Fast realistic lighting for video games

A.Iones*, A. Krupkin*, M. Sbert**, S. Zhukov***

*Saber Interactive, NY, NY
**University of Girona (Spain)
***Creat Studio (St.Petersburg, Russia)
{iones, krupkin}@saber3d.com www.saber3d.com
mateu@ima.udg.es
s.zhukov@creatstudio.com www.creatstudio.com

Abstract. Global lighting effects produced by diffuse interreflections are typically simulated using global illumination methods such as radiosity or ray tracing. Although diffuse interreflections are crucial to produce realistic images, radiosity-like methods are rarely used in production rendering because of slow performance, robustness problems and difficulty-to-control.

We present a novel technology that produces natural-looking lighting effects in a much faster way than radiosity and ray tracing. The solution is view-independent, and can be used in 3D real-time interactive applications, as well as for high-quality production rendering. Our method simulates global illumination solution using the ambient light illumination model based on *obscurances*. The technology outlines surface profile even without light sources, making it easy to rapidly compute realistically looking images. The simulation of ambient light distribution saves both light sources setting and rendering times.

The results of computations are stored in texture maps, which allows storing multiple light samples per polygon without explicit meshing. This is vital in 3D real-time applications, and speeds up rendering in 3D animation packages as well. Once computed, obscurance maps allow for rapid re-computation of the illumination solution for moving light sources and for scenes with animated objects.

Keywords: rendering, illumination effects, radiosity, texture mapping.
Additional keywords: illumination model, lightmaps, ambient light, form-factor, obscurance, production rendering.

1. Introduction

Usually an expensive method like radiosity or ray tracing (or a combination of both) is used to simulate global illumination effects produced by diffuse interreflections. Being powerful and capable of producing photorealistic images, these methods are as of today time- and space-consuming, and hard for a non-expert to control. For this reason, radiosity-like global illumination methods are rarely used in production rendering [Apod98]. In practice, moreover, it is frequently required to render images that are not photorealistic but rather "bend and shape the reality to the will of the [movie] director" [Apod98]. Such an approach works well for filmmaking industry, commercials, game production, and VR systems. In such applications lighting effects must be reproduced quickly, thus using the power but avoiding the expense of radiosity computations.

In this paper we present a novel technology that simulates naturally looking lighting effects. This technology can be effectively used in a similar way both in 3D real-time applications and production high quality rendering. Our method simulates global illumination effects using the ambient light illumination model [Zhukov98b]. This model estimates ambient light more accurately than Phong and other local illumination models but without using the expense of radiosity. The model utilizes the so-called *obscurance coefficients* of scene points to estimate ambient light. *Obscurance*, first introduced in [Zhukov98b], is a local purely geometric property of a scene point that resembles the integrated weighted form-factors computed for some neighborhood of a given point. Being empiric, our approach helps to rapidly compute the illumination solution and render believable and realistic looking images even for the most challenging scenes.

The illumination computation results are stored in special lightmap or obscurance map textures. These textures are used to modulate the base textures of the scene. Storing illumination data in texture maps enables visualization of fine shading details without explicit meshing, which results in much faster rendering.

The technology development is motivated by practical needs. The proposed solutions are easy to implement and convenient to control for an end-user. All parameters are intuitive and directly result in visual changes of produced images. The algorithms do not require user intervention, do not impose modeling constraints, and work robustly on complex production models that may not be well organized for illumination computations. All the techniques and algorithms described in this paper are implemented and used to complement our production renderers (Softimage and *mental ray*). Our lighting technology is now widely used in most challenging Studio's projects such as 3D video games and computer animations (intros and cut scenes) for Heretic-II, Civilization-II: Call To Power, Wu Tang, Star Trek, etc.

The proposed technology has numerous advantages that result in simplifying and speeding up the production pipeline of both real-time applications and pre-rendered graphics. Some of the benefits are briefly described below.

Since our illumination technique is based on ambient light estimation, it accentuates the geometry of the scene even with the absence of light sources. This helps to rapidly compute believable and easy-to-interpret images, and fewer light sources are needed to obtain the desired visuals. This decreases both light source setting and rendering time of image sequences.

Similarly to radiosity, our lighting solution is view-independent. Once computed, obscurance texture maps make it possible to rapidly reevaluate the illumination solution for moving light sources and for scenes with animated objects. Finally, because of the data locality of computations, the method allows a straightforward parallelization.

The rest of the paper is organized as follows. In section 2 we briefly survey the related previous work. Section 3 covers our techniques of patch generation for the scene. The heart of our technology, the ambient light illumination model, is described in section 3, and compared to other approaches in section 5. The details of practical implementation of our techniques and their impact on the production pipelines are presented in section 6. The overall analysis, conclusions and the directions of future work are given in section 7. In the appendix we present some images generated with the use of the described techniques.

2. Previous work

Our lighting technology is based on two principal issues addressed in the literature. They are methods for illumination computation and storage of computed data for efficient rendering.

2.1. Illumination computation methods

The principal goal of realistic image synthesis is to develop the methods that allow visualization of three-dimensional scenes realistically. To achieve this goal, a number of illumination models have been developed. The first models developed 30 years ago are local illumination models that are easy to implement and fast to compute. Later much more complex global methods were devised that allow the generation of photorealistic-quality imagery [Watt92, Foley92, Glas95]. The latter approaches range from classic radiosity [Goral84] and ray tracing [Whitt80] schemes to more modern approaches [Hanr91, Cohen93, Sill94, Sbert96, Beka98, Jensen95]. The other way to create visually plausible images (that are not photorealistic anymore) is based on the use of more ad hoc techniques such as negative intensity lights, non ray-traced lights, and procedural shaders (in *mental ray®* [Mental96] or RenderMan® [Upst92]). These approaches strive to recreate the most appealing and believable lighting effects and may sacrifice physical realism for this goal. Such methods prove to be especially useful in 3D animation packages for rendering computer-generated imagery where it is frequently desirable to use *photosurrealistic* techniques [Apod98] and in 3D real-time applications.

Several relevant methods for computing images by analyzing local geometry of the scene have been discussed in the literature. An empirical approach called *accessibility shading* was exploited in [Mill94], where similar geometric properties were used to recreate the visual effects of aging of materials. Recently, Martin et al. [Mart98] used *transmittance textures* to speed up hierarchical radiosity computation. Transmittance textures were used to store the information on how a polygon (receiver) is occluded with respect to another polygon (shooter) and used to modulate illumination textures that were computed without visibility information. Castro et al. [Castro01] compute an *extended ambient term* by classifying the surfaces into six classes according to their normals and solving a simplified radiosity system. *Obscurances*, which are the core of our lighting technology and will be developed in detail in this paper, were first introduced in [Zhukov98b], and its use resembles the accessibility shading one. They are stored in obscurance maps (light-maps) and simulate indirect diffuse illumination on a point (patch) through a kind of form-factor computation in the vicinity of the point (patch).

2.2. Illumination solution storage

Since to perform the illumination computation the scene is subdivided into patches, it seems natural to explicitly subdivide the original polygons into smaller ones and assign the computed illumination to their vertices [Glas95]. Then, to render the image, a conventional color interpolation scheme is applied to the polygons. A disadvantage of explicit subdivision is that it significantly increases the overall polygon count incurring considerably higher rendering time and storage requirements. This slowdown might be not so important if one needs to compute a radiosity solution for a static scene and render a few frames. In practice, it is frequently needed to render animation sequences of hundreds or thousands of frames, so the rendering time of already-lit scenes becomes crucial, and hence it is quite undesirable to increase polygon count [Myszk94]. The same situation happens in 3D real-time applications because frame rate is primarily dependent on scene complexity.

