ADDING REALTIME SHADOW FEEDBACK TO GLOBALLY ILLUMINATED INTERACTIVE VIRTUAL ENVIRONMENTS

Frank Schöffel

Fraunhofer Institute for Computer Graphics
Rundeturmstraße 6, 64283 Darmstadt, Germany
schoeffe@igd.fhg.de

ABSTRACT

In this paper, we introduce a new method for providing realtime shadow feedback in interactive environments with global illumination simulated by a radiosity process. Several approaches for updating global illumination in dynamic environments have been presented in recent years, but none of these can produce true realtime feedback for moderately complex scenes. The realtime shadows presented here are created by manipulating the vertex colors resulting from a previous radiosity calculation. The notion of *multipoints* is introduced, that allow significant speed-up of the shading process. This method works on the same scene representation as the radiosity simulation does. We show how the multipoint shading method can be integrated into a Virtual Reality system, in combination with a radiosity update process. Thus, our system provides realtime shadows during user interaction and still enables correct soft radiosity shadows to be updated after the interaction has been finished. The visual appearance of the realtime shadows is similar to the radiosity shadows, and its visual quality can easily be controlled by the user, which makes the method adjustable and thus applicable to complex environments.

Keywords: Shadow update, realtime feedback, interaction, radiosity, multipoints.

1 INTRODUCTION

In many interactive 3D applications, particularly in most Virtual Reality (VR) applications, the realistic rendering of the virtual environment is an important demand. Realistic shadows and lighting effects contribute significantly to the visual realism of a scene [Asthe94]. Moreover, shadows provide important visual cues about the spatial relationships in the virtual environment and, thus, support user interactions in 3D space [Slate95][Wange92a]. Therefore, global illumination effects are often simulated in a pre-process, e.g., using the radiosity method [Goral84], and are then added to the scene either as vertex colors or shadow textures [Asthe95][Mölle96][Myszk95].

Because of the view independence of the radiosity simulation, this technique is very well suited for walk-through applications, but it is restricted to static environments: Once the geometry is changed, the radiosity simulation has to be repeated in order to generate a correct illumination—which is much too slow for interactive applications. Several approaches have been presented for updating the radiosity solution in dynamic environments more efficiently [Chen90][Georg90][Mülle95][Drett97]. However, all these methods still fail to provide realtime shadow feedback in scenes of moderate complexity. Thus, current VR systems which support realistic soft shadows are restricted to either walk-through applications where no interaction except camera movement is allowed, or the shadows are updated with significant delay *after* scene modifications, if at all. Realtime shadow feedback *during* user interaction, e.g., when the user moves an object, is not possible in these systems.

On the other hand, shadow feedback during user interaction is highly desirable, since it helps the user to perform interactive tasks like, e.g., object positioning, faster and more exact. Such a shadow feedback has to be very fast, but not necessarily physically correct [Wange92b]. In order to keep the visual impression of the scene as realistic as possible, however, a certain degree of realism should still be provided by the fast shadow feedback. A shadow technique that is suitable for providing realtime shadow feedback during user interaction, in a scene with realistic soft radiosity shadows, should therefore meet the following requirements:

- Shadow update has to be very fast.
- The fast shadow should appear as realistic as possible. In the context of a radiosity-illuminated scene, this means its visual appearance should be close to that of the radiosity shadows.
- Existing radiosity results must not be altered by the realtime shadow, and the geometry of the scene should not be modified, in order to allow for efficient radiosity update procedures based on the previous radiosity solution to work properly.

Due to the contradictory nature of the first two requirements, a compromise has to be found between shadow quality and shadow update time, and the best solution should be a method that is easily adjustable by the user to his specific needs.

In the next section of this paper, we discuss previous approaches to fast shadow generation in the given context, before Section 3 presents the multipoint shading technique that meets the above mentioned goals. Its integration into a VR system in combination with a radiosity process is explained in Section 4. Results are reported for several test scenarios in Section 5 and, finally, conclusions are drawn and a future outlook is given in Section 6.

