Parallel Simulation of the Global Illumination

Olaf Schmidt, Jan Rathert,Ludger Reeker
University of Paderborn
Department of Computer Science
33102 Paderborn, Germany

Tomáš Plachetka
Comenius University
Department of Computer Science
84215 Bratislava, Slovakia

Abstract *In this paper, a rendering system is presented, which utilizes efficient parallel methods for solving the global illumination problem. The system allows interactive walkthroughs in complex architectural environments consisting of several thousands of polygons and an off-line animation production. The image synthesis system is designed as a portable and scalable client-server architecture. The parallel global illumination calculations are performed by a rendering server, either on massively parallel computer systems or on heterogeneous workstation-clusters. The visualization-interfaces are connected to the server as clients, which are started on workstations with specialized graphics hardware.*

Keywords: Parallel computing, image synthesis, radiosity, ray tracing, two pass method, global illumination

1 Introduction

Computer graphics technology is quickly becoming part of everyday experience. One important aim of the research in the area of image synthesis in the last couple of years has been the recreation of images of environments that do not exist up to now. The ability to create images of non existing environments is important to applications ranging from industrial or architectural design to advertising and entertainment. In particular the rendering of realistic images requires a precise treatment of lighting effects by simulating the underlying physical phenomena of light emission, propagation, and reflection. The most advanced algorithms for such simulations of radiative light transfer are radiosity methods [1] and Monte Carlo ray tracing. The visual realism provided by global illumination algorithms supports the ability to perceive the content of a 3-dimensional environment by simulating illumination effects such as indirect illumination, color bleeding between surfaces and soft shadowing. Drastic progress has been made in terms of 3D-animation, software modeling and shading capabilities. Ray tracing and radiosity algorithms, currently implemented in image synthesis systems, provide the necessary rendering quality, but these methods are suffering from their extensive computational costs, and their enormous need for memory. The use of scalable parallel systems offers a solution to these problems.

In this paper an image synthesis system is presented, which utilizes efficient parallel methods for solving the global illumination problem. These methods are introduced in section 2 and section 3. Section 4 provides a brief description of the rendering system. Performance measurements and conclusions are presented in section 5.

2 Data-Parallel Radiosity

Radiosity has become a popular technique in image synthesis to produce high quality images taking into account the correct global illumination of an environment. This method was originally developed in the area of heat transfer, and simulates the exchanges of heat energy between the objects in an environment.

In 1984 Goral et. al. [1] introduced the radiosity method for computer graphics in order to calculate the global illumination in environments consisting of perfectly diffuse reflecting surfaces. It is assumed that the environment is divided into discrete patches. The radiosity (radiant energy that leaves a surface) of a patch is determined by the self emitted radiosity plus the reflected radiosity, and is described by the radiosity equation. Such an equation exists for each patch of the environment. This system of simultaneous equations can be solved by using an iterative Gauss-Seidel method, which assigns a particular radiosity value to each patch, and therefore, a discrete representation of the diffuse illumination in the scene is calculated. The obtained solution is independent of the viewer's position in the scene. This means that the position of an observer can be modified without recalculating the radiosity solution. The full matrix solution is very time and memory consuming, and is impractical for larger numbers of patches.

To overcome the limitations of the full matrix solution, Cohen et. al. introduced the progressive refinement method [2]. The light energy is distributed in the environment in an iterative process. In every iteration the patch with the most unshot radiosity is determined and this amount of energy is distributed to all other patches in the environment. This process is repeated until the unshot radiosity of the selected patch drops below a predefined value. While the solution is progressively refined, the intermediate results can be displayed after every iteration.

A form factor specifies the fraction of energy leaving one patch and impinging on another patch of the environment. This fraction depends on the visibility between the two patches, and their orientation in 3D space. It is difficult to calculate the form factors analytically for general applications and complex geometries. Therefore, a numerical technique called hemicube approach was introduced to compute form factors for complex environments [3]. Other strategies for calculating proper form factors have been proposed [4] [5], which produce better form factors than the hemicube approach.