One of the approaches to resolve this problem is based on the use of *light map textures (lightmaps)* [Coll94, Soft97, Zhukov97]. These textures are of much lower resolution than the base textures of the scene and each texel of a lightmap corresponds to a patch created at scene subdivision for illumination computation. Therefore, lightmap textures allow storing multiple light samples for large polygons. This is in contrast to explicit scene subdivision with interpolative polygon shading, where color samples are stored in vertices only. Note that for lightmap computation the patches are still generated, but this is done implicitly (virtually) – see the next Section. Once illumination is computed and stored in textures, patches are dropped, and the polygon count is not increased at rendering time. In a related way textures were used to represent the precomputed radiosity [Myszk94] or sampled illumination values [Arvo86][Heck90]. An application of these methods to speed up walkthrough of static scenes was given in [Bast97].

Lightmap textures are used to modulate the base textures of the scene recreating fine lighting details. This process is currently supported by graphics acceleration hardware [D3D98] that makes lightmaps quite useful in 3D real-time applications. An example of lightmapping can be seen, for example, in famous game Quake developed by id Software.

A related technique called *illumination maps* was firstly introduced in [Arvo86][Ward88] in the context of ray tracing to avoid recomputing the diffuse illumination any time the viewpoint is changed. Currently commercially available renderers such as Lightscape® and Inspirer® employ illumination maps. In these systems the scene is explicitly subdivided into smaller polygons and illumination maps are stored in polygon vertices [Kopyl98]. Another approach, first introduced in [Jensen95] is to store illumination information in *photon maps* using kd-trees. Photon map computation algorithms were later developed in [Myszk97][Chris00].

A possible drawback of using lightmaps instead of explicit scene subdivision into polygonal patches is that lightmap texture density is constant over the polygon. In practice, this drawback is unimportant. After all, one may subdivide large polygons into a few smaller ones and use more lightmap texels in polygons with high variance of illumination, thus trading texture memory for image quality.

3. Scene subdivision into patches for lightmapping

Similarly to radiosity, our model generates discrete patches for the scene. This problem is well addressed in the literature [Baum91][Heck92][Sturz92]. Most of these works discussed the scene subdivision problem for storing the illumination results in polygon (patch) vertices. Since we are using texture maps to store illumination data, the subdivision should go in a slightly different manner as described below (see also [Zhukov98a]). We assume that the scene is represented by a collection of triangles.

The main part of scene subdivision into patches is polygon clustering. Each cluster contains a number of polygons, all of which can be mapped into a *2D mapping space* such that the mapping becomes a triangulation. We start with the creation of polygon connectivity data structure for all polygons in the scene, and then perform the breadth-first search algorithm on it to form clusters. A polygon P is added to the cluster if:

1) It is adjacent to some other polygon P' within the cluster.
2) Angle between *normal(P)* and *normal(P')* is less than a certain predefined discontinuity angle. (The notion of *discontinuity angles* is widely used in commercial raytracers; often it is equal to 60^0 [Mental96] [Soft97]).
3) When mapped into the mapping space the polygon P does not intersect any other cluster polygon.
4) It satisfies the *clustering criterion* described below.

One can choose different clustering criteria based on different clustering ideas. Two following approaches proved to be effective:

1) Polygons with similar normals are clustered. The mapping space becomes a plane, and mapping function is the projection onto this plane. When the first polygon is added to the cluster, we select one of World Coordinate System axes with the direction closest to polygon normal. Polygon P is added if it passes discontinuity test with this axis.
2) Polygons covered by a continuous texture mapping are clustered.

Second approach usually yields excellent results provided that the scene is evenly textured. The first approach is best for non-textured scenes. Both methods can be used together and selected based on other heuristics.

Once the polygon clusters are created, they are subdivided into patches. Each cluster has a mapping into 2D mapping space, which is defined by the clustering algorithm. This 2D space is tessellated by a regular grid; each grid cell becomes a patch, and hence a texel on lightmap texture (Figure 1). The density

of the grid is selected according to the desired density of lightmap textures. A patch is defined to be a collection of pieces of polygons in the cluster that are mapped onto a single grid cell. The process is illustrated on Figure 2 for a sphere.

For obscurance and other illumination computations we need to obtain a patch center and the normal at that point. Often a patch has only one face mapped onto the corresponding texel. In this case the patch center is well defined, and patch normal becomes a smoothed surface normal at that point. If there are several faces corresponding to a patch, the patch center is computed as an appropriate average point.

Since the patches are shared along polygon boundaries and the intensities of adjacent texels are filtered after computation, there are no abrupt shading changes along polygon edges and the images look smooth. Another advantage of this approach is that a patch may actually cover a large number of polygons in regions where it is assumed that geometric resolution is higher than the required illumination resolution. Since only a few patches are generated in such regions, the complexity of the illumination algorithm is much lower compared to the case of generating a patch for each polygon. In places were patch resolution is not enough it can be easily increased.

Once all clusters of the scene are subdivided into patches, the generated lightmap parts are packed into larger rectangular textures. Therefore, a single lightmap texture corresponds to a large number of polygon clusters (Figure 1, right). This is unlike other implementations where lightmap textures are created individually for each polygon or for polygon stripes [Myszk94][Bast97]. Although such an approach allows creation of lightmaps more adaptively, it results in inefficient usage of texture memory since only a part of a rectangular texture is covered by useful texels. Besides, in practice it is quite undesirable to have a huge number of small textures from hardware texture management point of view [D3D98].

patch p2

patch p1

Part of texture corresponding to drawn cluster

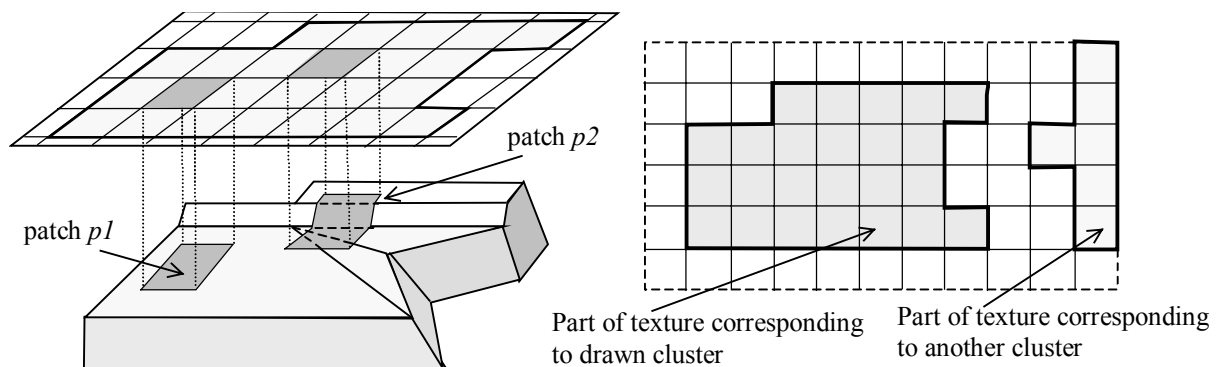Part of texture corresponding to another cluster

Figure 1. Scene subdivision into patches. A part of the lightmap texture corresponding to a cluster of polygons is shown on the mapping plane (left) and on the final lightmap texture (right).
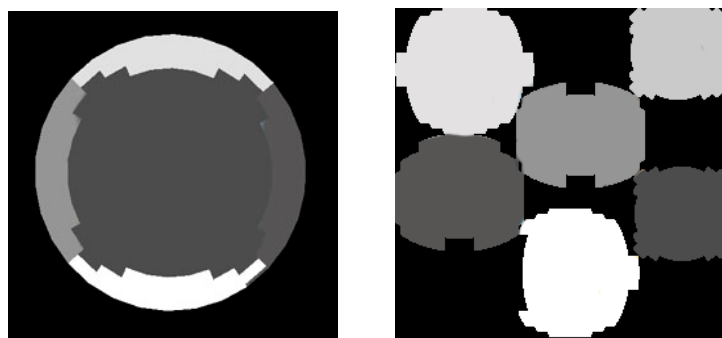

Figure 2. Sphere subdivision into patches using first criteria. Sphere mapped into a part of the lightmap texture corresponding to a cluster of polygons is shown on the mapping plane (left) and on the final lightmap texture (right).

