Interactive Global Illumination in Dynamic Scenes

Parag Tole Fabio Pellacini Bruce Walter Donald P. Greenberg

Program of Computer Graphics, Cornell University *

Abstract

In this paper, we present a system for interactive computation of global illumination in dynamic scenes. Our system uses a novel scheme for caching the results of a high quality pixel-based renderer such as a bidirectional path tracer. The Shading Cache is an object-space hierarchical subdivision mesh with lazily computed shading values at its vertices. A high frame rate display is generated from the Shading Cache using hardware-based interpolation and texture mapping. An image space sampling scheme refines the Shading Cache in regions that have the most interpolation error or those that are most likely to be affected by object or camera motion.

Our system handles dynamic scenes and moving light sources efficiently, providing useful feedback within a few seconds and high quality images within a few tens of seconds, without the need for any pre-computation. Our approach allows us to significantly outperform other interactive systems based on caching ray-tracing samples, especially in dynamic scenes. Based on our results, we believe that the Shading Cache will be an invaluable tool in lighting design and modelling while rendering.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism; I.3.3 [Computer Graphics]: Picture/Image Generation; I.3.2 [Computer Graphics]: Graphics Systems;

Keywords: Rendering, Ray Tracing, Parallel Computing, Rendering Systems, Illumination, Monte Carlo Techniques

1 Introduction

Interactive computation of global illumination is of major importance in the field of computer graphics, especially when applied to engineering and design. Effects such as soft shadows, diffuse interreflections and caustics are important cues for the human visual system. All these effects can be correctly simulated using global illumination algorithms such as path tracing [Kajiya 1986] or the RADIANCE system [Ward 1994], but only at a huge computational cost. Needless to say, these techniques are not suitable for interactive applications.

Modern graphics hardware can render complex environments at interactive rates. The realism of hardware-based approaches can be increased by using pre-computed radiosity textures, environment maps and sophisticated pixel and vertex shaders [Lindholm et al. 2001]. However, these approaches limit the choice of shading algorithms. In addition, the amount of pre-computation or hand-tuning required is often quite significant. Inspite of these drawbacks, hardware rendering remains the only viable choice for interactive applications, primarily because of its efficiency in performing hidden surface removal and texture mapping.

Recently, interactive walkthroughs of global illumination have become possible by clever caching of ray-traced samples [Walter et al. 1999; Simmons and Séquin 2000; Stamminger et al. 2000] or by efficiently updating partially pre-computed radiosity solutions [Drettakis and Sillion 1997; Granier and Drettakis 2001]. However, these techniques provide only limited interaction; large-scale scene manipulation or the movement of primary or bright secondary light sources typically results either in a loss of interactivity or a prolonged degradation in image quality.

Applications such as lighting design require interactive global illumination solutions while allowing unrestricted manipulation of the scene including the lights. Interactive walkthroughs with progressively refined global illumination solutions are also useful in such applications. In addition, it is also important to have scalability with processing power and flexibility in the choice of global illumination algorithms.

In order to meet these goals, we have designed a novel scheme for caching high quality samples generated by pixel-based rendering algorithms such as path tracing and then interpolating between these samples using graphics hardware. Our Shading Cache is a hierarchical subdivision mesh attached to each primitive in the scene, with lazily evaluated shading values at the mesh vertices. It is refined by choosing locations in the image plane that require more accuracy. The selection of these locations is made using a priority map, which captures the likelihood of error due to interpolation, view-dependence and object motion. We also introduce a novel flood filling operator for efficiently capturing the high frequency detail in the image to supplement the priority based sampling. Finally, we use graphics hardware to efficiently interpolate between the shading samples for polygonal or curved surfaces. By using the hardware for image reconstruction, we ensure that the scene geometry and textures are accurately reproduced at all times, even when the shading values are only available at a very low resolution. The entire process is illustrated in Figure 1.

While the idea of image reconstruction from a sparse set of high quality samples has been previously explored [Walter et al. 1999; Simmons and Séquin 2000; Stamminger et al. 2000], we believe that our approach offers significant improvements in speed and quality, especially in dynamic scenes. The reason for our superior performance is the use of object space caching and interpolation, combined with a superior sampling scheme that refines the mesh only in regions of high importance and interpolates in other regions.

After a discussion of previous work in Section 2, we give an overview of our system in Section 3 and the sample selection process in Section 4. We then show how moving objects and view dependent illumination can be efficiently handled in Section 5. In Section 6, we provide performance statistics and in Section 7, we compare our system to other interactive global illumination techniques. We conclude with a discussion of the advantages and limitations of our work and future directions for research.

*580 Rhodes Hall, Ithaca, NY 14853.
WWW: http://www.graphics.cornell.edu/
E-mail: {parag,fabio,bjw,dpg}@graphics.cornell.edu

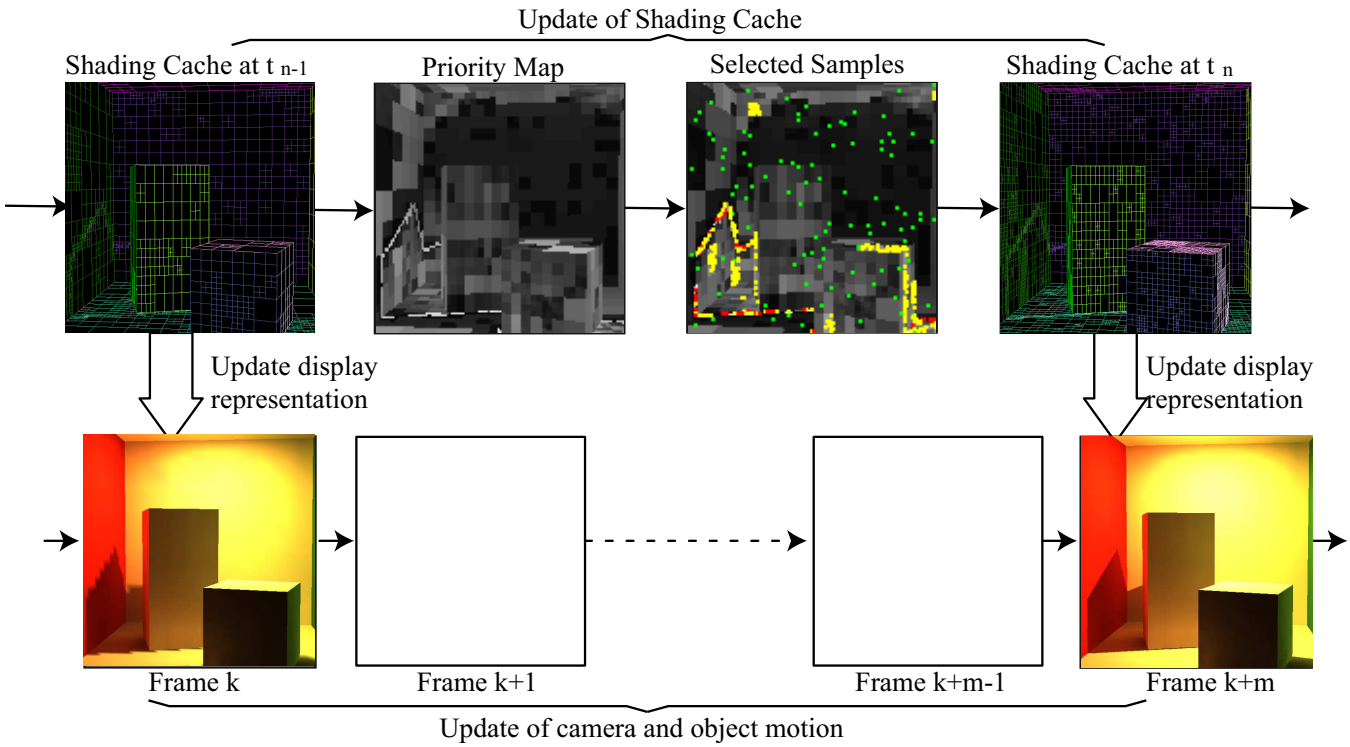


Figure 1: The Shading Cache at time t_{n-1} is used to generate a display representation which is drawn using graphics hardware. The Shading Cache is updated asynchronously of the display by first computing a priority map and then using it to select additional samples. The resulting mesh at time t_n is used to compute a new display representation which is used starting from frame $k+m$.