2 RELATED WORK

Shadow generation has been an important research topic for many years now, resulting in a broad variety of shadow algorithms with very different approaches. Some of these methods operate in image space, and many approaches concentrate on the exact identification of shadow boundaries and discontinuities in order to create high-quality shadows. However, it is obvious that in the context of fast, interactive shadow generation within scenes of ‘real world’ complexity, we cannot afford the time for *exact* shadow generation. Therefore, the following review concentrates on the approaches allowing for shadow generation at interactive rates. We do not consider approaches like, for instance, the BSP-tree shadow volume methods [Chin89], which are not applicable for fast shadow generation in interactive scenes of significant complexity. A more comprehensive survey on shadow algorithms can be found, among others, in [Woo90].

A common method for shadow generating is to apply multi-pass methods. These techniques first render the scene as seen from the light sources, extract the occlusion information to generate depth buffers or shadow maps, and then apply these in a final image rendering pass. Segal *et al.* showed how graphics hardware can be exploited to efficiently perform these tasks by applying texture mapping techniques [Segal92]. Using the accumulation buffer [Haebe90], soft shadows can be generated by averaging shadow maps for slightly different light source positions [Heckb96]. Diefenbach and Badler combined shadow volumes using stencil buffers [Diefe97]. The forward shadow mapping method [Zhang98] reverses traditional shadow map algorithms by using image warping techniques. While all these methods can provide shadows quite fast, most of them require special hardware features, and they all work very differently from radiosity approaches, which makes an integration of both approaches into one system difficult.

Another method for soft shadow generation is to convolute images of extended light sources and the receiver [Soler98a]. This convolution operation does not require special hardware, and the resulting images can be used as shadow maps. This technique fits smoothly into a radiosity system [Soler98b], but unfortunately cannot achieve shadow update rates that are sufficient for realtime shadow feedback.

3 REALTIME SHADOW GENERATION USING SHADED MULTIPOINTS

In this section, a new realtime shadow generation method is presented that operates on the same data structures as a radiosity process. However, it determines shadows less accurately and is therefore much faster. We assume that a radiosity simulation has already been performed on a given mesh, and that the patch subdivision remains fixed throughout the session. We use vertex colors to represent the illuminance distribution and establish the fast shadows by darkening existing vertex colors—we neither add shadow polygons to the scene nor modify the scene’s geometry to represent the shadows.

3.1 Overview

Our shadow generation method is designed for being integrated into an interactive system in which a user can select and move or manipulate objects. The steps performed during user interaction can be summarized as follows:

- When an object O_m is selected, the direct radiosity shadows that O_m has cast onto other objects are removed.
- The resulting colors $c'(v_i)$ of all patch vertices v_i are backed up for later re-use (see Sec. 3.4).
- Then, the realtime shadow is added by identifying the vertices in shadow and darkening the corresponding vertex colors. This shadow has to be updated whenever the user moves the object further or manipulates its geometry (see Sec. 3.2 – 3.4).
- Finally, when the user releases the object, radiosity shadow update is triggered. The resulting radiosity vertex colors simply override the vertex colors resulting from the last realtime shadow, thus replacing it by the radiosity shadow (see Sec. 3.5).

The steps for generating the realtime shadow are explained in more detail below.

3.2 Identifying Shadow Regions

For generating the shadow of O_m, we first have to identify the regions affected. We use shadow volumes for this task, constructing a shaft from each light source (area light sources are reduced to their patches' midpoints) over the bounding box of O_m, and checking the scene objects against it. When a bounding box hierarchy is available, this test can be performed quite fast. The result is a list of candidate objects, for which each vertex has to be checked whether it is in shadow or not (see Fig. 1). This shadow test is established by a simple ray-intersection test only against O_m, not against the whole scene.

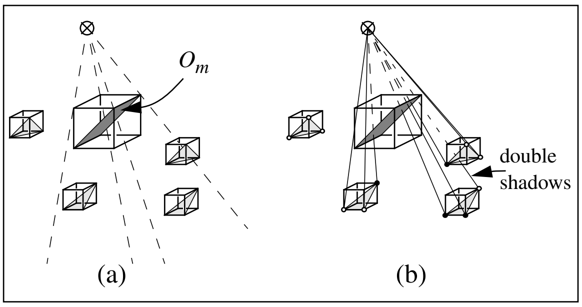


Figure 1: Identifying vertices in shadow by (a) shadow volumes and (b) ray casting for candidate objects, with elimination of 'double shadows'.