4. Ambient light illumination model

4.1. Intuition leading to the obscurance illumination model

Most local illumination models account for diffuse interreflections using ambient light that is usually crudely approximated to be constant over the whole scene. In reality, in diffusely illuminated scenes composed of mostly diffuse surfaces the illumination is not constant. E.g., in a diffusely lit room the illumination changes over wall surfaces - it is darker near room corners. A similar effect is a shadow under the car in a cloudy day. Such lighting (darkening) effects in the obscured areas can be visually good approximated without the computational expense of the radiosity method. Our illumination model, based

on the notion of *obscurance*, emphasizes corners yielding images that look realistic and easier to interpret. It must be emphasized here that, unlike radiosity, our model is a non-photorrealistic one, but obtains very visually pleasant images.

Roughly speaking, *obscurance* measures the part of the hemisphere obscured by the neighbor surfaces. E.g., near a corner of a room the obscurance of the patches is higher than in the central region. From the physics of light transport point of view, obscurance expresses the *lack* of secondary (reflected) light rays coming to the specific parts of the scene thus making them darker. This is unlike radiosity where secondary reflections are accounted to increase the intensity.

4.2. Empirical obscurance indirect illumination model (without light sources)

Let us assume first that there are no specific light sources in the scene, in other words, perfectly diffuse light coming from everywhere illuminates the scene. The intensity of the ambient light for the whole environment is a direction-independent constant I_A (this is the same as the one used in other local illumination models). We introduce the obscurances to avoid the disadvantages of classic ambient term. The basic idea of this new direction and distance dependent visually pleasant ambient term effect is to use a function ρ with values between 0 and 1 to represent the incoming ambient light.

Let P be a point on the surface in the scene, ω a direction in the normal hemisphere Ω with center P, aligned with the surface normal at P and lying in the outer part of the surface (Fig.3 left). We look for a function $\rho(L(P,\omega))$ to represent the *obscurance* [1] for a point P in direction ω, this is, the magnitude of ambient light incoming from direction ω. L is the distance to the first intersection in this direction. This function should have the (intuitive) following properties:

$$\rho(L) = \begin{cases} 0, for\ L = 0 \\ 1, for\ L = +\infty \end{cases} \qquad \rho'(L) = \begin{cases} > 0 \\ 0, for\ L = +\infty \end{cases} \qquad \rho''(L) < 0 \qquad (1)$$

That is, we look for a monotonous increasing smooth function (the farthest the intersection the higher is the contribution of the ambient term in this direction), null at intersection distance zero (fully occluded) and asymptotical value 1, and monotonous decreasing derivative (see Fig.3 right).

An evident family of functions that fits (1) is:

$$\rho(L) = (1 - e^{-\tau L}) \qquad (2)$$

where τ is a positive parameter.

Obscurance of the point P is now defined as follows:

$$W(P) = \frac{1}{\pi} \int_{\omega \in \Omega} \rho(L(P,\omega)) \cos\theta d\omega . \qquad (3)$$

Thus, the obscurance of a point P is the weighted average length of a chord originating from the point (the length of the chord is measured between P and the first intersection point with a surface in the scene). Clearly, for any surface point $P: 0 \leq W(P) \leq 1$. E.g., if P stands in the midpoint of the base of a hemisphere with radius R then $W(P) = \rho(R)$. The function $W(P)$ reflects the local geometric properties of point P. Obscurance value 1 means that the point is fully open, while a value of 0 means the point is fully closed (this may happen only in the degenerate cases). Now, we assume that the more a point is open, the greater its intensity (brightness). This reflects the fact that normally ambient lighting of a given point is primarily affected by its neighborhood. This is especially true for scenes without bright light sources that may affect the illumination at large distances. We write then for the reflected intensity at point P:

$$I(P) = I_A k_A(P) W(P) , \qquad (4)$$

where $k_A(P)$ is the diffuse reflectance for ambient light[2].

Observe that that if point P is totally un-obscured ($\rho()\equiv 1$ over the whole hemisphere) then $I(P)$ becomes $k_A * I_A$.

---

[1] The name 'obscurance' is a bit confusing, since it actually denotes the 'openness' of a point, but we prefer to stick to the earlier introduced name [Zhukov98b].

[2] Interestingly the obscurance model with $\rho = (1 - e^{-\tau L})$ function can be interpreted as the illumination at the non-reflecting boundaries of a non-scattering gas with opacity τ and constant volume emittance I_A [Arvo93].
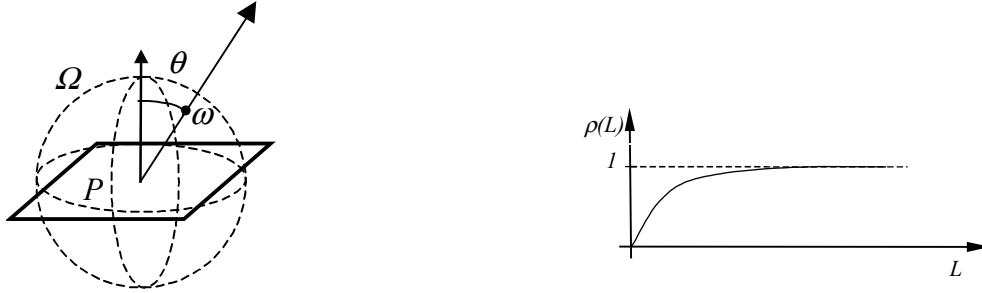
Figure 3. Left: Illustration of the notation in (4). Right: A plot of the function ρ(L)

The equation (4) actually determines the illumination model working in the absence of light sources. The obscurance of a patch is defined as the average obscurance value over all its points. Typical obscurance values for different patches are illustrated on Figure 4. In section 4.5 it will be shown that patch obscurance eventually resembles patch's average form-factor.

On Figure 4 we show the obscurance map for a *'Temple'* scene. Assuming all reflectances are equal, the illumination of this scene by ambient light (without specific light sources) using equation (4) gives the same image (up to a constant factor). Notice that a number of pseudo-shadow effects are clearly visible. We claim that the scene looks similar to the view on a cloudy day. The use of obscurances outlines the surface profile without any light source setting.

Obscurance is an intrinsically geometric property - it does not depend on light sources, so for a given scene obscurance maps need to be computed only once and stored for future use to recompute the lighting for moving light sources.

Figure 4. Obscurance map for 'Temple' scene.