2 Previous work

Interactive rendering: Current interactive rendering methods are mostly based on graphics hardware. Hardware rendering can be made more realistic by adding shadows [Segal et al. 1992], specular reflections and refractions [Diefenbach and Badler 1997] and general BRDF models [McCool et al. 2001]. [Udeshi and Hansen 1999] employ parallel graphics hardware for shadows and CPUs for computing indirect illumination, thus extending the hardware shading model to include approximate one-bounce indirect lighting. Alternatively, an approximate representation of the global illumination can be pre-computed and viewed interactively using graphics hardware, as in [Walter et al. 1997a; Stamminger et al. 1999; Bastos et al. 1999; Stürzlinger and Bastos 1997]. In the context of ray-tracing, [Parker et al. 1999] and [Wald et al. 2001] used low level optimizations and parallelism to show that interactive ray-tracing of complex scenes is feasible on today's computers, albeit with simple shading models. The Holodeck data structure [Larson and Simmons 1999] caches rays in order to provide an interactive walk-through, but it supports a very restricted object motion paradigm.

Radiosity based methods: The radiosity method [Goral et al. 1984] can be used to pre-compute view-independent radiosity values which can be viewed interactively. Extensions have been proposed for view-dependent illumination [Chen et al. 1991; Aupperle and Hanrahan 1993] and view-driven refinement [Aupperle and Hanrahan 1993; Smits et al. 1992], but they do not provide interactive rates. On the other hand, several researchers have demonstrated that a radiosity solution can be updated at interactive rates after localized object motion, see for example, [Chen 1990; Drettakis and Sillion 1997]. However, the image quality resulting from such systems may not be high because of the difficulty in reproducing discontinuities such as sharp shadows using radiosity.

Recently, [Granier and Drettakis 2001] used a combination of hierarchical radiosity with clustering [Sillion et al. 1995] and particle tracing [Jensen 1996; Walter et al. 1997b] to simulate global illumination including non-diffuse effects. In their system, diffuse scenes can be visualized using graphics hardware, while the RenderCache [Walter et al. 1999] is used for scenes containing non-diffuse surfaces. Their system even updates the global illumination solution on the diffuse surfaces at near-interactive rates when an object is moved, but they report pre-computation times ranging from 35 minutes to over an hour. In addition, their technique is likely to require a significant amount of time to update the illumination after large changes to the scene such as moving light sources.

Caching schemes: The RADIANCE system [Ward 1994] produces high quality images off-line by caching lazily computed diffuse inter-reflections and interpolating between the cached values. Radiance interpolants [Teller et al. 1996; Bala et al. 1999] cache radiance in 4-D line space, exploiting spatial and temporal coherence by using these interpolants over several frames. However, they assume the Whitted ray tracing model [Whitted 1980] for deriving the physical error bounds used to refine the interpolants. Moreover, their primary goal is to generate images within a specified error tolerance, so the frame rate may drop when additional interpolants need to be constructed to satisfy the given error bounds.

Reprojection-based schemes, inspired by frameless rendering [Bishop et al. 1994], are most closely related to our work. [Walter et al. 1999] cache the results of previous images in a RenderCache and use reprojection to reduce the number of pixels that need to be computed per frame. The RenderCache also supports simple object motions. The Tapestry system of [Simmons and Séquin 2000] uses an image-plane Delaunay mesh to reduce the visibility artifacts of the RenderCache. However, this does not eliminate all the geometric artifacts. In particular, geometric edges have to be reconstructed using a very large number of point samples and even then, may be distorted when the camera moves. [Stamminger et al. 2000], on the other hand, improve a pre-computed radiosity solution by using object-local or camera-local corrective textures. The radiosity solution together with the corrective textures are then

displayed without any geometric artifacts using graphics hardware. However, using camera-local projective textures introduces reprojection artifacts in the shading. Also, object-local textures must be of an extremely high resolution in order to reconstruct sharp shading features such as hard shadow boundaries. Finally, Tapestry and Corrective Texturing are primarily meant to provide global illumination walkthroughs and extending these systems to handle dynamic scenes seems non-trivial.

Image reconstruction from a sparse sample set and adaptive sampling: An entire image may be reconstructed from a sparse set of samples to enable rapid previewing [Darsa and Costa 1996]. The reconstruction may be improved by pre-processing the geometry [Pighin et al. 1997]. However, such pre-processes are too expensive for interactive applications. Computation of static images or animation sequences can be accelerated using adaptive refinement and perceptually-driven sampling [Guo 1998; Ramasubramanian et al. 1999; Myszkowski et al. 2001; Yee et al. 2001]. But the resulting improvement in efficiency is not sufficient to achieve interactivity.

3 System Overview

Computation of global illumination is very slow, so in order to provide a high frame rate, our system decouples the refresh of camera view and object motion from the update of shading values. This decoupling is achieved by caching recently computed shading values in an object-space mesh data structure called the Shading Cache. The data structure for the Shading Cache is a hierarchical patch tree, similar to that used in hierarchical radiosity [Hanrahan et al. 1991]. Our system can handle all locally parameterizable surfaces; the current implementation supports triangles, quadrilaterals and bicubic curved surfaces.

Each patch in the mesh stores the last computed shading values for its vertices. During initialization, the mesh for each geometric primitive is set to be a single un-subdivided patch. The shading values for these root patches are not pre-computed, but evaluated lazily when required for generating a good image. When a patch is selected for update, we either evaluate the shading values for its vertices or subdivide the patch into four children and evaluate the shading values for their vertices. For non-diffuse surfaces, the exitant radiance for the current viewing direction is used as the shading value, while for diffuse surfaces, the irradiance is used. This distinction allows us to use texture mapping hardware during image reconstruction.

The main building blocks of the system are shown schematically in Figure 2. The *Shading Cache* is updated in a view-driven fashion in order to refine the rendered image. An efficient display representation is then generated for the *Image Generator*, which uses the graphics pipeline for image reconstruction, including texture mapping, Gouraud interpolation and hidden surface removal. The *User Interaction Loop* [1] runs asynchronously from the update of the Shading Cache. This allows us to refresh the camera view and to display moving objects at a high frame rate without errors in geometry or texture, irrespective of the shading speed. The *Sample Selector* assigns priorities to each pixel depending on the estimated error in its currently displayed value and uses this priority map to select locations in the image plane for computing additional shading samples. These samples are then computed by the *Sample Renderer* using any suitable global illumination algorithm.

This basic idea of decoupling the camera view update from shading calculations has been explored in the RenderCache [Walter et al. 1999], Tapestry [Simmons and Séquin 2000] and Corrective Texturing [Stamminger et al. 2000]. Of these, only the corrective texturing approach reproduces scene geometry accurately. All three of them rely on point sampling to reproduce textures. Our system always reproduces the scene geometry and textures accurately and the frame rate is not related to the shading speed or display resolution (within the limits of graphics hardware). This results in higher quality images and greater interactivity. Our choice of a hierarchical mesh data structure is superior to a "regularly subdivided corrective texture" because it can reproduce high frequency features such as hard shadows without using excessive texture memory. We describe our system in detail in the following three sections. Later, in Section 7, we provide additional comparisons between our system and the other caching schemes.