A method of speeding up the radiosity computations is to use some sort of parallelism. Our rendering system uses a data-parallel method for computing a radiosity solution. This method is based on partitioning the geometry of the input scene into smaller subscenes and distributing them onto several processors of a scalable MPP-system [6] [7] [8]. In a preprocessing step, an appropriate meshing of the input scene is calculated, and the resulting mesh of patches is divided into 3-dimensional cells, every cell containing a part of the scene (subscene). The processes of meshing and partitioning are closely related. The scene is partitioned into convex regions (boxes) with each patch being assigned to exactly one cell due to the geometrical position of the patch. A patch lies completely within a cell. This restriction makes it difficult to subdivide an arbitrarily meshed input scene into convex regions. A solution to this problem is to clip overlapping patches at the cell boundaries, which leads to meshes of various complexity for different cell topologies. In our solution, a 2d-grid placed above the input scene (parallel to xy-plane) is projected onto every surface. The meshing process uses the projected edges of the grid as patch boundaries. When the scene is partitioned into convex regions, cell boundaries can only be placed along edges of the 2d-grid. This ensures that a patch lies exactly within one cell. Obviously, this solution does not lead to an optimal meshing of the surfaces, and cells are only produced along two dimensions. The advantage of this solution is that the same mesh can be used for parallel radiosity calculations with different cell topologies, and so the calculation times are comparable. Note that this solution can easily be extended to cell topologies in three dimensions. This is possible by the use of additional 2d-grids parallel to the xy-, xz- and yz-plane during the meshing and partitioning process. This preprocessing step performs static load balancing by defining subscenes with nearly the same number of patches within every cell. Af-

ter this preprocessing step, the cell topology is mapped onto the given processor topology. In every cell the iterations of the progressive refinement algorithm [2] are performed asynchronously. Our method distinguishes between a local shooting iteration with the shooting patch being part of the local scene, and an external shooting iteration with the shooting patch not belonging to the local scene. During a local shooting iteration a shooting patch of the local subscene is being selected. The form factors to the local patches are calculated using the method introduced by Baum et. al. [4]. This method uses the hemicube [3] only to determine the visibility between the shooting patch and sample points on the other patches of the environment. Form factors are evaluated analytically using the visibility information stored within the hemicube pixels. All the patches of the subscene of the shooting patch are projected onto the hemicube, and the energy is transfered to the local environment according to the calculated form factors. As a next step, all the subscenes are determined which include patches that may receive unshot radiosity of the selected shooting patch (see Fig. 1). The form factors to

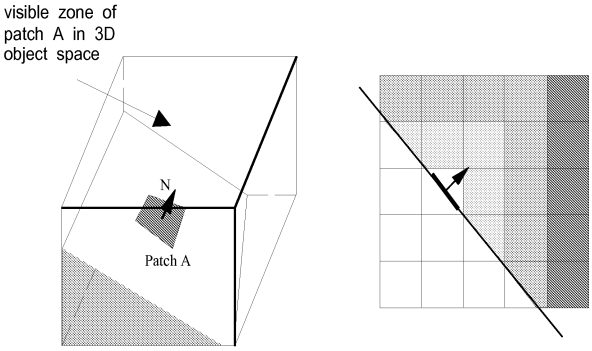

Figure 1: The visible zone of a shooting patch

the patches of these subscenes have to be computed by projecting all patches onto the same hemicube placed above the selected shooting patch. Due to the distributed storage of the scene either parts of the scene database or visibility information have to be exchanged between the processors in order to calculate form factors correctly. In our parallel approach, visibility information, and a copy of the shooting patch are packed into the hemicube message, which is passed to the cells in the visible zone of the shooting patch in a certain order. After receiving a hemicube message, an external shooting iteration is performed taking into account the included information. Visibility information is encoded in a so called hemicube bitmap. One bit of the hemicube bitmap corresponds to a pixel of a hemicube face, and is set to 1 if a patch has been projected onto that pixel of the hemicube (see Fig. 2). If