4.3. Ambient light illumination model accounting for light sources

The goal of our illumination model is to account for indirect illumination caused by diffuse interreflections in an easy and fast manner. To compute the illumination of a given patch caused by a given light source, we separate the illumination into the direct and indirect factors. We compute the direct illumination using the standard local diffuse illumination model. Similarly to ambient light, the indirect illumination from the light sources is empirically accounted for with the help of the obscurance coefficient. Specifically, we use the following equation (the extension of equation (4)):

$$I(P) = (I_A + I'_S(P)) * k_A(P) * W(P) + k_D(P) * I_S(P), \qquad (5)$$

where:

k_D - diffuse reflection coefficient;

I_S - intensity of direct illumination from all *visible* light sources,

$$I_S(P) = \sum_{\substack{by\ all\ visible \\ light\ sources}} \frac{I_j}{r_j^2} \cos\alpha_j\ ; \qquad (6)$$

I'_S - intensity of indirect illumination coming from *all* light sources,

$$I'_S(P) = \beta * \sum_{\substack{by\ all\ light \\ sources}} \frac{I_j}{r_j^2};$$ (7)

where:

I_j - the intensity of *j-th* light source;

r_j - the distance from light source *j* to patch *P*;

α_j - the angle between direction toward the *j-th* light source and patch normal;

β - a constant, $0 < \beta < 1$.

Indirect illumination from light sources is accounted for by the term I'_S. The constant β scales down the indirect component with respect to the direct one. The indirect term I'_S does not contain *cos()* factors since we assume that secondary rays come to a patch from every direction. Note that all light sources are accounted for regardless of whether they are visible or not. Obviously, not rejecting invisible light sources we may get artifacts like lighting through an opaque wall. On the other hand, we properly account for a light source in a patch that is just locally obscured by other surfaces.

On Figure 5 (left) we show a fragment of the same *'Temple'* scene rendered using the equation (5) (with one light source). Note that although projected shadows appeared, the overall impression does not differ much from the scene rendered with only obscurances. On Figure 5 (right) the same scene is rendered using conventional first order ray tracing with constant ambient light. Note that the geometry of the scene cannot be properly discerned and, therefore, the image is hard to interpret – additional light sources are needed to outline the profiles everywhere in the picture. Other ad hoc techniques such as using local lights having negative intensity that actually dim the scene can be used for such purposes, but these approaches are more expensive compared to the use of obscurances.

The proposed technique allows the use of significantly fewer light sources to obtain the desired lighting effects, thus decreasing both light source setting and ray tracing time.
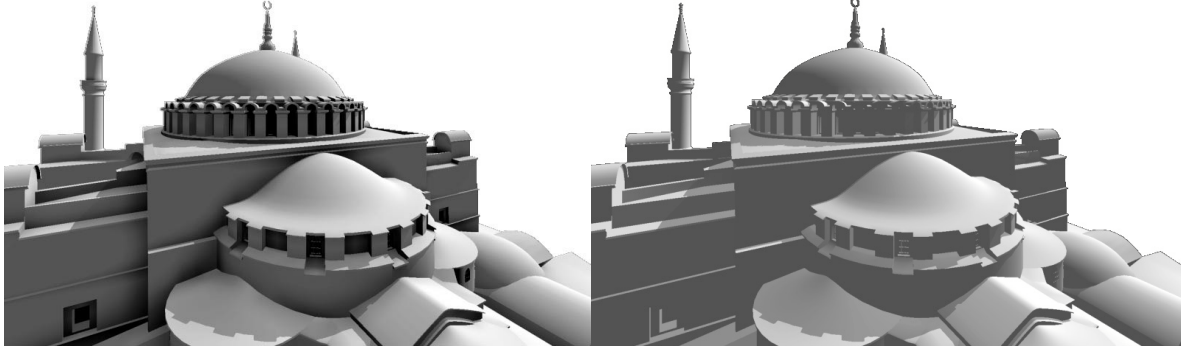

Figure 5. 'Temple' scene with one directional light source rendered using equation (5) (left). Note that apart from pseudo-shadows projected shadows appeared. On the right the same scene is rendered with first order ray tracing with constant ambient light.

4.4. Limiting the maximum distance.

From a practical point of view we take in consideration only an L_{max} –environment of the patch, where is a scene dependent parameter. This represents a compromise between taking into account all occlusions and the efficiency of the method, as we have only to consider for the computation of (2) a fraction of the scene. Function $\rho(L)$ becomes then

$$\rho(L) = \begin{cases} 0, for\ L = 0 \\ 1, for\ L \geq L_{max} \end{cases}$$ (8)

Experiments with different kinds of scenes have shown that $\tau=1$ is a good choice in (2), and thus L_{max} is the primary parameter that controls the illumination model. Next we discuss some issues about calibration of L_{max}. Assume that we need to compute the illumination by ambient light for the scene composed of two perpendicular half-planes (Figure 6). This task cannot be resolved because there are no 'meters' in such a scene. So for practical purposes the size of the shadowed regions near the corner depends on the selection of the constant L_{max} that should be chosen with respect to the size of the objects that are relevant to the scene (e.g., avatars that will inhabit the scene in a VR application). Therefore L_{max} controls the relative size of the shadowing effect. Assuming that a patch is several times smaller than a relevant object, L_{max} is usually chosen to cover 10-100 patches.
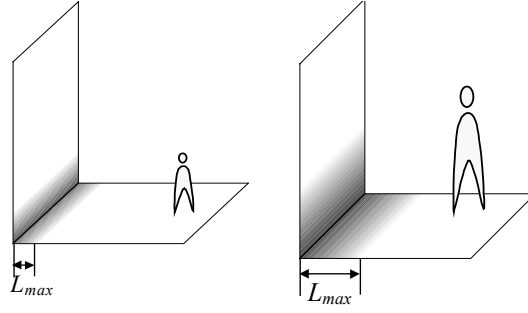
Figure 6. L_{max} selection depending on the width of the desired shadow region. Even though the left and right images are different, both of them are equally acceptable depending on the size of other relevant objects.

In practice L_{max} can be selected rather loosely within certain reasonable limits. Once an obscurance texture map is computed, its contrast and gamma can be edited in an image processing tool that effectively results in changing shadow region shapes and sizes that are otherwise controlled by L_{max}.

4.5. Computing the obscurances

Obscurance coefficients are similar to integrated weighted form factors. Indeed, for a point Q visible from point P on a given patch p, the differential form-factor dF_{PQ} is defined as:

$$dF_{PQ} = \frac{1}{\pi r^2} \cos\theta_P \cos\theta_Q dS_Q dS_P , \qquad\qquad (9)$$

where:

θ_P, θ_Q - angles between the line PQ and respective patch normals
dS_Q - differential area at Q
r - distance between P and Q

Since $d\omega = \dfrac{1}{r^2}\cos\theta_Q dS_Q$, using the definition of obscurance of a point (3), we obtain for the obscurance of point P:

$$W(P) = \int_{\substack{over\ all\ visible\ surface\ po\mathrm{int}\ s \\ (differential\ patches\ Q)}} \rho(dist(P,Q))dF_{PQ} \qquad\qquad (10)$$

and taking the average over the area of patch p we obtain the obscurance of p.