Vertices thus identified as being inside the shadow region might already lie within the shadow of another, static object. These vertices should not be darkened again. We therefore should identify these 'double shadows' by casting another ray towards the light source—if it hits any patch not belonging to O_m, the vertex is already located in a shadow region and should not be darkened again. However, we found that this operation is rather expensive and can often be avoided in practice, since errors caused by 'double shadows' are usually difficult to recognize in the image.

The result of this procedure is a list of vertices that are to be darkened according to O_m's new shadow. Repeating this process for all light sources, we obtain lists of shaded vertices with respect to each light source.

3.3 Shading Vertices

Once we have thus identified the patch vertices in O_m's new shadow, we darken their vertex colors by some amount. We perform this operation directly on the *RGB* values of the vertex colors. In order to save calculation time, we do not take the light source distance, color, or intensity into consideration. All light sources are treated equally, allowing for a very simple equation to be applied for darkening the color of a vertex v^l in shadow:

$$c(v^l) \leftarrow c(v^l) - \frac{c(v^l) - Amb}{m} \tag{1}$$

Thus, darkening the current vertex color depends only on the total number of light sources in the scene (m) and on an ambient value *Amb*, which represents a minimum brightness value for shaded vertices. Eq. 1 is applied once for each light source. Since vertices may be in shadow with respect to more than one light source, they may get shaded several times. This results in different levels of shading, providing the visual impression of penumbras and approaching the appearance of radiosity shadows.

The Ambient Value

Amb represents the amount of indirect illumination at a given point. Its value can be either specified by the user or estimated by evaluating the indirect portion of the illumination at all vertices in the scene. The indirect portion of the radiosity at vertex v is given by the following equation, where $B_p(v)$ is the radiosity due to the light source p, and $B(v)$ is the total radiosity at v:

$$B_{indir}(v) = 1 - \frac{\sum_{p=1}^{m} B_p(v)}{B(v)} \tag{2}$$

Averaging over all n vertices in the scene results in the following approximation for *Amb*:

$$Amb \approx \frac{1}{n} \cdot \sum_{i=1}^{n} B_{indir}(v_i) \tag{3}$$

Computing this value separately for the *Red*, *Green,* and *Blue* channel and averaging again, we obtain one achromatic value for *Amb,* which we apply in Eq. 1. Estimating *Amb* as described above is quite costly, but fortunately it has to be performed only once at initialization time. Nevertheless, we found that a user-defined, fixed value for *Amb* often suffices for good visual results. Usually a value of ~2 % of the brightest displayable color is a good choice for *Amb.*

Increasing Accuracy

Obviously, Eq. 1 is only a rough estimation. If one wants to achieve more exact results, one might take into consideration, for example,

- the distance $d_{v^l, l}$ between light source l and vertex v^l,
- the intensity I_l of the light source,
- the angle $\theta_{v^l, l}$ between the surface normal at v_l and the direction of the light source,

which leads to:

$$c(v^l) \leftarrow c(v^l) - (c(v^l) - Amb) \cdot \left(\frac{L_l}{\sum_{p=1}^{m} L_p} \cdot \frac{1/d^2_{v^l, l}}{\sum_{p=1}^{m} 1/d^2_{v^l, p}} \cdot \frac{\cos\theta_{v^l, l}}{\sum_{p=1}^{m} \cos\theta_{v^l, p}} \right) \tag{4}$$

Although the denominators can be pre-calculated, we feel that applying Eq. 4 is too expensive to allow the shading of the vertices in realtime in complex environments. We therefore preferred applying Eq. 1, in order to obtain higher response rates.

Rendering

For display, we smooth shadow edges by Gouraud-shading between the vertices. Since the same technique is used for rendering the radiosity results, both kinds of shadow look similar, and the fast shadow does not upset the realistic appearance too much.

3.4 Updating the Realtime Shadow

Throughout user interaction, the fast shadow is continuously updated. For moving the shadow along with a moved object, the old shadows are removed by simply restoring the colors of the previously modified vertices to their original values $c'(v_i)$ that have been stored when the interaction started. After that, the steps described in Sec. 3.2 and 3.3 are repeated for creating the new shadow.