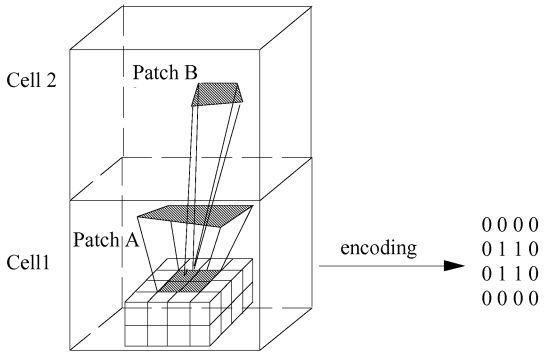

Figure 2: The hemicube bitmap

a hemicube bitmap is sent to a neighboring cell, projections are only performed on those pixels of the hemicube with the corresponding bitmap entry set to 0. It is ensured by the routing of the hemicube bitmaps that all the hemicube pixels for which the bitmap entry is set to 1 are covered by the projection of a patch that is closer to the shooting patch than all the patches in the actual cell [7]. Vice versa, if the bitmap entry is set to 0 there is no patch closer to the shooting patch than all the patches of the actual cell, that was projected onto the corresponding hemicube pixel in another cell. At the beginning of each iteration, it is tested if the maximum unshot radiosity of the received external shooting patch (stored in the external shooting patch buffer) is larger than the unshot radiosity of the locally selected shooting patch. In this case, an external shooting iteration is performed. During the external iteration, in a first step, the patches of the local scene are projected onto the hemicube, which is placed above the external shooting patch. Projections

are only allowed on pixels for which the corresponding bitmap entry is 0. The hemicube bitmap is updated while the projections are being performed. The updated bitmap, and the shooting patch are packed into a hemicube message and sent to the directly neighboring cells. In a second step, the determined visibility information is used to calculate form factors between the external shooting patch and the local patches in the same manner as during a local shooting iteration. The unshot radiosity is distributed in the local cell according to the calculated form factors.

The 2-dimensional example in Fig. 3 shows that sending the hemicube bitmap to neighboring cells in the wrong order will lead to projection errors. If the hemicube of shooting patch

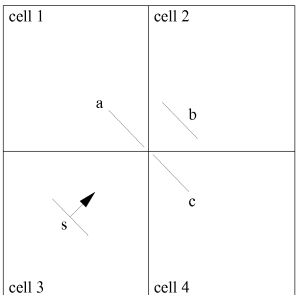

Figure 3: Projection errors due to false routing of messages

s is sent from cell 3 via cell 1 and 2 to cell 4, patch b is projected onto the hemicube before patch c. This leads to an error during the following shooting iteration, since patch b is covered by patches a and c. A correct routing is achieved by sending the hemicube bitmap from cell 3 to the directly neighboring cells 1 and 4. The local scenes of cell 1 and cell 4 are projected independently onto the hemicube. A valid hemicube bitmap for cell 2 is calculated by performing a logical OR operation on the corresponding bitmap entries of the bitmaps created in cell 1 and cell 4. The resulting bitmap is used to perform correct projections within cell 2. An example for the routing of a hemicube bitmap in a 2d-cell topology is given in Figure 4. This routing scheme can easily be extended for 3d-cell topologies. In this

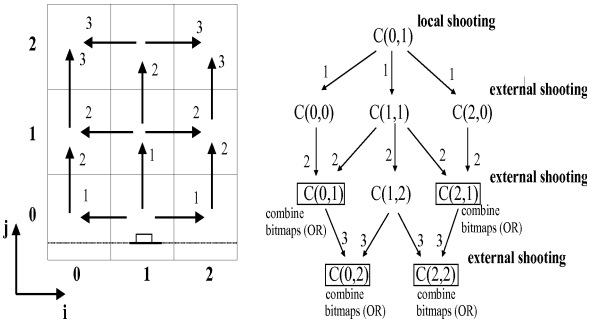

Figure 4: Routing of a hemicube bitmap

case, situations can occur with a maximum of 3 bitmaps having to be combined in order to get a valid bitmap (in 2d the maximum is 2). At the beginning of the parallel calculations, only those processors related to subscenes containing primary light sources are able to perform local shooting iterations. All other processors are waiting until they receive a hemicube message with an external shooting patch. After performing an external shooting iteration, energy is transferred into these subscenes. Now these processors can also start their progressive refinement iteration with interleaved local and external shooting iterations. After a short startup phase, all processors are performing their iterations asynchronously.

It is important to notice that only the hemicube bitmaps (one bit per hemicube pixel) are passed between the processors in order to perform distributed visibility calculations. Because of the described routing it is not necessary to send z-buffer information (depth values). So the communication overhead of our method is very low compared to previously introduced data-parallel radiosity methods [9]. The idea of hemicube bitmaps was introduced in [7] for the first time. A similar concept named *visibility masks* was introduced by Arnaldi et. al. [10]. They are using ray casting for the form factor calculation. The visibility mask is a sub-sampled hemisphere identical with the one involved in the computation of form factor calculation. Every pixel of the hemisphere, that is used for the form factor

computation, corresponds to a boolean value in the visibility mask. This visibility masks are exchanged between different processors in order to calculate correct form factors. The routing of messages is also very similar to the one described in [7] [8].