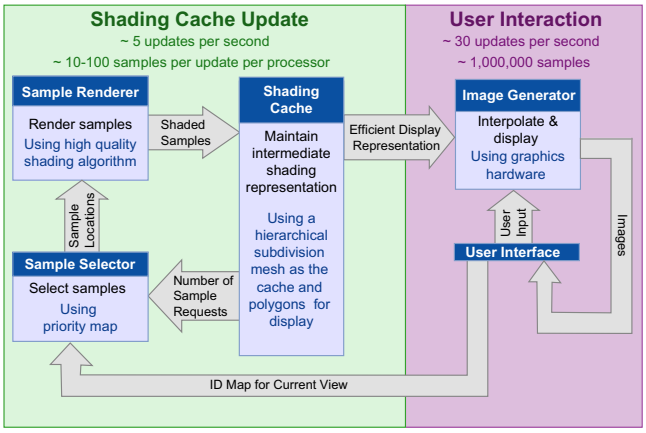


Figure 2: An Overview of the System

3.1 Updating the Shading Cache

The Shading Cache must be updated in order to progressively refine the displayed image. This may involve computing regions of the Shading Cache that were not required for previous frames or those that have inaccurate shading values. In addition, a display representation that can be efficiently handled by the graphics pipeline must be generated. The sequence of steps that perform these operations is described next.

A patch identifier map is drawn and read back by the Image Generator when required by the update process. The Sample Selector goes over this ID map and evaluates the priority for each patch that is visible in the current view. These priorities are then used to select image plane locations for refinement. The process of assigning priorities and selecting samples is described in detail in Section 4.

The patches that correspond to each of the sample locations are then determined using the ID map. A list of selected patches with the number of samples lying on each patch is created. For each patch in this list, we either update the shading values for its vertices or subdivide the patch and compute new shading values for the vertices of its children. Patches with more sample requests are subdivided faster than others until eventually all patches project to one pixel in the image plane.

We use aging to limit the size of the Shading Cache and to prevent the frame rate from dropping to an unacceptable level. For this, patches that are visible in the current view are marked as "seen". Then, if the cache size has grown above a given threshold, patches that are no longer useful for rendering the current or immediately foreseeable views are deleted. Thus, all children of a patch are deleted if (a) none of them has been seen in a user-specified number of frames, or, (b) the patch itself projects to less than one pixel in the given view. This "not recently used" replacement strategy is based on the assumption that recently seen patches are likely to be in view again in the near future and hence should be retained, whereas patches that have not been in view recently are likely to remain outside our view.

[1] We have implemented the User Interaction loop using OpenGL

3.2 Sample Renderer

Shading values for selected samples are computed by the Sample Renderer. In our implementation, we use area sampling of light sources to compute direct illumination with soft shadows and a bidirectional path tracer [Veach and Guibas 1994] to compute indirect illumination. All examples in this paper have been rendered using 100 area samples per light and between 400 and 1200 samples for the indirect illumination. Sampling is performed using the Halton sequence [Halton and Weller 1964]. With these settings, our renderer can shade 10 to 100 points per second on one 1.7GHz Pentium 4 processor in moderately complex scenes. This compares to to half a million pixels in a 800 X 600 image. Our system is designed to scale with the processing power available for rendering samples. We have found the system to be usable on dual processor workstations with one of the processors rendering the samples, scaling up to 16 processors or more.

3.3 Efficient display

The Shading Cache is used to generate an efficient display representation consisting of Gouraud interpolated, texture-mapped polygons. For planar objects, the patches in the Shading Cache can be displayed as is without introducing visibility artifacts. However, doing so for curved surfaces results in geometric errors, such as tears in the interior and incorrect silhouettes. Therefore, we tessellate the curved surfaces in a view-dependent fashion and display this tessellation instead of the Shading Cache. Note that the vertices of this tessellation are not shaded using the expensive sample renderer, but simply by interpolation from the Shading Cache.

The display representation uses OpenGL vertex arrays for rendering. In order to prevent synchronization overhead between the user interaction loop and the update loop, we maintain two copies of the vertex arrays for double buffering. Thus, synchronization is required only when swapping the arrays. We perform incremental updates to the arrays by keeping track of which patches are affected in the current update cycle. This way, the overhead of maintaining the Shading Cache is minimal.

To summarize, the Shading Cache is updated using a view-driven refinement process and then used to generate a display representation that can be displayed at a high frame rate without errors in geometry or texture.

4 Sample Selection

Since the Sample Renderer can only render a very small number of pixels per frame, the pixels to render must be selected carefully in order to provide good image quality. We use a priority-based scheme similar to [Simmons and Séquin 2000] to direct the image plane samples into regions where the object-space Shading Cache needs more accuracy. In addition, we also use a novel flood filling operator to quickly refine the Shading Cache in regions containing discontinuities or high gradients. The process of sample selection is illustrated in Figure 3. In Section 5, we extend this basic scheme to handle view-dependent illumination and dynamic scenes efficiently.

4.1 Estimation of interpolation error

The priority of a patch is based on the estimated error in the shading values of the pixels that it projects to. We estimate this error as the difference between the maximum and minimum colors of the vertices of the patch, after tone-mapping with the s-shaped operator of [Tumblin et al. 1999] with weights of $5/16$, $9/16$ and $2/16$ for the red, green and blue channels. This is because the visual difference between two radiance values is linear in the difference of their tone-mapped values.

The error metric is simply the estimated interpolation error of each pixel covered by this patch. Since geometric discontinuities are always accurately reproduced by our system, we do not need the depth term used by Simmons and Séquin. In addition, we also do not quantize and pack the priority into 8 bits. Instead we simply generate an id map by drawing the 32-bit pointer of the patch into the color channel. This allows us to evaluate the priorities lazily for just the visible patches and also provides greater flexibility in choice of error metrics. For example, our system can support view-dependent perceptual error metrics similar to [Ramasubramanian et al. 1999; Yee et al. 2001; Dumont et al. 2001].

4.2 Sampling the priority map

The patch priorities can be thought of as the likelihood of error in the value of a pixel covered by the patch. Therefore, we would like to sample the patches with the highest priorities. As correctly observed by [Walter et al. 1999], sorting the priorities and selecting the highest priority patches will lead to a very poor spatial distribution of samples in the image plane. To avoid this problem, we use a hit-and-test approach. We draw an ID map at the beginning of each update cycle and compute the priority of every patch in view. After this, we normalize the priorities to a 0 to 1 range and treat the priority of a patch as the probability of selecting it for accurate computation.

In our hit-and-test approach, a random pixel in the id map is selected for testing. Since the priorities are normalized between 0 and 1, we simply generate a random number in that range and accept the pixel if its priority is greater than the random number. Thus, the total probability of selecting a patch is proportional to its priority and its projected area in pixels. Note that our approach does not require the selection of an ad hoc threshold for accepting samples and is also guaranteed to eventually select all pixels in the image plane. Outlier priorities can be a problem in the normalization step, so instead of mapping all priorities between the minimum and maximum value linearly to a [0,1] range, we instead compute the average and variance of the priorities and map values that are 4 standard deviations above and below the average to 1 and 0 respectively.

4.3 Avoiding bias in the sampling

If we simply use a difference-based metric (or even a contrast-based one), it is possible that certain patches might get a zero or very low priority if the shading values at their corners are almost equal. However, this does not necessarily mean that the radiance function is constant in the interior of those patches. So, in order to avoid bias in the sampling, we accept samples with a certain small probability, ϵ, irrespective of their priority. Thus, the total priority of selecting a pixel is given by

$$p_{total} = \epsilon + (1 - \epsilon) \times p_{normalized} \quad (1)$$

We have found that ϵ values from 0.01 to 0.05 work well in practice; we used 0.03 in all our examples.