3.5 Switching Back to the Radiosity Shadow

After the user released O_m, the fast shadow is replaced by a more exact radiosity shadow. The radiosity update process described in [Schöf97] is triggered, which calculates the new radiosity solution in a background process and eventually provides new vertex colors. This update process does not perform a complete re-calculation of the radiosity solution, but instead exploits coherences and incrementally updates vertex colors only where necessary. The resulting new vertex colors simply replace the existing ones, thus automatically removing the outdated realtime shadow.

3.6 Speed-Up Through the Usage of Multipoints

In typical virtual environments, there are often objects present that have been modeled in very different degrees of fineness. Often small objects possess huge numbers of vertices that we do not want to consider in detail for a fast shadow computation. Similarly, we sometimes have to deal with fine-resolution patch grids, resulting from the radiosity calculation that are not necessary for realtime shadows.

The Notion of Multipoints

Therefore, in order to speed up the shadow creation, we cluster patch vertices, treating vertices that are close together equally during shadow creation. Clustering is only done once in a pre-process before the interaction starts and results in so-called *multipoints*. It is performed only within objects—vertices of different objects are not clustered into the same multipoint, even if they are close together. The multipoint's coordinates are found by simply averaging the coordinates of the vertices it represents.

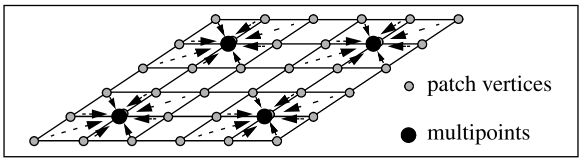


Figure 2: Clustering of patch vertices to multipoints.

During shadow testing (Sec. 3.2), the ray-intersection tests are performed per multipoint, not on a per-vertex basis. All vertices belonging to the same multipoint are then treated equally, being in shadow or not, and are shaded in the same way. Thus, the number of shadow tests is significantly reduced.

Controlling the Shadow Granularity

The resolution of the resulting realtime shadow can be controlled by the size of the region within which the vertices are clustered together. At start-up time, the user defines a clustering factor *Cf*, which is defined as a fraction of the scene's bounding box. I.e., $Cf = 0$ denotes that only vertices with exactly the same coordinates are clustered together, whereas $Cf = 1$ results in all vertices of an object to be clustered into one single multipoint per object. Note that even $Cf = 0$ results in a speed-up, if vertices of adjacent patches are stored separately at each patch.

Strategies for the Multipoint Generation

Scaling the scene's bounding box extension by *Cf*, one obtains the *clustering region Cr'*. The maximum extension of *Cr'* is used for all three axes of the final clustering region *Cr,* in order to avoid different shadow resolutions for each axis. The multipoint clusters can be built in two ways:

- *Voxel-based strategy:* Subdividing the scene into voxels with a size according to *Cr*, one easily can group vertices within one voxel into a multipoint. The cost for this strategy is $O(n)$.
- *Vertex-based strategy:* Selecting a vertex and comparing all other vertices of the same object if they lie within the distance given by *Cr*, one can identify the vertices corresponding to the same multipoint. The region for clustering vertices is thus adjusted by the vertices found, and it is extended until the size of *Cr* is reached. If no more vertices are found that fit into the region, a new multipoint is generated and the search is performed over the remaining vertices. In the worst case ($Cf = 0$) this strategy has cost of $O(n^2)$, in the best case ($Cf = 1$) its cost is linear in the number of vertices.

The voxel-based strategy can result in inefficient clusters: Imagine two vertices close together, but in different voxels—they will be inserted into separate clusters, even if they are the only vertices in these clusters. We therefore think that the vertex-based strategy is better suited, although it is more expensive to compute. In addition, multipoint generation is done at initialization time, and thus does not affect user interaction.

4 INTEGRATION INTO AN INTERACTIVE VR SYSTEM IN COMBINATION WITH A RADIOSITY UPDATE PROCESS

The multipoint shading method presented in the previous section can be easily integrated into an interactive VR system, combined with an efficient radiosity update process. We show here the concept of such an integration and give details on which steps have to be performed by the system in a typical interactive session.

