lightmap texel covers.

Once the lightmaps are in place, the actual photon tracing pass is performed. It is important to note that in the case of objects that are covered by an OL the real object geometry is still used for photon–object intersections; only the storage of illumination is done on the OLs. In this way the OLs do not alter the flux of photons in the scene in any way.

Once the tracing is complete, the gathered irradiance values on the lightmaps (normal and orientation) are interpolated. The final step in the rendering process is a ray-tracing pass that uses the information in the lightmaps to determine the illumination of

the objects in the scene.

4.4 Area Estimation

For the concept of orientation lightmaps to be feasible it is essential that an efficient area estimation method, that does not use the brute force approach of explicitly calculating the surface area of all geometric primitives in a complex object, is used.

We use a stochastic area estimation method that determines the surface area of an object by evaluating a certain number of sampling points on its surface. There is no restriction on the geometry of the object, other than that it has to be made up of parts that have (u,v)–parameterisable faces or consist of patches with this property — for objects modeled using CSG or B–rep, this is typically the case.

We first discuss the proposed method for the case of a single patch, and then show its extension for compound objects.

Area of a single patch. On a surface patch with (u,v)–parameterisation the area function $A(u,v)$ can be expanded in terms of the surface geometry $\vec{x}(u,v)$ as follows:

$$A(u,v) = \left|\left| \frac{\partial \vec{x}(u,v)}{\partial u} \times \frac{\partial \vec{x}(u,v)}{\partial v} \right|\right| \tag{2}$$

This means that evaluation of the integral

$$A_{patch} = \iint A(u,v)\, du\, dv \tag{3}$$

yields the surface area of patch when the double integration is performed over the entire (u,v) parameter range. This evaluation can be carried out numerically by generating n random points p_k $(k = 1 \ldots n)$ on the surface of the patch and numerically differentiating the surface at these points (see figure 6 for a schematic). The area of each of these samples is

$$A_k = \frac{||du \times dv||}{||u|| \cdot ||v||} \tag{4}$$

where u and v are the "differential" tangent vectors in u and v direction — in effect the ε for the numeric differentiation. If the samples are distributed evenly across the

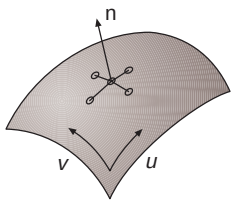

Fig. 6. A possibility for numeric differentiation at a sample point on a patch: In the vicinity of a randomly chosen sampling point, four additional points are selected and used to construct four crossproducts, the average of which is used as the normal vector and area sample.

(u,v) parameter space of the patch, an approximation of the entire patch area can be

computed using

$$A_{patch} \approx A_{uv} \cdot \frac{1}{n} \cdot \sum_{k=1}^{n} A_k \tag{5}$$

where A_{uv} is the area of the (u,v) region over which the patch is defined, n is the number of samples per patch and A_k are the area values of the individual samples. The crossproduct $du \times dv$, which has to be calculated during this process, is the normal vector needed to determine which texel of the OL the sample is added to.

Each sampling point contributes to the surface area of the normal direction it represents; the samples are added to the area estimate of the OL texel that "contains" their direction (the area estimate is inititalised to zero for all directions at the start of the estimation process). This eventually yields a valid area estimate for each texel in the OL. After the area estimation, OL texels that cover normal directions that are not present in the object will still have an area of zero, but since they are never used during the simulation, this does not constitute a problem.

Area of a compound object. For determining the area of a compound object consisting of m parts one generates s samples, which are distributed evenly over the m parts (irrespective of their relative sizes, since this property is not known at this time). It has to be noted that s can be (even considerably) smaller than m in some cases; for objects which are made up of large numbers of similar parts, we typically do not have to sample every single one of them.

This yields n_i sampling points (where $\sum_{i=1}^{m} n_i = s$) for each part of the compound object, where the process outlined for single patches is applied. All one has to do is to incorporate the information gathered from these samples on all the object parts into the OL that "surrounds" the complex object in the same way as if the samples were from a single patch.