4.4 Reconstructing shading discontinuities

Since the probability of hitting a patch with random sampling is proportional to its projected area, small patches are very unlikely to be tested. However, since the sampling scheme is trying to direct samples into regions of high gradients, such regions inevitably contain a large number of small patches. Thus, the patches around the shading discontinuity that need to be refined quickly end up being rarely tested by the hit-and-test process. In order to improve the rate of convergence in these cases, we run a second image plane selection phase that tests the neighboring locations of the already selected high priority samples and accepts them if their priority is greater than the current sample. This can easily be done by a flood filling algorithm so that already visited regions will not be touched again. The flood fill is stopped when all neighboring pixels have priority below threshold. This second phase is similar in spirit to the one presented by [Hart et al. 1999].

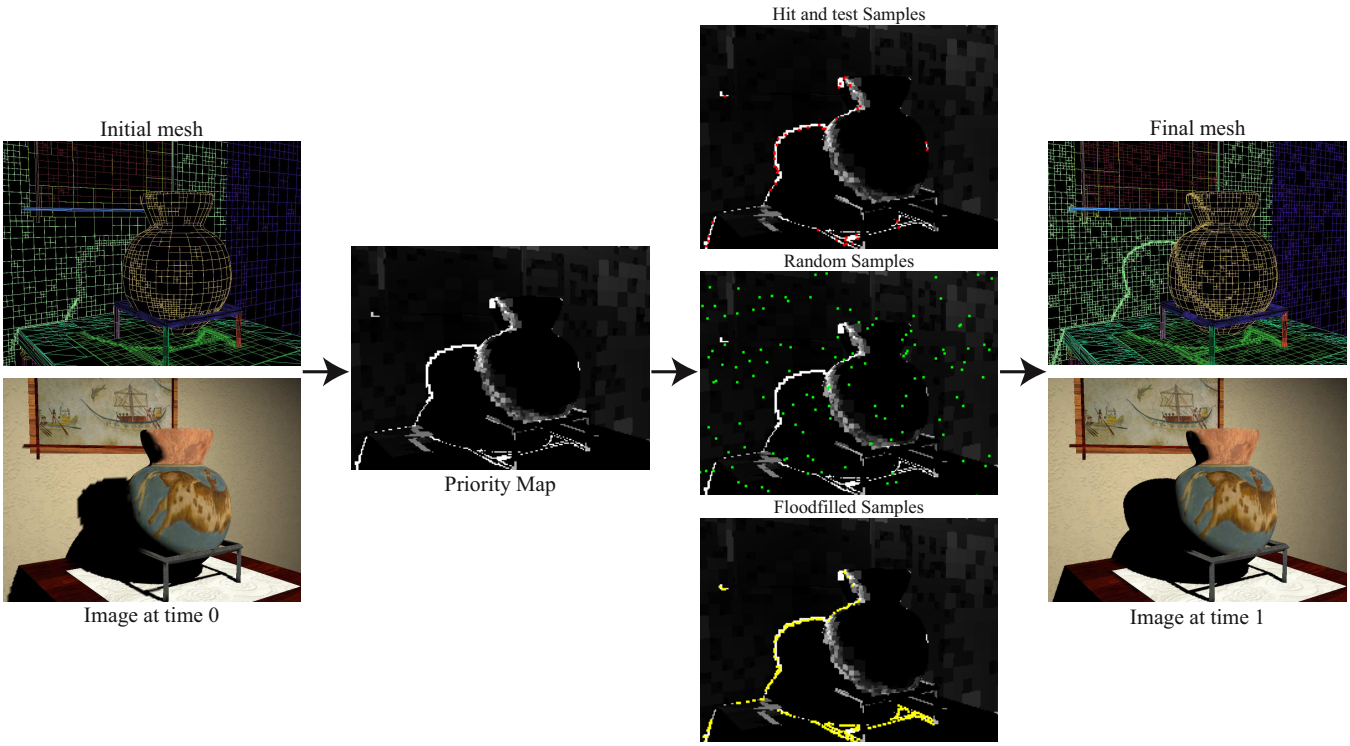


Figure 3: Illustration of the sample selection algorithm. The sampling procedure starts with the assignment of priorities to the patches in the id map. The priority map is used to select patches for accurate computation according to three criteria as illustrated above.

We found that this two phase approach of selecting samples behaves much better than the sorting approach or the simple hit-and-test. This is because the first phase has the advantage of uniformly spreading the samples in the image plane (as for hit-and-test), and the second one ensures that regions with small, high priority patches will be highly sampled (as for sorting).

Figure 3 illustrates the entire sample selection process. The leftmost column shows the initial mesh and the corresponding image, which is used to compute the priority map. Samples selected using this map are shown in the third column, color coded to show the mechanism of acceptance: red for samples selected by the hit-and-test mechanism, green for random samples and yellow for flood filled samples. The final mesh and the resulting image are shown in the rightmost column.

5 View-dependent effects and object motion

When the shading values in a scene change with camera or object motion, it is necessary to direct sampling specifically into the regions that are most affected by such motion. In this section, we extend the basic priority assignment and sample selection scheme of Section 4 to efficiently update view-dependent and motion-dependent shading.

5.1 View-dependent illumination

The shading of non-diffuse objects changes with the viewpoint, and hence should be recomputed when the camera moves. However, given the constraint of interactivity, we cannot afford to recompute the entire image each frame. Instead, we use the mechanism of priorities described in the previous section to preferentially re-compute the illumination on non-diffuse objects by increasing their priorities. This is achieved by keeping track of the last time at which each patch in the Shading Cache was computed. Non-diffuse patches computed before the last camera motion are assigned an age priority proportional to the difference between the current time and their time of computation (measured in cycles of the Shading Cache update loop). The constant of proportionality is always at least equal to 1; it is higher for objects whose shading is expected to change more rapidly with viewpoint. For glossy objects, we select this constant using a heuristic based on the perceptually uniform gloss scale proposed in [Pellacini et al. 2000]. Similar strategies can be constructed for other reflection models. More sophisticated heuristics that take into account the change in the viewing direction can also be developed. However, our simple heuristic performed adequately in our tests; for example, notice the sample distribution for the glossy objects in Figure 4. Note that the estimated error and the age priority are added together before the normalization and biasing steps described in the previous section.

5.2 Motion-dependent shading

Moving objects present a tougher challenge than camera motion. We know that when the camera moves, we should only recompute the shading for non-diffuse surfaces. However, when an object moves, we first need to detect a change in shading and then re-compute all the affected regions of the scene. We rely on the random sampling to detect large changes in shading caused by object motion. As soon as the shading values of one patch change, the interpolation error of its neighboring patches increases, thereby raising their priorities. In addition, objects that detect such a change in shading are also aged just like non-diffuse objects, and eventually they are selected for re-computation. For objects with both view-dependent and motion-dependent shading, the age priority is chosen to be the maximum of the age priorities from view dependence and motion dependence.

It is important to note that we update the position of the moving objects instantaneously. The shading changes resulting from the object motion are updated asynchronously from position changes. Thus, users can receive useful feedback quickly even when the shading cannot be updated fast enough.