4.1 Concept

We integrated our module for realtime shadow generation directly within the VR system. In the radiosity update module, that runs as a separate process in parallel to the VR system, we inserted a routine for deleting the radiosity shadow of a single object without performing a complete re-computation of the radiosity solution. Both processes are connected via a shared memory segment, and messages are used for triggering the radiosity module. The radiosity process hands resulting vertex colors to the VR system's renderer by writing the data into the shared memory segment. Data and control flow are outlined in Fig. 3.

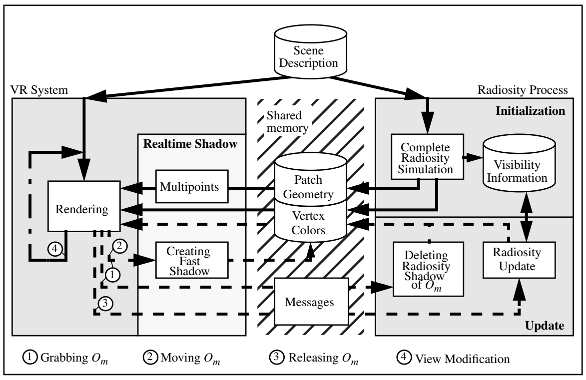


Figure 3: Integrating multipoint shading as realtime shadow feedback into an interactive VR system.

4.2 Initialization

At startup time, the VR system launches the radiosity sub-process. The main steps for initializing the system are:

- *Loading scene geometry, material properties and light source definitions:* Both processes read this data from a joint file.
- *Establishing communication channels:* The VR system initializes the shared memory segment, and the radiosity process plugs itself onto it.
- *Computing an initial radiosity solution:* The radiosity process performs a complete radiosity simulation. Visibility information and form factors are stored for later re-use.
- *Initializing VR components:* While the radiosity simulation goes on, other VR modules may be initialized—e.g., I/O devices, collision detection, etc.

After the initialization is finished on the VR system's side, the scene can be rendered and the user may start moving around in the virtual environment, which is rendered with hardware shading only, as long as the radiosity process has not finished its calculation. After each rendered frame, the VR system checks if vertex colors have been provided by the radiosity process. Once the radiosity module has provided this data, the following steps are performed:

- *Replacing original geometry by patch geometry:* The renderer reads the new geometry and vertex colors from the shared memory segment. From now, the rendered scene contains global illumination.
- *Initializing multipoints:* The fast shadow module is being initialized, and the lists of multipoints are generated as explained in Sec. 3.6, using a clustering factor specified by the user.

4.3 Interactive Phase

After the initialization has been finished, the user may select and manipulate objects. During interaction, the system checks after each frame if new data has been written to the shared memory segment—be it vertex colors calculated by the fast shadow module or updated radiosity values. Thus, updated vertex colors are automatically used for display as soon as they are available. Furthermore, it is checked if a selected object has been moved or otherwise manipulated since the last check—if so, the fast shadow is updated. Similarly, it is registered if the object has been released—if so, the radiosity update process is triggered. The steps that have to be performed for different kinds of user interaction are described in more detail below:

- *Grabbing an object:* If an object is being selected, a message containing an identification of O_m is sent to the radiosity process causing it to remove the former shadow of that object. We apply the *shadow-form-factor-list (SFFL)* [Mülle95] for efficient radiosity update and consider only direct shadows (with respect to the light sources). Since this radiosity update may take some time, it should be performed without blocking the interaction process. The resulting vertex colors $c'(v_i)$ are transferred to the VR system via the shared memory segment and are backed up within the fast shadow module. The fast shadow is calculated, exploiting the multipoints, and the resulting vertex colors are written to the shared memory segment. Thus, an implicit switch to the fast shadow is performed.
- *Moving O_m:* Once a manipulation of O_m is registered, the fast shadow is updated as described in Sec. 3.4. Again, the vertex colors are made available to the renderer via shared memory.
- *Releasing O_m:* When O_m is released, a message is sent to the radiosity process indicating the ID of O_m and a transformation matrix describing the total object modification since the grabbing. The radiosity solution is then incrementally updated, based on the previous solution and by exploiting the visibility information stored at earlier steps (applying again the *SFFL* method), considering also indirect illumination. The resulting vertex colors are presented to the VR system via the shared memory segment.
- *View modifications:* These are handled completely within the VR system. The user may move around at any time, even when an objects is selected for modification or while the global illumination is updated by the radiosity process in the background.