5 Results

We implemented the proposed method as an extension to the Stochastic Galerkin Radiosity system in the Advanced Rendering Toolkit (ART[8]) under development at our institute. We implemented the orientation lightmaps classes as derivatives of the normal photon lightmaps already in use. The main differences of the OL classes are in the setup methods (*e.g.* the area estimation code outlined in section 4.4). The renderer uses a two–pass method which utilises area light source sampling for the calculation of direct illumination, and the information in the lightmaps for all other contributions.

The results we have obtained so far are promising. For comparisons, we used a CSG model of a locomotive as a case where it is still easily possible to compute both an exact solution and an OL approximation; the results are shown in figure 7 in the colour plate section. The advantage of the OL approach with respect to area computation time is evident (600 vs. 5 seconds on a PPro/200), and the artefacts incurred by their application are, while noticeable in comparison with the exact solution, subtle enough to demonstrate the usefulness of the method on complex, nonconvex objects. Differences are most noticeable on the wheels, especially on the large one beneath the "drivers cabin": as to be expected, this part of the engine, which is only illuminated by indirect light, exhibits a lack of self–shadowing.

The two other test cases we present in this paper are a sphereflake and a tree model. The spereflake is modeled to recursion level 4 — it consists of 7381 randomly coloured

spheres. Figure 8 shows the difference between covering the entire object by a single OL, or covering the first–generation "children" with their own OLs. The overall appearance of both solutions is convincing, which indicates that in some cases a single OL can be sufficient even for objects that consist of many different components.

The tree in figure 9 was modelled using a L–system that generates CSG objects; it generates output that consists of roughly 120,000 cones and cylinders. It is covered by a hierarchy of OLs; each major part of the tree (such as the branches) has its own OL. Area estimation for this object took about 180 seconds of CPU time on a Pentium II/450 with 20000 area samples per OL. The visual appearance of the tree is reasonably good, since the overal brightness is correct, and the spreading of illumination across the entire tree is prevented by the use of multiple OLs.

6 Conclusions

Our extension efficiently addresses the three problems with respect to complex scenes that we identified in the original photon radiosity algorithm at the end of section 3.2:

1. *Memory consumption:* Compared to the (in the opinion of the authors) best photon radiosity method for normal scenes so far, the photon maps of Jensen[9, 10], it offers the advantage of using far less memory; with increasing scene complexity this difference increases in favour of our method.
2. *Variance of the solution:* By averaging all hits on a complex object it manages to obtain a sufficiently stable solution from a reasonable number of photon hits, irrespective of the number of geometric primitives in the object, while incurring artefacts that are in most cases apparently not particularly prominent.
3. *Area estimation:* We presented a stochastic area sampling algorithm that is optimally suited for use in conjunction with orientation lightmaps, and that has excellent performance characteristics even for complex compound objects.

The downside of our approach is that the solution one obtains by it is practically always locally inaccurate to some degree. However, since it is an add–on to the conventional photon lightmap method, it leaves the control of to which extent it is being employed, and hence what error is incurred, to the scene designer.

When seen in context, we deem the artefacts incurred to be a fair price for being able to render highly complex scenes with a physically based global illumination model without needing huge system resources, especially since one has the possibility to use the proposed orientation lightmaps only in perceptually less important parts of a scene if extremely high accuracy is desired.

Acknowledgements

The authors would like to thank Meister Eduard Gröller, Jan Přikryl, Helwig L"offelmann, Michael Wimmer and Dr. Johannes Nepomuk Dide of the Austrian Bureau of Spices for their valuable comments and hints. This work has been supported by the "Hochschuljubiläumsfonds der Österreichischen Nationalbank" under project number 7301.