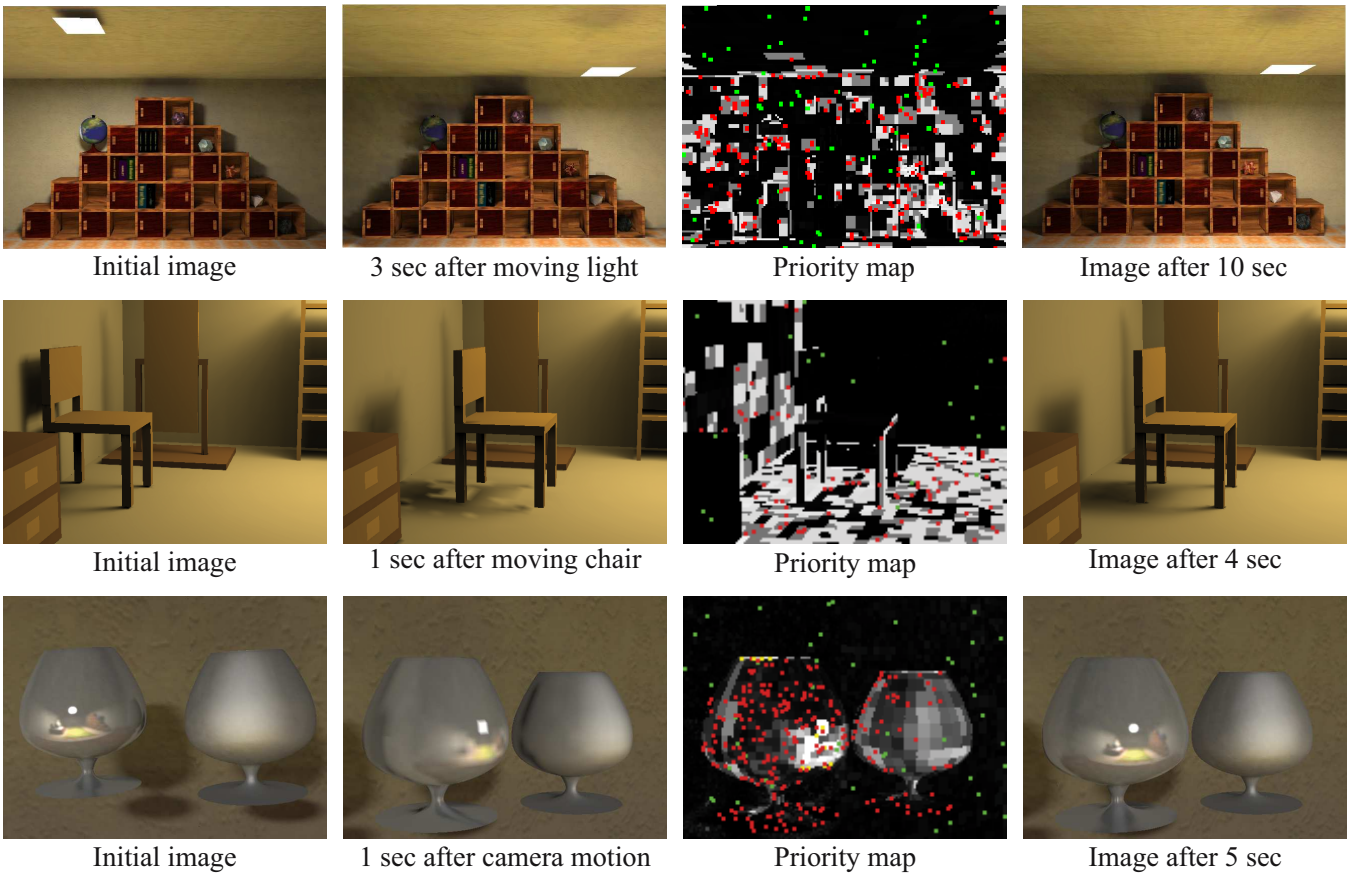


Figure 4: Handling moving and non-diffuse objects: The top row shows a moving light source, the middle row shows a moving chair in an (only)indirectly lit scene and the bottom row shows view dependent illumination.

5.3 Providing fast updates

When the shading of a large portion of the scene is changing due to camera or object motion, the slow Sample Renderer often cannot provide updates at a sufficient rate. So in order to improve the quality of the image presented to the user, it is necessary to limit the number of patches that need to be re-computed. This is achieved by lowering the resolution of the Shading Cache on the objects whose shading has been found to be changing. The new resolution is determined based on a target "cleaning up" time, which is a user-specified parameter. The resolution drop is greater when the target time is lower and vice versa. When the shading stops changing, we continue refining the Shading Cache up to a 1 pixel projected size, so that the image can converge at the original screen resolution.

6 Results

We tested our system on the scenes shown in Figure 5. The first scene is a room with 1 area light and about 4000 primitives (including 9 untessellated curved objects). The second scene has 1 area light and about 300 primitives. The illumination in this scene is almost entirely indirect. The third scene is a pool hall with 11 point lights and about 10000 primitives (including 15 untessellated curved objects). For the purpose of these tests, the sample renderer used 100 samples per area light to compute direct illumination and 400 bidirectional samples for indirect illumination. There was no pre-computation other than building a hierarchical regular grid for accelerating ray-casts.

We ran walkthrough sessions on 2 sets of hardware for timings. First, we used one dual 1.7GHz Pentium 4 workstation, and then we added 8 more dual Pentium 4 workstations to render the samples. The display machine in both cases had a GeForce3 graphics card.

	Scene 1	Scene 2	Scene 3
Points rendered per update	70	72	61
Sample Selection (ms)	50	36	12
Patch subdivision (ms)	<1	<1	<1
Curved surface tessellation (ms)	12	0	9
Aging (ms)	<1	<1	<1
Updating display meshes (ms)	1	1	2
Rendering time (ms)	611	490	506
Time per Iteration (ms)	675	529	533
Frame rate	45	60	40

Table 1: Performance statistics: 1 rendering client

The results of these tests are shown in Tables 1 and 2. Note that for the multi-processor version, the samples to be rendered are sent to the renderers right after the sample selection is completed and read back at the end of the update iteration. Thus, the total update time per iteration may be less than the sum of the individual steps. These results show that our system has very little overhead other than sampling and curved surface tessellation. Of the two, sampling is approximately 10% of the total time. Also, the GeForce3 graphics card supports view-dependent tessellation of curved surfaces in hardware. This feature could be used to reduce the tessellation overhead. In addition to these timings, Figure 4 shows typical performance with 8 dual Pentium 4 rendering clients in the case of dynamic scenes.

Our system can provide global illumination solutions in scenes with moving geometry and view-dependent lighting. The system is usable for interaction on a single workstation and is scalable with processing power. The overhead of maintaining and refining the



Figure 5: Test scenes used for measuring performance

Shading Cache is minimal and so most of the available processing power is devoted to the task of rendering high quality samples. In a typical walkthrough, the number of patches (in excess of the original geometry) is about 20,000. When the camera stops moving, this number goes up and may approach the number of pixels in the image. The memory usage per patch (including the double buffered display representation) is under 200 bytes.

Caustics present a challenge to all global illumination renderers based on path tracing from the eye. Since we compute the shading of only one point at a time, we cannot compute caustics effectively using bidirectional path tracing. However, this is a limitation of the underlying sample renderer, not of our system. Lighting effects such as caustics can be simulated using a sample renderer that can efficiently render them. To demonstrate this, we wrote a caustic renderer that specifically samples the specular surfaces in the scene, shown in Figure 6. While this particular algorithm is not suitable for rendering caustics in complex scenes, it does show that our system can support different global illumination algorithms. For example, dynamically updated caustic textures [Granier and Drettakis 2001] could be used for rendering caustics in more complex scenes.