5 RESULTS

The fast shadow method based on multipoints has been applied to several scenes of different complexity and proved to achieve shadow feedback in realtime, even for complex environments. This is feasible due to the adjustable clustering factor, which allows the user to trade visual quality for speed, and thus makes the method adjustable to his specific needs and the complexity of the scene. Useful values for *Cf* appeared to lie usually between 0 and 0.01, while bigger values result in coarse shadows 'jumping' from one place to another and thus destroying any realistic impression. But even for small values fast update rates are achieved, as can be seen from Fig. 4(b).

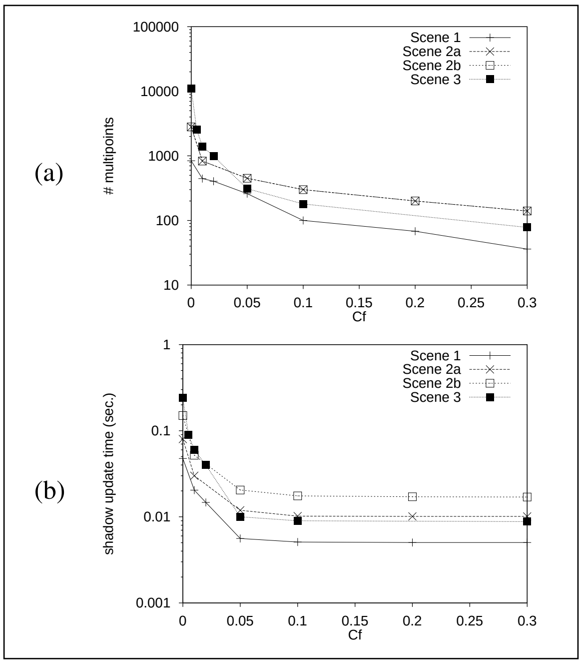
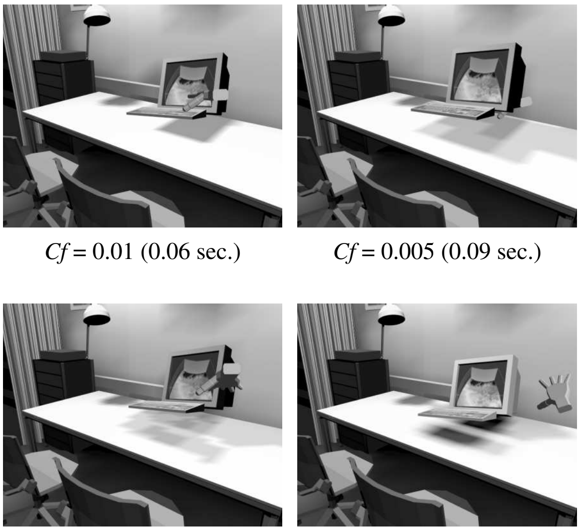


Figure 4: Resulting number of multipoints (a) and shadow update time (b) for different clustering factors in several test scenes:
Scene 1 – 3040 vertices, 4 light sources, Scene 2a – 9095 vertices, 4 light sources, Scene 2b – 9095 vertices, 8 light sources, Scene 3 – 37539 vertices, 4 light sources.

All timings have been measured on a SGI Onyx IR with 194 MHz R10000 processors. In the biggest test scene (Scene 3), that is shown in Fig. 5, the terminal has been moved above the table top. Comparing fast shadow update times for this scene with radiosity update times (~ 1 sec.)[1], one finds speed-up factors between 4 and 16 for small values of *Cf*. For bigger clustering factors, speed-up factors up to 100 can be reached, but with the drawback of significant loss in visual quality.



Cf = 0.01 (0.06 sec.)   *Cf* = 0.005 (0.09 sec.)

Cf = 0.0 (0.24 sec.)   Radiosity shadow (4.5 sec.)

Figure 5: Scene 3 with fast shadows in different resolutions, and with radiosity shadow for comparison.