This last equation indicates the computation method for the obscurance of a given patch. To compute the obscurance, one may use, e.g., the hemicube method. In this case we place a single hemicube over the patch center, this is, we approximate the obscurance of a patch with the obscurance of the patch's center. Since the function $\rho(L)$ is less than 1 only in the range $[0..L_{max}]$, we may use only patches in L_{max}-neighborhood of P. A slight modification of the hemicube method is needed then to take into account 'missing patches' (initializing pixels to 0) and weight differential form-factors using $\rho(L)$ (storing weighted with $\rho(L)$ values instead of differential form-factors). The locality of the obscurance method makes it much faster than the conventional radiosity. Other form factor computation techniques, such as Monte Carlo (see for instance [Keller96]), can be applied to compute obscurances as well.

5. Analysis of the ambient light illumination model

We consider here the cost and complexity of obscurance computation using the hemicube approach, the technique used in our implementation. Obviously, other algorithms to compute obscurances may exhibit different cost/complexity. Now, suppose we have fixed the hemicube resolution and the parameter L_{max}. Let the number of patches and polygons in the scene be n_p and n_s, respectively. Following [Sbert97], we distinguish two different complexities.

To compute obscurance for a patch, all polygons in L_{max}-neighborhood must be projected onto the hemicube. If we consider this neighborhood defined by a constant fraction k of scene volume (L_{max} is the same for all patches), then always the same fraction of scene polygons is projected. This makes a cost of $O(k*n_p*n_s)$, with k very small. Now, if we increase the resolution of patches without increasing the number of polygons, the cost will grow as $O(n_p)$. However, if we increase the number of polygons retaining the same patch resolution, the cost will grow as $O(n_p*n_p)$, but the constant is small. This smallness implies that roughly linear radiosity methods like Hierarchical Radiosity with clustering [Smits94][Sill95] are outperformed in fairly big scenes. Despite its lower asymptotic complexity, Hierarchical Radiosity is simply not feasible to tackle scenes with millions of patches.

On the other hand, it is not fair to compare a pure geometrical quantity computation with the determination of the density of flux of energy (radiosity). We should compare obscurance computation

with its peer, form factor computation. Obscurances retain from form factors the capability to generate nice soft shadows, and do not need to be recomputed under a change in the illumination settings. And clearly obscurance computation outperforms form factor one, with no need to mention the quadratical order in form factor storage, while the memory required for obscurance maps is linear relative to the number of patches. For the scenes used in 3D interactive applications of roughly 20.000 polygons such as the ones shown on Fig.11-12, the typical memory requirements for storage of all lightmap textures are about five 256x256 textures.

5.1. Comparison with other methods

The initial motivation in developing the described ambient light illumination model was the need to compute the illumination of scenes used in our real-time 3D games where the scene complexity is up to 20,000 polygons (see shots in appendix). Currently our lighting tool is capable of handling complex scenes composed of up to 400,000 polygons subdivided into a million of implicit patches. These scenes were created in-house using Softimage® for our animation projects (see appendix).

To compare the performance of our method with other approaches, we computed the illumination of common Peter Shirley's scenes as seen in the figures below. On Figure 7 we show the test scenes computed using our method.
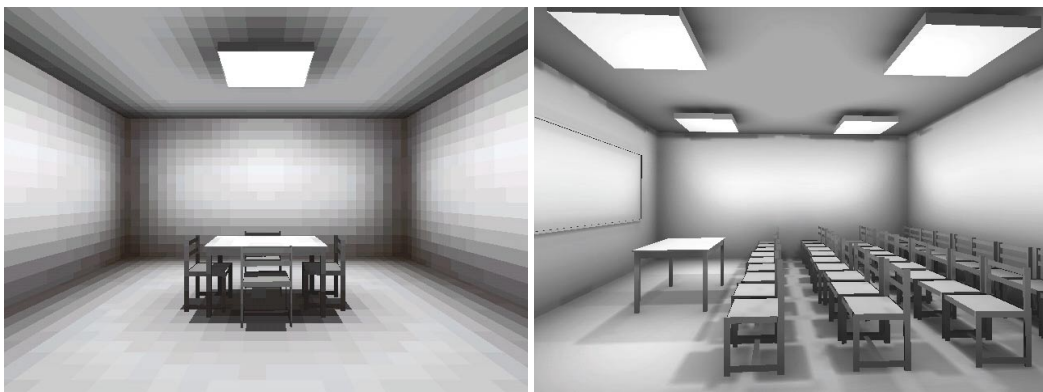

Figure 7. Shirley's scenes computed with obscurance method

The comparison of our method with other Hierarchical Radiosity (HR), Hierarchical Monte Carlo Radiosity (HMC) [Beka98,99], multipath Monte Carlo radiosity (MC) [Sbert96] are presented in Tables below. The timings are given for Pentium-200 or similar machines. Note that the images produced using HMC/MC radiosity methods with the timings given are noisy. To obtain high-quality images, the timings must be significantly increased since the methods converge slowly. Observe that obscurance timings in both scenes correspond roughly to $O(k*n_p*n_s)$ complexity. The same behaviour can be observed between scenes in Fig.13&14 (see Appendix). Apparent differences in this behaviour, such as between outdoor (Fig.11&12) and indoor (Fig.13&14) scenes are caused by a different selection of L_{max} value which in turn gives a different k value.

Table1. 'Table and chair' (180 polygons).

Method	# of patches	Time (sec)
HR	23000	480
HMC	1777	20
	17617	320
MC	12000	1100
Obscurance	2000	1
	15000	6

Table2. 'Classroom' (1244 polygons).

Method	# of patches	Time (sec)
HR	18000	450
	60000	Ran out of memory (>64Mb)
HMC	77202	570
Obscurance	60000	240

We do not make comparisons with a radiosity method in terms of accuracy. Indeed, the ultimate goal of our model is to obtain visually pleasant realistic like lighting effects, not solving the radiosity system. The primary justification of its utility is that the images produced look believable and realistic. Nevertheless, obscurances mimic at its best radiosity when light sources are positioned near the ceiling and are visible from most of the scene (see Fig.8). On the other side, when the light source points against and is near i.e. a wall, this is, when there is a very important secondary reflector, obscurances poorly imitate radiosity (see Fig.9). Considering these important secondary reflectors as additional light sources like in progressive radiosity [Cohen88] could solve this problem.

Figure 8. Cornell box scene computed with obscurance method(left) and shooting random walk radiosity [Beka99] (right)
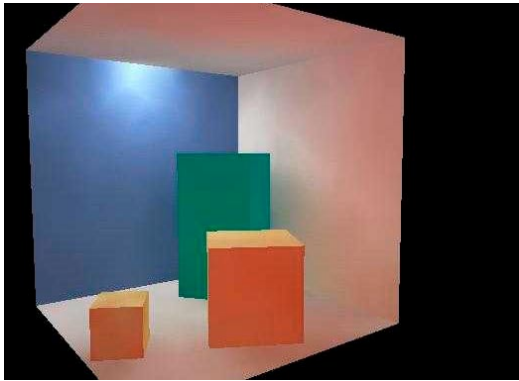

Figure 9. Cornell box scene with light pointing to a wall. Computed with obscurance method(left) and Hierarchical Monte Carlo radiosity [Beka98] (right). The sources are not shown in the images.

6. Implementation

In our implementation, we developed a standalone lighting tool that is used to compute obscurances or a full illumination solution for static scenes. The results of the computations are stored in texture maps that are assigned to the given model.