	Scene 1	Scene 2	Scene 3
Points rendered per update	288	363	328
Sample Selection (ms)	48	12	16
Patch subdivision (ms)	<1	1	<1
Curved surface tessellation (ms)	5	0	8
Aging (ms)	<1	1	1
Updating display meshes (ms)	1	2	2
Rendering time including network overhead (ms)	325	424	420
Time per Iteration (ms)	375	444	440
Frame rate	45	60	40

Table 2: Performance statistics: 8 dual Pentium 4 rendering clients

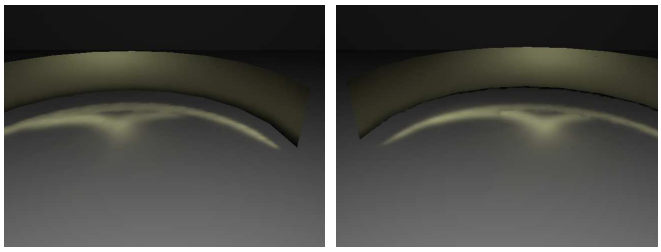


Figure 6: Rendering caustics: The first image is computed by area sampling the glossy surfaces. The second image is a screen shot 5 seconds after moving the ring.

7 Comparison with similar systems

Our approach based on caching shading values in object space is similar to the RenderCache [Walter et al. 1999], Tapestry [Simmons and Séquin 2000] and Corrective Texturing [Stamminger et al. 1999]. Of these, the Tapestry data structure uses an image plane Delaunay triangulation, and it is unlikely that this approach can be applied efficiently to dynamic scenes. Corrective texturing, on the other hand, is primarily useful as a walkthrough technique because of its reliance on a pre-computed radiosity solution. Finally, the RenderCache is a general caching scheme that can be used with any pixel-based renderer and supports moving objects. Since the RenderCache is the only system which provides the same feature set as our system, we performed additional performance comparisons between the two.

The author of the RenderCache paper has allowed us to use his original implementation for comparisons. This implementation has been hand optimized to run on dual Pentium 4 workstations and supports multiple rendering clients, similar to our system. We interfaced our rendering clients to the RenderCache and compared the two systems running on identical hardware (one dual 1.7GHz Pentium 4 for display and 8 dual Pentium 4 rendering clients). The equal time and equal quality comparisons in Figures 7, 8 and 9 show that our system offers substantial performance improvement both in static and dynamic scenes.

In order to compare different caching schemes, we extend the concept of the render mismatch ratio from [Walter et al. 1999] and

	Shading Cache	RenderCache	Tapestry	Corrective Texturing	Hierarchical Radiosity
Pre-computation	No	No	No	Yes	Yes
Normalized Mismatch Ratio	1000	10	1000	1000	N/A
Display Frame Rate	>30	10-20 (at 512×512)	2-10	2-10	>30
Dynamic scenes	Yes	Yes	No	No	Yes
Reprojection artifacts	No	Yes	Yes	In shading only	No
High frequency shading detail	Yes	Yes	Yes	Yes (using large textures)	No
Restrictions on geometry	Locally parameterizable	None	None	Parameterizable	Parameterizable

Table 3: Comparison between various interactive global illumination techniques

define the normalized mismatch ratio as the number of pixels in a frame divided by the number of new points rendered per second by the sample renderer. The normalized mismatch ratio is essentially the time in seconds to compute each pixel in the image, and is a characteristic of the sample renderer being used. In all of our scenes, the normalized mismatch ratio is between 250 and 2500 (using an image size of 512×512 and 8 dual Pentium 4 rendering clients). Our system is very responsive even at such high ratios, while the RenderCache is only effective at ratios near 10 (or lower).

In addition to the caching-based schemes, hierarchical radiosity solutions can also be updated at interactive rates after object motion [Drettakis and Sillion 1997]. An extension of this approach is the hybrid hierarchical radiosity/particle tracing algorithm of [Granier and Drettakis 2001] which can compute the correct lighting on diffuse surfaces including non-diffuse effects. Both of these systems require an initial radiosity solution to be computed and only illuminate the diffuse surfaces. Also, large changes to the scene such as moving a light source will cause a severe slowdown. Moreover, these techniques can only provide a coarse resolution at interactive rates and so the quality of images may be poor, especially near shadow boundaries. Our system uses a hierarchical radiosity-like data structure, but the view-driven refinement allows us to generate high quality images quickly even in the presence of sharp shadows and high frequency indirect illumination (see for example, Scene 2 in Figure 5).

The comparison between the various interactive global illumination techniques is summarized in Table 3. We believe that our approach based on the Shading Cache is the best suited for applications such as interactive lighting design and modelling while rendering.

8 Discussion and Future Work

The renderings produced by our system in the initial stages have aliasing artifacts in sharp shadows and specular reflections, along with light and shadow leaks similar to radiosity. However, unlike traditional radiosity systems, these artifacts are quickly detected and removed by our sampler. Non-importance-based radiosity systems typically need to use discontinuity meshing to remove these artifacts, whereas the Shading Cache merely needs to be subdivided to a level that is sufficient for image plane anti-aliasing.

All caching systems suffer from popping artifacts when the rendered samples are noisy; blending new values slowly with the old values may reduce the popping. In addition, contrast-based adaptive sampling schemes are falsely led into regions of noise. In our current system, we have side-stepped this problem by using Quasi Monte Carlo sampling and setting the sampling rate high enough to reduce banding artifacts. However, noise can be easily handled by our sampler by storing a variance estimate in addition to the radiance estimate at each vertex. The variance can then be used to decide whether to reduce noise by refining the radiance estimate at the vertices or to reduce contrast by subdividing the mesh. Such a scheme would probably also be useful in the RenderCache and Tapestry systems.

We do not claim to provide real-time global illumination. Indeed, that goal is still at least an order of magnitude away. At the same time, we do not stall object or camera motion while shading values are being updated. This way the user can at least have a smooth interface and a high frame rate. However, depending on the application, blocking all motion until a reasonable image is available may be the right choice and can be easily incorporated into our system.

Our current implementation can handle all parameterizable surfaces. The parameterization does not have to be well-behaved; we have rendered surfaces with singularities such as spheres without difficulty. In a manner similar to the REYES algorithm [Cook et al. 1987], our subdivision scheme can be easily extended to handle all locally parametric surfaces that can be "split" into parameterizable primitives, including subdivision surfaces of arbitrary topology. Alternatively, a hybrid approach of point sampling the non-parametric surfaces and using the Shading Cache representation elsewhere also seems promising.

We have demonstrated the usability of our system in moderately complex models with up to 10,000 primitives and difficult illumination conditions. For handling environments that are geometrically much more complex, the hardware side may need to use frustum culling or occlusion culling [Schaufler et al. 2000]. However, for the purpose of computing shading values, only the on-screen geometric complexity matters. Our method will not provide significant speedups if the on-screen complexity is large, since at least one sample per visible primitive is required to get a reasonable estimate of the image. Additional separation of the shading representation from the geometry is needed, similar to clustering approaches for radiosity [Sillion et al. 1995; Willmott et al. 1999; Holzschuch et al. 2000]. Note that even without clustering, our system will perform at least as well as point sampling.

8.1 Conclusion

In this paper, we have described a novel caching and interpolation scheme for global illumination that provides interactivity even when per pixel shading costs are high. Spatial coherence is exploited in the reconstruction of a single image from a sparse set of shaded samples and temporal coherence is exploited by caching shading values and reusing them for several frames. While our system is useful in walkthrough applications, our biggest contribution is in the area of interactive update of global illumination after drastic changes to the illumination in the scene, such as when moving the light sources. As a result, we believe that our system will be very useful in applications such as lighting design and modelling.