2.1 Radiosity for Dynamic Scenes

For common modeling systems the paradigm *modeling then rendering* is valid. In these systems the processes of modeling and rendering are strictly separated. The image synthesis system introduced in this paper, tries to change this paradigm to *modeling while rendering* as proposed by Chen [11]. The user is allowed to interact with the objects of the environment, e.g. turn on/off light sources, move objects, change materials, and after each interaction the primary influences of the modification of the scene on the global illumination are displayed with short delay. The integrated data-parallel radiosity method was modified in order to calculate radiosity solutions in dynamic environments. After a user interaction the change of geometry is broadcasted to all processors. Incremental changes of affected radiosity values are calculated instead of recalculating a complete radiosity solution [11]. In most of the cases, a scene modification has only local influences on the illumination in a given environment (especially in large buildings with several rooms). Thus, in our data-parallel radiosity method, incremental radiosity values only have to be calculated in subscenes of the partitioned input scene which are affected by the interaction. Although this initially causes more load on some processors, other processors receive work due to propagation of light energy. During an usual modeling session, several changes are made. Every processor has a local geometry-queue, where the changes, which affect the local scene, are accumulated [11]. The queues contain the history of all the geometry changes which, when executed in order, will describe the up-to date scene. It is very likely that the successive modifications affect different subscenes, which are handled by different processors. Thus, a good load distribution in produced by the changes.

3 Parallel Two Pass-Method

A parallel ray tracing method has been integrated into the image synthesis system. In the generic ray tracing algorithm, the ambient term in the local illumination model is replaced by a constant. This is not correct, but usually works well (the output seems to be realistic). In order to obtain more realistic results, radiosity and ray tracing algorithms can be combined to a two pass method. This idea is not new, see e.g. [12] for a reference. In the first pass of our implementation, the data-parallel radiosity algorithm is started in order to simulate the diffuse reflection of light. The second pass is running parallel ray tracing on the same scene, using the radiosity information from the first pass. The constant ambient term is replaced by a radiosity value obtained from the first pass. At the moment, the scene description is stored on every processor of the parallel system during the second pass. This means that the ray tracing pass is not data-parallel, and so the size of the scenes, which can be handled, is limited. For parallel ray tracing, a *demand-driven* strategy for dynamic load balancing based on an adaptive image space partitioning is used. Initially, equally sized regions of a part of the total image space are assigned to the processors. As soon as one processor finishes his work, he sends a request for new work to a central load balancing process. In Figure 5, the pseudo-code for the load balancer is given. In this pseudo-code, C is the number of screen columns, and N the number of workers. A single screen column is an atomic job. Let us assume, that for any two different jobs of equal sizes

$$\frac{time\ needed\ for\ computation\ of\ job_1}{time\ needed\ for\ computation\ of\ job_2} \leq T$$

with $T \geq 1.0$ being a constant. Unfortunately, in practice, we usually do not know the parameter T beforehand. Usually, it must be set

```
WorkLoad=C;
Ratio = 1/(1+T(N-1));
JobSize = (WorkLoad*Ratio);
....assign a job of size JobSize to each worker;
WorkLoad = WorkLoad - N*JobSize,
while (WorkLoad > 0 )
{
  ...wait until an idle worker asks for a job;
  JobSize = WorkLoad*Ratio;
  ... assign the worker a job of Size JobSize;
  WorkLoad = WorkLoad - JobSize;
}
```

Figure 5: Pseudo-Code of the load balancer

empirically. The size of the new workpackage (region of image space) is adaptively decreased depending on the number of processors and the work that is still to be done. This strategy ensures that the idle times of the processors are minimal at the end of the calculation of one picture. The communication overhead of this strategy is low, compared to a demand driven strategy with an atomic sized job (small box or column) being assigned to each request. Thus, a nearly linear speedup can be achieved for a large number of processors.

4 Image Synthesis System

The methods introduced in section 2 and section 3 have been implemented and integrated into an interactive image synthesis system. The system is developed as an open, modular client-server architecture. It allows easy extensions of the system with additional components of the application. These components are either new global illumination methods on the side of the server or application interfaces as clients. The integration and parallelization of additional applications is made easier by the provision of different heuristics for mapping, mechanisms for dynamic load balancing, monitoring, and communication mechanisms (Fig.6).