6.1. Use of the technology in 3D real-time applications

If the scene is going to be used in a real-time application, the overall illumination of the scene with light sources is computed and stored in lightmap textures. Then, at run-time, lightmap textures are used to modulate (add to or multiply) the base textures of the scene.
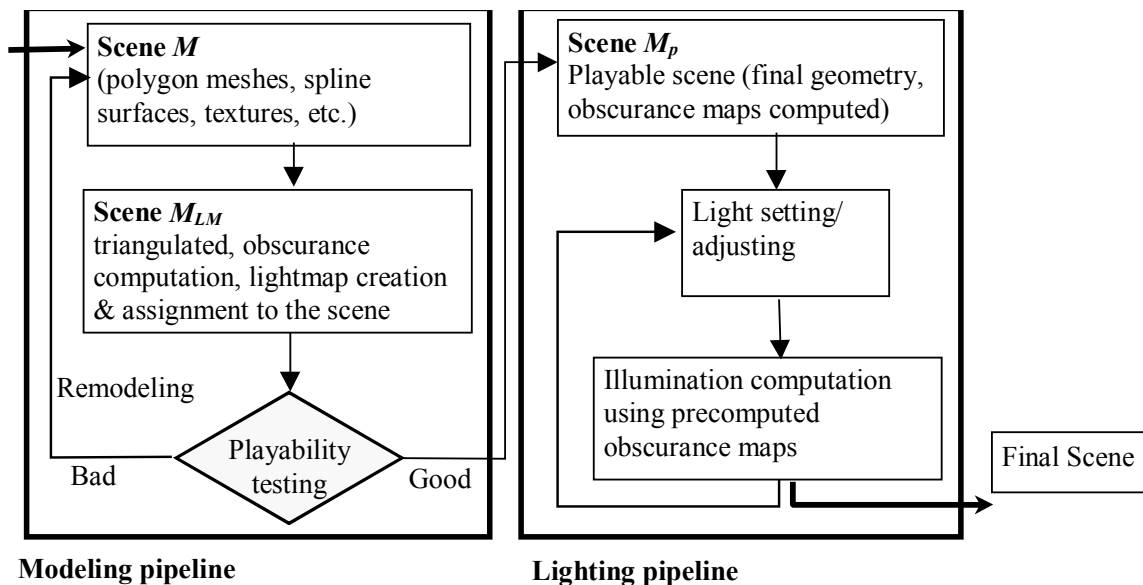

Figure 10. Production pipeline for game environment production

The use of the described technology greatly helps to speed up the production pipeline. We briefly discuss here the technology application for our game development. The game environment development cycle consists of two stages – geometry modeling and lighting (Figure 10). The crucial aspect in geometry modeling for games is that it requires an enormous (in the order of hundreds) number of iterations to achieve and finetune playability. Clearly, it is useless to spend much time texturing an unfinished model, and here our technology comes in handy. We compute the illumination of the scene using ambient light only (without light source setting at all!) and apply the computed lightmaps for the scene. This makes scene geometry easily discernable. That is not the case at all for a preliminarily textured scene (see Fig.13). As a result of the computational effectiveness and usefulness of the technology, obscurance computation became a standard step of model conversion from Softimage into the game format.

Once the geometry is finished, the scene model is textured and light sources must be set. Since illumination computation with precomputed obscurance maps is very fast, a modeler may easily do as many iterations as needed to carefully set and adjust light sources to achieve the desired lighting effects. In practice for scenes containing up to 20,000 polygons (which are used in our real-time applications) this recomputation is done at interactive rates.

6.2. High-quality rendering applications

Let us discuss the task of rendering the animation sequence of a camera walkthrough in a static scene. Our goal is to account for ambient light distribution using the ambient light illumination model in a standard ray tracer rendering pipeline (in our case, Softimage / *mental ray*). For this purpose, we compute the obscurances that are stored in obscurance map textures assigned to the given Softimage model.

In our implementation, we wrote a material shader that describes the visible material of an object [Mental96]. Such shader is executed on each hit of a ray with a surface. Instead of computing a constant ambient light I_A, this shader computes its estimation $I_A*W(P)$, where P is the hit point. Obscurance $W(P)$ is taken from the appropriate obscurance texture map. Implementation of the ambient light illumination model via a material shader is the most accurate.

The use of a standard ray tracer allowed us to concentrate on developing just the code we needed – scene subdivision into patches, the computation of obscurances, efficient storage of the data in texture maps and assignment of the textures to the given model. Standard ray tracer (*mental ray*) takes care of all the important rendering tasks, including ray tracing itself, scene management, antialiasing and much more. It is not a problem to handle objects created with advanced modeling techniques such as spline surfaces, objects with shape animation, etc. Before rendering, such models are usually converted into polygon meshes. We then assign obscurance texture maps to these meshes, thus avoiding artifacts caused by incoherent meshing.

7. Conclusion

In this paper we presented a new lighting simulation technology that is capable of producing realistic lighting effects in a fast and easy way. The technology can be smoothly incorporated into the production pipeline providing an easy-to-use solution. Currently the technology is used in our game development projects, as well as on most of the animation projects at Studio. Among them are intros, outros and cut scenes for Activision's games Heretic-II, Quake-II Ground Zero mission pack, Civilization-II:Call to power and others. The use of our technology significantly improves the image quality and reduces the overall production time on various stages of project development. The key features and advantages of the described technology are listed below.

- Ambient light illumination model simulates non-constant ambient light distribution in the environment without recourse to expensive global illumination methods. Use of the suggested model enables one to account for the indirect illumination from light sources in an easy and rapid manner.
- Being able to accentuate the profile of a scene without a single light source makes it easy to rapidly compute realistically looking images. This is especially important in early stages of the animation project development or when modeling environments for a game when the models are not textured and model's geometry is not distinguishable without lighting.
- The described technique allows use of fewer light sources to obtain the desired image, thus decreasing both rendering time at ray tracing and time spent by a modeler to set and adjust light sources. A realistic-looking illumination and discernable geometry is ensured by the technology.
- The technology is easy to control for an end-user (modeler). Parameter control (L_{max}) and image processing of the obscurance texture maps are intuitive and conventional. Thus (pseudo) shadow effects are easily manageable.
- The technology is easy to implement. The algorithms are straightforward and do not rely on complex data structures. This leads to much lower multiplicative constants in big-Oh bounds both for time and

space complexities that are moderate for obscurance computation algorithm compared to radiosity. The technology does not require user intervention and does not impose modeling constraints. It works robustly on complex production models that may not be well organized for illumination computations.

- Our lighting solution is view-independent. This makes the method useful for real-time applications enabling the visualization of global illumination effects at interactive rates. Rendering an animation sequence, the combination of ray tracing and our solution significantly improves the visual quality. This improvement is relatively inexpensive because of the efficiency of the illumination model used.
- Once the obscurance maps for the given static scene are computed, it is easy to recompute the lighting for moving light sources.
- Similarly, if there are animated objects in the scene, the re-computation of obscurance maps must be carried out only in a neighborhood of the moving objects.
- The method allows a straightforward parallelization due to its data locality.
- The radiosity realistic effect of color bleeding can be easily incorporated in the obscurances method, as shown in [Garcia01]. Fig.8(left)&9(left) illustrate first results of our running research in this direction.

The technology presented in the paper lays the groundwork for future development in different directions. To mention a few, we plan to improve the incorporation of the lighting tool in the production pipeline, thereby effectively performing obscurance recomputations in scenes with animation models. Temporary aliasing artifacts must be properly handled in this scenario.

Since obscurances are similar to form factors, many ideas developed in radiosity research can be applied for obscurance computations as well. Specifically, improved adaptive, hierarchical, clustering or Monte Carlo schemes can be found to speed up computations especially for large L_{max} values.