Acknowledgements

Thanks to Philip Dutré, Steve Westin, Randy Fernando and the anonymous reviewers for their comments, and to Jeremy Selan for helping with the Poolhall model. This work was supported by the NSF Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-8920219) and MRA parallel global illumination project (ASC-9523483), and performed using equipment generously donated by the Intel Corporation and NVIDIA Corporation.

References

AUPPERLE, L., AND HANRAHAN, P. 1993. A hierarchical illumination algorithm for surfaces with glossy reflection. *Proceedings of SIGGRAPH 93* (August), 155–162. ISBN 0-201-58889-7. Held in Anaheim, California.

BALA, K., DORSEY, J., AND TELLER, S. 1999. Radiance interpolants for accelerated bounded-error ray tracing. *ACM Transactions on Graphics 18*, 3 (July), 213–256. ISSN 0730-0301.

BASTOS, R., HOFF, K., WYNN, W., AND LASTRA, A. 1999. Increased photorealism for interactive architectural walkthroughs. *1999 ACM Symposium on Interactive 3D Graphics* (April), 183–190. ISBN 1-58113-082-1.

BISHOP, G., FUCHS, H., MCMILLAN, L., AND ZAGIER, E. J. S. 1994. Frameless rendering: Double buffering considered harmful. *Proceedings of SIGGRAPH 94* (July), 175–176. ISBN 0-89791-667-0. Held in Orlando, Florida.

CHEN, S. E., RUSHMEIER, H. E., MILLER, G., AND TURNER, D. 1991. A progressive multi-pass method for global illumination. *Computer Graphics (Proceedings of SIGGRAPH 91) 25*, 4 (July), 165–174. ISBN 0-201-56291-X. Held in Las Vegas, Nevada.

CHEN, S. E. 1990. Incremental radiosity: An extension of progressive radiosity to an interactive image synthesis system. In *Computer Graphics (Proceedings of SIGGRAPH 90)*, vol. 24, 135–144. ISBN 0-201-50933-4.

COOK, R. L., CARPENTER, L., AND CATMULL, E. 1987. The reyes image rendering architecture. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, no. 4, 95–102.

DARSA, L., AND COSTA, B. 1996. Multiresolution representation and reconstruction of adaptively sampled images. *Proceedings of SIBGRAPI 96* (October), 321–328.

DIEFENBACH, P. J., AND BADLER, N. I. 1997. Multi-pass pipeline rendering: Realism for dynamic environments. *1997 Symposium on Interactive 3D Graphics* (April), 59–70. ISBN 0-89791-884-3.

DRETTAKIS, G., AND SILLION, F. X. 1997. Interactive update of global illumination using a line-space hierarchy. *Proceedings of SIGGRAPH 97* (August), 57–64. ISBN 0-89791-896-7. Held in Los Angeles, California.

DUMONT, R., PELLACINI, F., AND FERWERDA, J. A. 2001. A perceptually-based texture caching algorithm for hardware-based rendering. In *Rendering Techniques 2001: 12th Eurographics Workshop on Rendering*, Eurographics, 249–256. ISBN 3-211-83709-4.

GORAL, C. M., TORRANCE, K. E., GREENBERG, D. P., AND BATTAILE, B. 1984. Modelling the interaction of light between diffuse surfaces. *Computer Graphics (Proceedings of SIGGRAPH 84) 18*, 3 (July), 213–222. Held in Minneapolis, Minnesota.

GRANIER, X., AND DRETTAKIS, G. 2001. Incremental updates for rapid glossy global illumination. *Computer Graphics Forum 20*, 3, 268–277. ISSN 1067-7055.

GUO, B. 1998. Progressive radiance evaluation using directional coherence maps. *Proceedings of SIGGRAPH 98* (July), 255–266. ISBN 0-89791-999-8. Held in Orlando, Florida.

HALTON, J., AND WELLER, G. 1964. Algorithm 247: Radical inverse quasi-random point sequence. *Comm. ACM*, 701–702.

HANRAHAN, P., SALZMAN, D., AND AUPPERLE, L. 1991. A rapid hierarchical radiosity algorithm. *Computer Graphics (Proceedings of SIGGRAPH 91) 25*, 4 (July), 197–206. ISBN 0-201-56291-X. Held in Las Vegas, Nevada.

HART, D., DUTRÉ, P., AND GREENBERG, D. P. 1999. Direct illumination with lazy visibility evaluation. *Proceedings of SIGGRAPH 99* (August), 147–154. ISBN 0-20148-560-5. Held in Los Angeles, California.

HOLZSCHUCH, N., CUNY, F., AND ALONSO, L. 2000. Wavelet radiosity on arbitrary planar surfaces. In *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering*, Eurographics, 161–172. ISBN 3-211-83535-0.

JENSEN, H. W. 1996. Global illumination using photon maps. *Eurographics Rendering Workshop 1996* (June), 21–30. ISBN 3-211-82883-4. Held in Porto, Portugal.

KAJIYA, J. T. 1986. The rendering equation. *Computer Graphics (Proceedings of SIGGRAPH 86) 20*, 4 (August), 143–150. Held in Dallas, Texas.

LARSON, G. W., AND SIMMONS, M. 1999. The holodeck ray cache: An interactive rendering system for global illumination in non-diffuse environments. *ACM Transactions on Graphics 18*, 4 (October), 361–368. ISSN 0730-0301.

LINDHOLM, E., KILGARD, M. J., AND MORETON, H. 2001. A user-programmable vertex engine. *Proceedings of SIGGRAPH 2001* (August), 149–158. ISBN 1-58113-292-1.

MCCOOL, M. D., ANG, J., AND AHMAD, A. 2001. Homomorphic factorization of brdfs for high-performance rendering. *Proceedings of SIGGRAPH 2001* (August), 171–178. ISBN 1-58113-292-1.

MYSZKOWSKI, K., TAWARA, T., AKAMINE, H., AND SEIDEL, H.-P. 2001. Perception-guided global illumination solution for animation rendering. *Proceedings of SIGGRAPH 2001* (August), 221–230. ISBN 1-58113-292-1.

PARKER, S., MARTIN, W., SLOAN, P.-P. J., SHIRLEY, P., SMITS, B., AND HANSEN, C. 1999. Interactive ray tracing. *1999 ACM Symposium on Interactive 3D Graphics* (April), 119–126. ISBN 1-58113-082-1.

PELLACINI, F., FERWERDA, J. A., AND GREENBERG, D. P. 2000. Toward a psychophysically-based light reflection model for image synthesis. *Proceedings of SIGGRAPH 2000* (July), 55–64. ISBN 1-58113-208-5.

PIGHIN, F. P., LISCHINSKI, D., AND SALESIN, D. 1997. Progressive previewing of ray-traced images using image plane discontinuity meshing. *Eurographics Rendering Workshop 1997* (June), 115–126. ISBN 3-211-83001-4. Held in St. Etienne, France.

RAMASUBRAMANIAN, M., PATTANAIK, S. N., AND GREENBERG, D. P. 1999. A perceptually based physical error metric for realistic image synthesis. *Proceedings of SIGGRAPH 99* (August), 73–82. ISBN 0-20148-560-5. Held in Los Angeles, California.

SCHAUFLER, G., DORSEY, J., DECORET, X., AND SILLION, F. X. 2000. Conservative volumetric visibility with occluder fusion. *Proceedings of SIGGRAPH 2000* (July), 229–238. ISBN 1-58113-208-5.

SEGAL, M., KOROBKIN, C., VAN WIDENFELT, R., FORAN, J., AND HAEBERLI, P. E. 1992. Fast shadows and lighting effects using texture mapping. *Computer Graphics (Proceedings of SIGGRAPH 92) 26*, 2 (July), 249–252. ISBN 0-201-51585-7. Held in Chicago, Illinois.