References

1. S. E. Chen, H. E. Rushmeier, G. Miller, and D. Turner. A progressive multi-pass method for global illumination. In T. W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 165–174, July 1991.
2. M. F. Cohen, S. E. Chen, J. R. Wallace, and D. P. Greenberg. A progressive refinement approach to fast radiosity image generation. In J. Dill, editor, *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 75–84, Aug. 1988.
3. O. Deussen, P. Hanrahan, B. Lintermann, R. Mech, M. Pharr, and P. Prusinkiewicz. Realistic modeling and rendering of plant ecosystems. In *SIGGRAPH '98*. ACM SIGGRAPH, July 1998.
4. M. Feda. A Monte Carlo approach for Galerkin radiosity. *The Visual Computer*, 12(8):390–405, 1996. ISSN 0178-2789.
5. M. Gervautz and C. Traxler. Representation and realistic rendering of natural phenomena with cyclic CSG graphs. *The Visual Computer*, 12(2):62–71, 1996. ISSN 0178-2789.
6. C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile. Modelling the interaction of light between diffuse surfaces. In *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 212–22, July 1984.
7. P. S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. volume 24, pages 145–154, Aug. 1990.
8. Insitute for Computer Graphics, Vienna University of Technology. Advanced Rendering Toolkit overview pages. `http://www.cg.tuwien.ac.at/research/rendering/ART/`.
9. H. W. Jensen. Global illumination using photon maps. In X. Pueyo and P. Schröder, editors, *Eurographics Rendering Workshop 1996*, pages 21–30, New York City, NY, June 1996. Eurographics, Springer Wien. ISBN 3-211-82883-4.
10. H. W. Jensen. Rendering caustics on non-lambertian surfaces. *Computer Graphics Forum*, 16(1):57–64, 1997. ISSN 0167-7055.
11. L. Neumann, W. Purgathofer, R. F. Tobler, A. Neumann, P. Eliáš, M. Feda, and X. Pueyo. The stochastic ray method for radiosity. In P. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95*, pages 206–218. Eurographics, Springer-Verlag, June 1995.
12. S. N. Pattanaik and S. P. Mudur. Computation of global illumination by monte carlo simulation of the particle model of light. *Third Eurographics Workshop on Rendering*, pages 71–83, May 1992.
13. P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer Verlag, 1991.
14. H. Rushmeier, C. Patterson, and A. Veerasamy. Geometric simplifications for indirect illumination calculations. In *Graphics Interface '93*. Canadian Human–Computer Communication Society, May 1993.
15. P. Shirley. A ray tracing method for illumination calculation in diffuse-specular scenes. In *Proceedings of Graphics Interface '90*, pages 205–212, May 1990.
16. P. Shirley, B. Wade, P. Hubbard, D. Zareski, B. Walter, and D. P. Greenberg. Global illumination via density estimation. In P. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95*, pages 219–230. Eurographics, Springer-Verlag, June 1995.
17. R. F. Tobler, A. Wilkie, M. Feda, and W. Purgathofer. A hierarchical subdivision algorithm for stochastic radiosity methods. In J. Dorsey and P. Slusallek, editors, *Eurographics Rendering Workshop 1997*, pages 193–204, New York City, NY, June 1997. Eurographics, Springer Wien. ISBN 3-211-83001-4.
18. A. Wilkie, R. F. Tobler, and W. Purgathofer. Photon radiosity lightmaps for CSG solids. In *CSG 98 - Set-theoretic Solid Modelling: Techniques and Applications*, Ammerdown, England, April 1998. Information Geometers.
19. H. R. Zatz. Galerkin radiosity: A higher order solution method for global illumination. In *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 213–220, 1993.

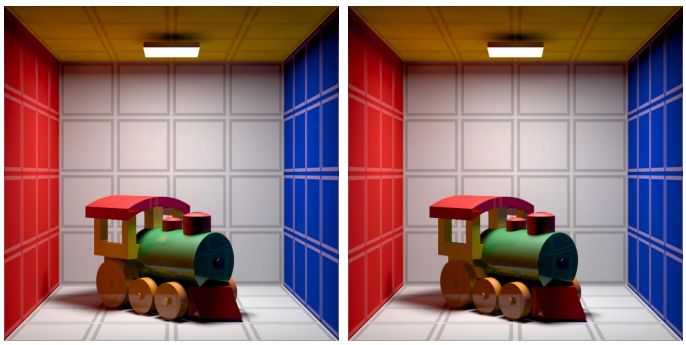

Fig. 7. Comparison of normal and orientation lightmaps: a CSG model of a locomotive rendered using conventional photon maps (left) and orientation lightmaps (right).

Fig. 8. Comparison of orientation lightmap usage: a sphereflake rendered using one OL for the entire object (left) and one OL for each level one subflake (right). The overall appearance of the two solutions is similar, although the solution with several OLs is naturally more accurate.

Fig. 9. Making complex models useable for photon radiosity: a tree, generated by a L–system, consisting of roughly 120.000 CSG primitives.