Overall, realistic lighting simulation technology is perfectly suitable for applications where it is desirable to sacrifice the physical accuracy of the computations, but gain in speed and ease of light source setting, rendering the scenes and simplification of overall production pipeline.

Acknowledgments

References

[Apod98] Apodaca, A.: "Photosurrealism", *Rendering Techniques'98*, Springer-Wien, NewYork (Proc. of Eurographics Rendering Workshop'98 - Vienna, Austria), pp. 315-322

[Arvo86] Arvo, J.: "Backwards Ray Tracing", in ACM SIGGRAPH'86 Course Notes – Developments in Ray Tracing, v. 12, August 1986

[Arvo93] Arvo, J.: "Transfer Equations in Global Illumination", *Global Illumination*, in ACM SIGGRAPH'93 Course Notes, volume 42, August 1993

[Bast97] Bastos, R., Goslin, M., Zhang, H.: "Efficient Radiosity Rendering using Textures and Bicubic Reconstruction", ACM Symposium on Interactive 3D Graphics, 1997, pp. 71-74

[Baum91] Baum, D., Mann, S., Smith, K., Winget, J.: "Making Radiosity Usable: Automatic Preprocessing and Meshing Techniques for the Generation of Accurate Radiosity Solutions", Computer Graphics'91 (SIGGRAPH'91 Conference Proceedings), Vol.25(4), July 1991, pp.51-60

[Beka98] Bekaert, P., Neumann, L, Neumann, A., Sbert., M., Willems, Y.: "Hierarchical Monte Carlo Radiosity", Springer-Wien, NewYork (Proc. of Eurographics Rendering Workshop'98 - Vienna, Austria), pp. 259-268

[Beka99] Bekaert, P,.: "Hierarchical and Sochastic Algorithms for Radiosity", PhD dissertation, Katholieke Universiteit Leuven, 1999

[Castro01] Castro, F., Neumann, L. and Sbert, M., "Extended Ambient Term", Journal of Graphics Tools, 5(4), pp. 1-7, 2000.

[Cohen88] Cohen, M.F., Chen, S.E., Wallace, J.R. and Greenberg, "A Progressive Refinement Approach for Fast Radiosity Image Generation", Computer Graphics 22(4):75-84 (ACM SIGGRAPH'88 Proceedings).

[Cohen93] Cohen M., Wallace J.: "Radiosity and Realistic Image Synthesis", Academic Press Professional, CA, USA, 1993

[Coll94] Collins, S.: "Adaptive Splatting for Specular to Diffuse Light Transport", Proc. of Fifth Eurographics Workshop on Rendering, pp.119-135, Darmstadt, Germany, 1994.

[Chris00] Christensen, P.: "Faster Photon Map Global Illumination", ACM Journal of Graphics Tools, Vol.4(3), 2000, pp. 1-10

[D3D98] Direct3D Programmer's Guide, Microsoft, 1998

[Foley92] Foley J., van Dam A., Feiner S., Hughes J.: "Computer Graphics. Principles and Practice." Addison-Wesley, 1992.

[Garcia01] M.Garcia, M.Sbert. A.Iones, A.Krupkin, "Improved obscurances with color bleeding", Research Report IIiA 01-12-RR, 2001, University of Girona, Girona

[Glass95] Glassner, A. S., Principles of Digital Image Synthesis, Morgan Kaufmann, San Francisco, CA, 1995.

[Goral84] Goral C., et al.: "Modeling the interaction of light between diffuse sources", ACM SIGGRAPH'84 Conf. Proc.

[Hanr91] Hanrahan P., Salzman D., Aupperle L.: "A Rapid Hierarchical Radiosity Algorithm." Computer Graphics'91 (SIGGRAPH'91 Conference Proceedings) Vol.25(4), July 1991, pp.197-206

[Heck90] P. Heckbert: "Adaptive Radiosity Textures for Bidirectional Ray Tracing", Computer Graphics (SIGGRAPH '90 Proceedings), vol. 24, 1990, pp. 145-154

[Heck92] P. Heckbert: "Discontinuity Meshing for Radiosity", *Rendering Techniques'92* (Proc. of Eurographics Rendering Workshop'92 – Bristol, UK), pp. 203-216

[Jensen95] Jensen, H., Christensen, N.: "Photon Maps in Bidirectional Monte Carlo Ray Tracing of Complex Objects", Computers and Graphics 19(2), pp, 215-224, 1995

[Keller96] Keller, A.: "The fast Calculation of Form Factors using Low Discrepancy Point Sequences", Proc. of the 12 Spring Conference on Computer Graphics, Bratislava, 1996, pp. 195 - 204

[Kopyl98] Kopylov, E., Khodulev, A., Volevich, V.: "The Comparison of Illumination Maps Technique in Computer Graphics Software", Proc. of GraphiCon'98 - Moscow, Russia, pp. 146-153

[Mill94] Miller G.: "Efficient Algorithms for Local and Global Accessibility Shading", SIGGRAPH'94 Conference Proceedings, 1994, pp.319-326

[Mart98] I. Martin, X. Pueyo and D. Tost: "A Hardware Based Method for Hierarchical Radiosity", Computers Graphics Forum (Eurographics '98 Proceedings), 17(3):159-164, 1998

[Mental96] Softimage® 3D v. 3.7 "mental ray. Programmer's Guide". Printed in Canada, 1996

[Myszk94] Myszkowski, K., Kunii, T.: "Texture mapping as an alternative for meshing during walkthrough animation", *Rendering Techniques'94*, Springer-Wien, NewYork (Proc. of Eurographics Rendering Workshop'94 – Darmstadt, Germany)

[Myszk97] Myszkowski, K.: "Lighting Reconstruction Using Fast and Adaptive Density Estimation Techniques", *Rendering Techniques'97*, Springer-Wien, NewYork (Proc. of Eurographics Rendering Workshop'97 – St. Etienne, France), pp. 251-262

[Sbert96] Sbert, M. et al.: "Global multipath Monte Carlo algorithms for radiosity", The Visual Computer, 12:47-61, 1996

[Sbert97] Sbert, M.: "Error and Complexity of Random Walk Monte Carlo Radiosity", IEEE Transactions on Visualization and Computer Graphics, v 1(3), 1995, pp. 240-254.

[Smits94] Smits,B., Arvo,J. and Greenberg,D.: "A Clustering Algorithm for Radiosity in Complex Environments", Computer Graphics Proceedings, Annual Conference Series, 1994 (ACM SIGGRAPH '94 Proceedings), pp.435-442.

[Sill94] Sillion F., Puech C.: "Radiosity and Global Illumination." Morgan Kaufmann, SF, US, 1994

[Sill95] Sillion F.: "A Unified Hierarchical Algorithm for Global Illumination with Scattering Volumes and Object Clusters" , IEEE Transactions on Visualization and Computer Graphics, v 3(1), 1997, pp. 23-38

[Soler98] Soler, C., Sillion, F.: "Automatic calculation of soft shadow textures for fast, high quality radiosity", *Rendering Techniques'98*, Springer-Wien, NewYork (Proc. of Eurographics Rendering Workshop'98 - Vienna, Austria), pp. 199-210

[Soft97] Softimage® 3D v. 3.7 "Release Notes". Printed in Canada, 1997

[Sturz92] Sturzlinger, W.: "Radiosity with Voronoi diagrams", Rendering Techniques'92, (Proc. of Eurographics Rendering Workshop'92 - Amsterdam, Holland), pp. 169-177

[Upst92] Upstill, S.: *The RenderMan Companion. A Programmer's Guide to Realistic" Computer Graphics*, Addison-Wesley, 1992

[Ward88] Ward, G, Rubinstein, F, Clear, R: "A Ray Tracing Solution for Diffuse Interreflection", ACM SIGGRAPH '88 Proceedings, vol. 22, No 4, pp. 85-92, 1988

[Watt92] Watt A., Watt M.: "Advanced Animation and Rendering Techniques. Theory and Practice." Addison-Wesley, 1992.