SILLION, F. X., DRETTAKIS, G., AND SOLER, C. 1995. A clustering algorithm for radiance calculation in general environments. *Eurographics Rendering Workshop 1995* (June), 196–205. Held in Dublin, Ireland.

SIMMONS, M., AND SÉQUIN, C. H. 2000. Tapestry: A dynamic mesh-based display representation for interactive rendering. *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering* (June), 329–340. ISBN 3-211-83535-0.

SMITS, B. E., ARVO, J. R., AND SALESIN, D. H. 1992. An importance-driven radiosity algorithm. *Computer Graphics (Proceedings of SIGGRAPH 92) 26*, 2 (July), 273–282. ISBN 0-201-51585-7. Held in Chicago, Illinois.

STAMMINGER, M., SCHEEL, A., GRANIER, X., PEREZ-CAZORLA, F., DRETTAKIS, G., AND SILLION, F. X. 1999. Efficient glossy global illumination with interactive viewing. *Graphics Interface '99* (June), 50–57. ISBN 1-55860-632-7.

STAMMINGER, M., HABER, J., SCHIRMACHER, H., AND SEIDEL, H.-P. 2000. Walkthroughs with corrective texturing. *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering* (June), 377–390. ISBN 3-211-83535-0.

STÜRZLINGER, W., AND BASTOS, R. 1997. Interactive rendering of globally illuminated glossy scenes. *Eurographics Rendering Workshop 1997* (June), 93–102. ISBN 3-211-83001-4. Held in St. Etienne, France.

TELLER, S., BALA, K., AND DORSEY, J. 1996. Conservative radiance interpolants for ray tracing. *Eurographics Rendering Workshop 1996* (June), 257–268. ISBN 3-211-82883-4. Held in Porto, Portugal.

TUMBLIN, J., HODGINS, J. K., AND GUENTER, B. K. 1999. Two methods for display of high contrast images. 56–94. ISSN 0730-0301.

UDESHI, T., AND HANSEN, C. 1999. Towards interactive, photorealistic rendering of indoor scenes: A hybrid approach. *Eurographics Rendering Workshop 1999* (June). Held in Granada, Spain.

VEACH, E., AND GUIBAS, L. 1994. Bidirectional estimators for light transport. In *Fifth Eurographics Workshop on Rendering*, 147–162.

WALD, I., SLUSALLEK, P., BENTHIN, C., AND WAGNER, M. 2001. Interactive rendering with coherent ray tracing. *Computer Graphics Forum 20*, 3, 153–164. ISSN 1067-7055.

WALTER, B., ALPPAY, G., LAFORTUNE, E. P. F., FERNANDEZ, S., AND GREENBERG, D. P. 1997. Fitting virtual lights for non-diffuse walkthroughs. *Proceedings of SIGGRAPH 97* (August), 45–48. ISBN 0-89791-896-7. Held in Los Angeles, California.

WALTER, B., HUBBARD, P. M., SHIRLEY, P., AND GREENBERG, D. F. 1997. Global illumination using local linear density estimation. *ACM Transactions on Graphics 16*, 3 (July), 217–259. ISSN 0730-0301.

WALTER, B., DRETTAKIS, G., AND PARKER, S. 1999. Interactive rendering using the render cache. *Eurographics Rendering Workshop 1999* (June). Held in Granada, Spain.

WARD, G. J. 1994. The radiance lighting simulation and rendering system. *Proceedings of SIGGRAPH 94* (July), 459–472. ISBN 0-89791-667-0. Held in Orlando, Florida.

WHITTED, T. 1980. An improved illumination model for shaded display. *Communications of the ACM 23*, 6 (June), 343–349.

WILLMOTT, A., HECKBERT, P. S., AND GARLAND, M. 1999. Face cluster radiosity. In *Eurographics Rendering Workshop 1999*, Springer Wein / Eurographics, Granada, Spain.

YEE, H., PATTANAIK, S., AND GREENBERG, D. P. 2001. Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments. *ACM Transactions on Graphics 20*, 1 (January), 39–65. ISSN 0730-0301.



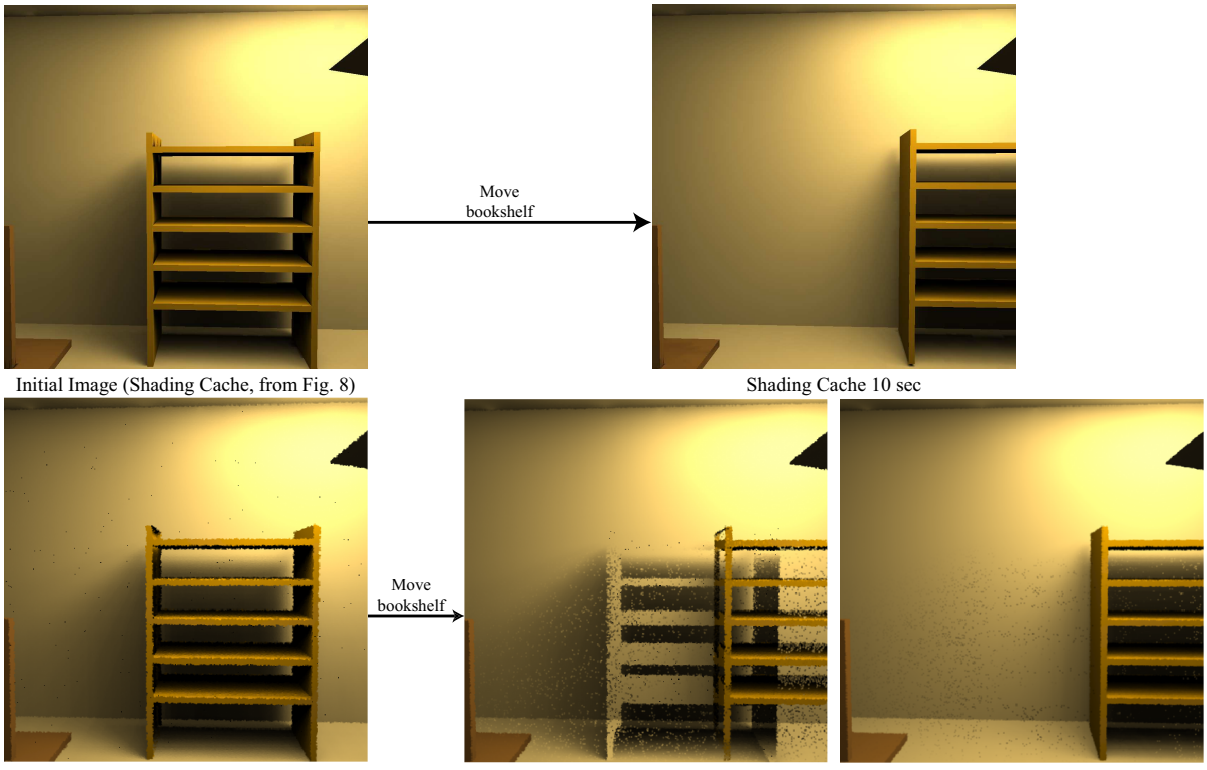
Shading Cache 15 sec | RenderCache 15 sec | RenderCache 180 sec

Figure 7: Equal time and equal quality comparison between our system and the RenderCache for glossy reflections



Shading Cache 10 sec | Render Cache 10 sec | Render Cache 120 sec

Figure 8: Equal time and equal quality comparison between our system and the RenderCache for diffuse inter-reflections



Initial Image (Shading Cache, from Fig. 8) | Shading Cache 10 sec

Initial Image (Render Cache, from Fig. 8) | Render Cache 10 sec | Render Cache 360 sec

Figure 9: Equal time and equal quality comparison between our system and the RenderCache in dynamic scenes