The architecture distinguishes between clients that run on one or more workstations within a network (frontend components), and the rendering server, that runs on a parallel machine (backend). The rendering server processes the instructions from clients (illumination calculations), and returns the results to the corresponding visualization clients. The

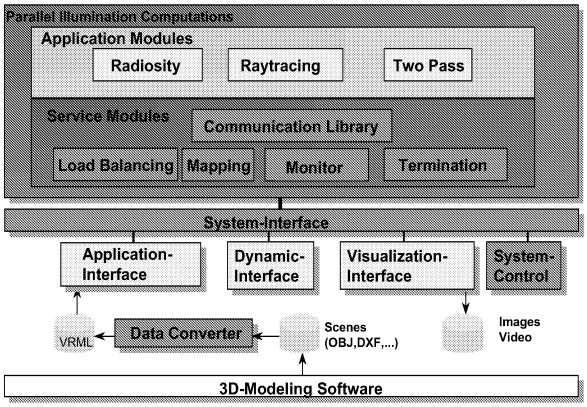

Figure 6: Architecture of the system

visualization-, meshing-, interaction-interfaces and the system control are started as clients on frontend machines. The system-control allows the user to choose between the different global illumination methods of the rendering server. From there the corresponding application components on the frontend system are started, and the rendering server is initialized to run the selected illumination method. Scene descriptions can be converted from standard file formats into the internal file format of the rendering system. Next, this internal format is read by an application interface of the image synthesis system. The backend performs a parallel simulation of the light distribution for the scene description. The image synthesis system is a complete, integrated solution for the efficient visualization of architectural data. It is therefore possible to simulate global illumination effects caused by specular and diffuse inter-object reflection. Radiosity calculations are independent of the viewpoint. Thus, the results of the simulation can be used for an interactive walkthrough [3]. During the walkthrough, parallel radiosity calculations continue and, on demand, the actual results are visualized. With this procedure the quality of the visualization permanently improves (progressive refinement method) [2]. At the end of

the radiosity calculations, it is possible to define a camera path to produce an animation with the help of the two-pass extension of the radiosity method. Because of the use of parallel computer systems with distributed memory and efficient parallel global illumination methods, the performance of the system is scalable. Thus, an image synthesis based on the simulation of the correct lighting conditions is also possible in complex architectural models.

5 Results

In this section, performance measurements of the implementation of the new data-parallel radiosity method, and the two pass extension are are presented. These measurements were performed on the Parsytec CC (peak performance of 12.7 GFLOPS). Each node of this MPP-system is a Power PC 604 (133 MHz, 64Mbyte main memory).
The basis for the measurements presented in this section is a highly complex input scene. This scene represents the Rosenthaler Hof at

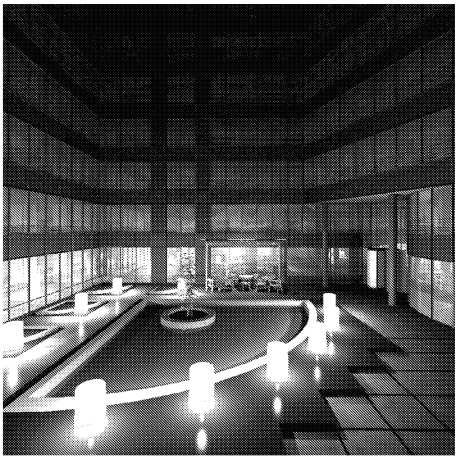

Figure 7: Inside of Rosenthaler Hof

the center of Berlin. It consists of the former department store Wertheim, which is being remodeled to a modern building with apartments, offices and stores. Figure 7 shows a two-pass visualization of the roofed courtyard of the office building at night. This test scene had been substructured into 145588 patches before the parallel radiosity calculations were started. During the calculations, adaptive mesh refinement is used to produce more exact illumination effects.