[Whitt80] Whitted T.: "An Improved Illumination Model for Shaded Display", Comm. Of ACM Vol. 6, No. 23, 1980

[Zhukov97] Zhukov, S., Iones, A., Kronin, G.: "On a practical use of light maps in real-time application", in Proc. of SCCG'97 Conference (Bratislava, Slovakia)

[Zhukov98a] Zhukov, S., Iones, A., Kronin, G.: "Using Light Maps to create realistic lighting in real-time applications", in Proc. of WSCG'98 Conference (Plzen, CZ), pp. 464-471

[Zhukov98b] Zhukov, S., Iones, A., Kronin, G.: "An Ambient Light Illumination Model", *Rendering Techniques'98*, Springer-Wien, NewYork (Proc. of Eurographics Rendering Workshop'98 - Vienna, Austria), pp. 45-55

Appendix

To illustrate our technology we include images of some scenes used in actual production. All timings for obscurance computation are given for Pentium-200 machine. All images were produced at CREAT Studio.

Name	Used in	# of polygons	# of patches	Time
Automobile	Civilization-III cut scene	43,000	75,000	9 mins
Temple	Civilization-III cut scene	106,000	95,000	14 mins
Human envir.	Game project	3,000	28,000	45 secs
Alien envi.	Game project	6,000	98,000	5 mins

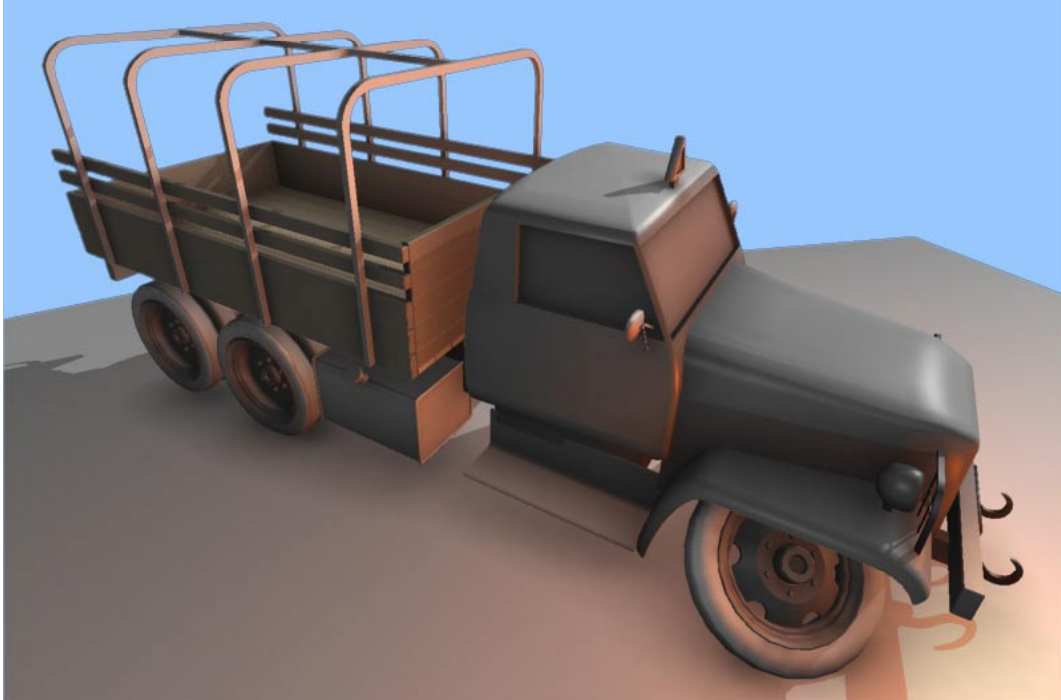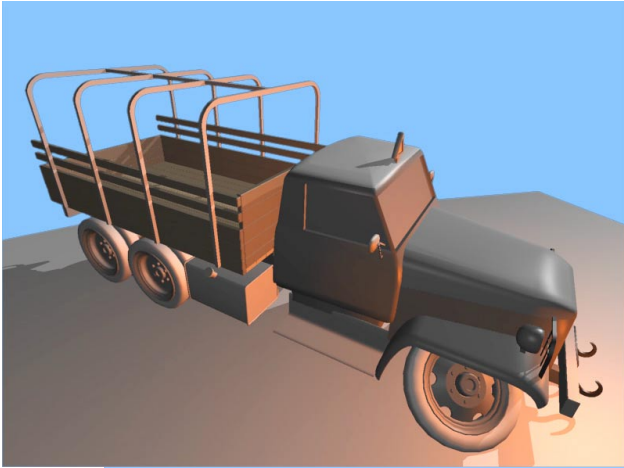Note: Large patch count was used since the scene was intended for close-up rendering.

Figure11. Automobile model ray traced traced with 1 light source (top left) and obscurances map (top right).Bottom:final composed image with obscurances

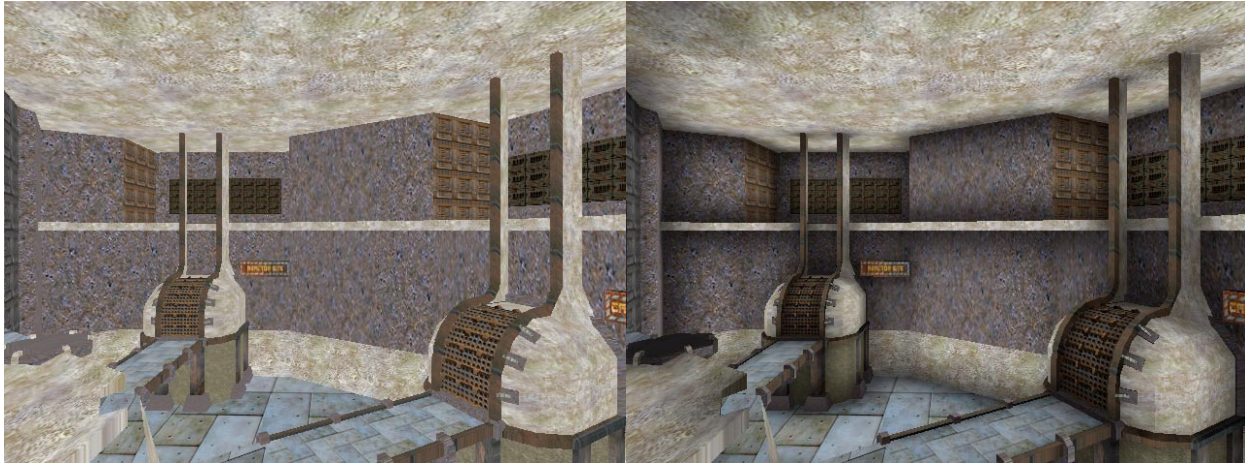*Figure12. Temple model ray traced with 1 light source (left) and final composed image with obscurances (right).*

Figure 13.Human environment with textures only (left) and with textures and obscurance maps (right). Note that the geometry on the right image is much better discernable than on the left one.

Figure 14. Alien environment with textures and obscurance maps.