The time needed for shadow update depends linearly on the number of light sources and on the number of multipoints. Because of the ray-intersection tests, the shadow update time furthermore depends slightly on the complexity of the moved object. As can be seen from Fig. 4, the shadow update time is not further reduced for *Cf* values bigger than ~ 0.1, while the number of multipoints still goes down and decreases the visual quality. This is due to the fact that other tasks with constant costs then become dominant, such as vertex shading. Another interesting point to mention is that shadows sometimes can be updated faster in scenes with a higher number of original vertices than for smaller scenes, since the number of multipoints is in fact smaller. This effect, which can be seen in Fig. 4 for Scene 2 and Scene 3, is a result of different scene structures: Scene 3 consists of fewer objects than Scene 2, but these objects have been subdivided into much more polygons, so more vertices are clustered into one multipoint.

[1] A full radiosity simulation for the given scene (direct illumination only) takes approximately 4.5 sec.

6 CONCLUSIONS AND FURTHER WORK

A method for fast shadow generation has been presented that can easily be integrated as realtime shadow feedback in an interactive system in which radiosity simulation is applied for global illumination simulation. The presented method enables exact radiosity shadows to be added to the environment while providing realtime shadow feedback during user interaction. The multipoint concept offers the possibility for the user to adjust the speed-quality trade-off according to his specific needs. Taking into account poor visual quality, fast shadow updates can be obtained even in complex environments.

The fast shadow algorithm is based on the same vertices as the radiosity algorithm, which makes the integration of both methods into one system easy. Furthermore, the shape of both shadow types is not too different, at least for small values for *Cf*. Nevertheless, we feel that the fast shadow's visual appearance could be improved, for example, by fading towards shadow edges, especially for bigger *Cf* values. In addition, mechanisms for smooth blending between fast and exact shadows should be integrated to avoid 'popping' effects when switching from the fast shadow to the radiosity shadow and vice-versa.

Light sources are considered in a very simplified way, but we are confident that the resulting shadows are good enough to give a visual feedback to the user. However, we have outlined how light sources can be considered more exactly, if necessary.

The method is restricted to fixed patch vertices, since all shadow updates are established by exchanging colors of otherwise unchanged vertices. This makes the shadow update fast, but it is not very flexible. We applied a progressive refinement approach [Cohen88] for the radiosity update, in combination with the shadow-form-factor-list method [Mülle95]. The usage of radiosity methods that refine patch grids adaptively (like, e.g., hierarchical radiosity [Hanra91]), would require an expensive update of the multipoints. Nevertheless, it should be investigated how radiosity update strategies like the Line-Space hierarchy [Drett97] could be combined with the multipoint approach.

7 ACKNOWLEDGEMENTS

This work was supported in part by the European Union (Esprit LTR 24944: ARCADE). Special thanks go to Michael Meixner for valuable help in implementing the multipoint shading algorithm and to Dirk Reiners for fruitful discussions.

8 REFERENCES

[Asthe94] Astheimer, P., Dai, F., Göbel, M., Kruse, R., Müller, S., Zachmann, G.: *Realism in Virtual Reality*, in: Thalmann, N. M., Thalmann, D. (Eds.): Artificial Life and Virtual Reality, Wiley & Sons, 1994, pp. 189–210.

[Asthe95] Astheimer, P., Göbel, M.: *Virtual Design II – An Advanced VR Development Environment,* in: Göbel, M. (Ed.): Virtual Environments, Springer-Verlag, Wien, 1995, pp. 247–272.

[Chin89] Chin, N., Feiner, S.: *Near Real-Time Shadow Generation Using BSP Trees,* ACM Computer Graphics (SIGGRAPH '89 Proceedings), 23(3):99–106, July 1989.

[Chen90] Chen, S. E.: *Incremental Radiosity: An Extension of Progressive Refinement Radiosity to an Interactive Image Synthesis System,* ACM Computer Graphics (SIGGRAPH '90 Proceedings), 24(3):135–144, August 1990.

[Cohen88] Cohen, M. F., Chen, S. E., Wallace, J. R., Greenberg, D. P.: *A Progressive Refinement Approach to Fast Radiosity Image Generation,* ACM Computer Graphics (SIGGRAPH '88 Proceedings), 22(4):75–84, August 1988.