Rosenthaler-Hof scene number of patches: 145588 (eps.: 5e-4)			
proc.	time (sec)	speedup	efficency
1	8236	1.00	1.00
2	4417	1.86	0.93
4	2368	3.48	0.87
8	1209	6.81	0.85
16	658	12.51	0.78
24	470	17.52	0.73
32	422	19.50	0.61
36	405	20.32	0.56

Table 1: Time for radiosity computations

In table 1 the parallel calculation times which were needed for the radiosity-pass, are presented. The radiosity calculations were performed until the unshot radiosity of the selected shooting patches dropped below a predefined percentage of the maximum energy emitted initially by primary light sources. This table shows an improvement in the calculation time with up to 36 processors. With more than 24 processors, no significant reduction of the calculation time was obtained. The reason for this is an unevenly and dynamically produced load, which is caused by the spatial subdivision. Because of the different groupings, and the different material properties of the objects inside the partitions, the amount of light energy which is distributed and received in the local environments differs. The static load balancing strategy, which produces the spatial subdivision of the input scene, does not accurately predict these amounts of energy at the beginning. Thus, there are some subscenes receiving a large amount of energy while others receive much less energy. Processors that are responsible for distributing energy in dark regions are very soon becoming idle, and so the speedup decreases. This aspect has to be accounted for by dynamic load balancing, which is currently in its implementation stage. Table 2 shows the calculation times for the ray tracing pass, which were needed to produce figure 7.

Rosenthaler-Hof scene Number of Light-sources: 12				
	Boxes		Adaptive	
proc.	time (sec)	speedup	time (sec)	speedup
1	1768	1.00	1768	1.00
2	934	1.89	886	2.00
6	364	4.89	298	5.93
14	174	10.16	129	13.71
30	92	19.22	62	28.52
38	73	24.22	49	36.08

Table 2: Time for ray tracing pass

The calculation times of our dynamic load balancing strategy (adaptive) are compared with a demand driven strategy with the image partitions being equally sized boxes. The measurements show that the new strategy is superior to the classical one.

6 Conclusion

We presented an image synthesis system, which uses efficient parallel methods for the simulation of the global illumination. A data-parallel radiosity method and a parallel ray tracing method are combined in order to calculate two pass visualizations of complex environments. Future work has to concentrate on the aspect of dynamic load balancing for the parallel radiosity method and data-parallel ray tracing calculations.

References

[1] Goral C.M., Torrance D.E., Greenberg D.P., and Battaile G. Modeling the interaction of light between diffuse surfaces. In *Computer Graphics*, volume 18, pages 213–222, 1984.

[2] Cohen M.F., Chen S.E., Wallace J.R., and Greenberg D.P. A progressive refinement approach to fast radiosity image generation. In *Computer Graphics*, volume 22, pages 75–84, 1988.

[3] Cohen M.F. and Greenberg D.P. The hemi-cube: A radiosity solution for complex environments. In *Computer Graphics*, volume 19, pages 31–40, 1985.

[4] Baum D.R., Rushmeier, H.E., and Winget J.M. Improving radiosity solutions through the use of analytically determined form factors. In *Computer Graphics*, volume 23, pages 325–334, 1989.

[5] Wallace J.R., Elmquist K.A., and Haines E.A. A ray tracing algorithm for progressive radiosity. In *Computer Graphics*, volume 23, pages 315–324, 1989.

[6] Arnaldi B. and Pueyo X. Environments by virtual walls for radiosity computations. In *Proc. of the 2nd Eurographics Workshop on Rendering*, volume 29, pages 4373–4376, 1991.

[7] Schmidt O. *Verteilte Energiesimulation in geschlossenen Räumen.* Diploma thesis, University of Paderborn, 1994.

[8] Menzel K., Schmidt O., and Stangenberg F. Distributed rendering techniques using virtual walls. In *EuroPVM 94*, 1994.

[9] Chalmers A.G. and Paddon D.J. Parallel processing of progressive refinement radiosity methods. In *Proc. of the 2nd Eurographics Workshop on Rendering*, 1991.

[10] Arnaldi B., Priol T., Renambot L., and Pueyo X. Visibility masks for solving complex radiosity computations on multiprocessors. In *Proc. First Eurographics Workshop on Parallel Graphics and Visualization*, pages 219–232, 1996.

[11] Chen S.E. Incremental radiosity: An extension of progressive refinement radiosity to an interactive image synthesis system. In *Computer Graphics*, volume 24, pages 135–144, 1990.

[12] Sillion F. and Puech C. A general two-pass method integrating specular and diffuse reflection. In *Computer Graphics*, volume 23, pages 335–344, 1989.