[Drett97] Drettakis, G., Sillion, F. X.: *Interactive Update of Global Illumination Using A Line-Space-Hierarchy*, ACM Computer Graphics (SIGGRAPH '97 Proceedings), 31(3):57–64, August 1997.

[Georg90] George, D. W., Sillion, F. X., Greenberg, D.P.: *Radiosity Redistribution for Dynamic Environments,* IEEE CG&A, 10(4):26–34, July 1990.

[Goral84] Goral, C. M., Torrance, K. E., Greenberg, D. P., Battaile B.: *Modeling the Interaction of Light Between Diffuse Surfaces,* ACM Computer Graphics (SIGGRAPH '84 Proceedings), 18(3):213–222, July 1984.

[Haebe90] Haeberli, P., Akeley, K.: *The Accumulation Buffer: Hardware Support for High-Quality Rendering,* ACM Computer Graphics (SIGGRAPH '90 Proceedings), 24(4):309–318, August 1990.

[Hanra91] Hanrahan, P., Salzman, D., Aupperle, L.: *A Rapid Hierarchical Radiosity Algorithm,* ACM Computer Graphics (SIGGRAPH '91 Proceedings), 25(4):197–206, July 1991.

[Heckb96] Heckbert, P. S., Herf, M.: *Fast Soft Shadows*, Visual Proceedings, SIGGRAPH '96, August 1996, p. 145.

[Mölle96] Möller, T.: *Radiosity Techniques for Virtual Reality – Faster Reconstruction and Support for Levels Of Detail,* Fourth Intern. Conf. in Central Europe on Computer Graphics and Visualization '96, Plzen, Czech Republic, February 1996

[Mülle95] Müller, S., Schöffel, F.: *Fast Radiosity Repropagation For Interactive Virtual Environments Using A Shadow-Form-Factor-List,* in: Sakas G., Shirley P., Müller S. (Eds.): Photorealistic Rendering Techniques, Springer-Verlag, 1995, pp. 339–356.

[Myszk95] Myszkowski, K., Kunii, T. L.: *Texture Mapping as an Alternative for Meshing During Walkthrough Animation,* in: Sakas G., Shirley P., Müller S. (Eds.): Photorealistic Rendering Techniques, Springer-Verlag, 1995, pp. 389–400.

[Schöf97] Schöffel, F.: *Online Radiosity in Interactive VR Applications*, Proceedings of ACM VRST '97, September 1997, pp. 201–208.

[Segal92] Segal, M., Korobkin, C., van Widenfelt, R., Foran, J., Haeberli, P.: *Fast Shadows and Lighting Effects Using Texture Mapping*, ACM Computer Graphics (SIGGRAPH '92 Proceedings), 26(2):249–252, July 1992.

[Slate95] Slater, M., Usoh, M., Chrysanthou, Y.: *The Influence of Shadows on Presence in Immersive Virtual Environments*. In: Göbel M. (Ed.): Virtual Environments '95, Springer-Verlag, 1995, pp. 8–21.

[Soler98a] Soler, C., Sillion, F. X.: *Fast Calculation of Soft Shadow Textures Using Convolution*, ACM Computer Graphics (SIGGRAPH '98 Proceedings), 1998, pp. 321–332.

[Soler98b] Soler, C., Sillion, F. X.: *Automatic Calculation of Soft Shadow Textures for Fast, High Quality Radiosity*, Proceedings of the 9th Eurographics Workshop on Rendering, Vienna, Austria, 1998, pp. 199–210.

[Wange92a] Wanger, L. R., Ferwerda, J. A., Greenberg, D.P.: *Perceiving Spatial Relationships in Computer-Generated Images*, IEEE CG&A, 12(3):44–58, 1992.

[Wange92b] Wanger, L. R.: *The Effect of Shadow Quality on the Perception of Spatial Relationships in Computer Generated Imagery*, Proceedings of the 1992 Symposium on Interactive 3D Graphics, 1992, pp. 39–42.

[Woo90] Woo, A., Poulin, P., Fournier, A.: *A survey of shadow algorithms*, IEEE CG&A, 10(6):13–32, 1990.

[Zhang98] Zhang, H.: *Forward Shadow Mapping*, Proceedings of the 9th Eurographics Workshop on Rendering, Vienna, Austria, 1998, pp. 131–138.