DOI : 10.1145/1837026.1837049

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A Fast Deferred Shading Pipeline for Real Time Approximate Indirect Illumination

Cyril Soler — Olivier Hoel — Franck Rochet — Nicolas Holzschuch

N° 7162

Decembre 2009

Thème COG

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--7162--FR+ENG

A Fast Deferred Shading Pipeline for Real Time Approximate Indirect Illumination

Cyril Soler , Olivier Hoel [*], Franck Rochet [†], Nicolas Holzschuch [‡*] [†]

Thème COG — Systèmes cognitifs
Équipes-Projets Artis

Rapport de recherche n° 7162 — Decembre 2009 — 17 pages

Abstract: In this paper, we present a deferred shading algorithm for computing iapproximate screen-space multi-bounce indirect illumination with visibility, in real time. For each frame, we compute mipmapped G-Buffers of depth, normals, illumination and voxelized geometry. To each mipmap level we apply a single shader that gathers screen-space illumination using local Monte-Carlo integration. We upsample the illumination for all levels and smoothly combine them together. Our calculation is approximate but does not show artifacts, because it relies on noise-free Monte-Carlo integration of incoming illumination and temporal filtering. Our method simulates arbitrary distant illumination including visibility at a very low cost, because we only perform local texture lookups during computation. Besides, its deferred shading nature makes it independent of geometric and lighting complexity.

Key-words: Global Illumination, Real Time, Video Games

[*] INRIA Rhône Alpes
[†] EDEN Games
[‡] INRIA Rhône Alpes

Centre de recherche Inria de Grenoble – Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Un Algorithme Rapide d'Eclairage Indirect Hierarchique en Espace Image

Résumé : Nous présentons un methode de calcul en espace image de l'éclairage indirect pour des scènes animées. Notre méthode est compatible avec la technique du *deferred shading* utilisée dans les jeux vidéos. Elle est basée sur un calcul multi-echelles de l'éclairage dans un *mipmap*, qui est ensuite re-combiné et composé avec l'eclairage direct. L'eclairage ainsi calculé est approximatif, mais sans artéfacts visuels, car nous utilisons un filtre temporel et spatial pour réduire efficacement la variance due à l'échantillonnage. Notre méthode calcule l'éclairage indirect à distance quelconque, en prennant en compte la visibilité. Son coût reste très faible car elle n'effectue que des requètes locales dans les *G-buffers*. Son coût est par nature indépendant de la complexité géometrique et photométrique de l'image.

Mots-clés : Eclairage Global, Temps Réel, Jeux Vidéos

Contents

1 Introduction

Computing indirect lighting in real time is both challenging and computationally expensive, as it requires to compute potentially several integrals for each visible point.

In 3D video games,it adds a lot to both gameplay and scene realism. However, to be usable in a real 3D video game, indirect light computation should fulfill restrictive constraints: (1) the computation should be very fast and –most importantly– with a constant cost (independent of both the geometric and lighting complexity of the scene); (2) the indirect illumination algorithm should work with any kind of source of direct illumination –not only point light sources; (3) the computed result may be approximate (e.g computed in screen space) but must be artifact-free and temporally coherent on dynamic scenes; (4) The algorithm must be easy to integrate into a game engine pipeline, *i.e.* must have a reasonnable memory footprint and no need for geometry-dependent structures.

We present a simple, yet efficient solution to computing multiple-bounce screen-space indirect illumination in real time accounting for visibility, that satisfies these needs. Our algorithm is a pure deferred-shading method: we gather illumination at several scales, using mipmapped G-Buffers for illumination, normals, positions and visibility. The mipmapped contributions are then upsampled and combined together to result in indirect lighting between all visible parts of the scene. To enhance the stability of the algorithm, we combine indirect lighting across successive frames, with filtering to avoid noise. In addition, the gathering nature of the algorithm allows us to treat glossy surfaces (for the last bounce only) with importance sampling.

Our method is very fast (less than $50ms/10ms$ with/without visibility for a 1280×720 resolution) because it only performs close-range texture lookups while computing arbitrary distance indirect illumination. It naturally scales up to very complex scenes, since both our internal representations of light and visibility are totally decorrelated from the geometric and lighting complexity of the scene. It is however approximate in some ways, which we also discuss in this paper. In summary, our contributions are:

- we present a real-time deferred shading method for computing indirect illumination
- we adapt a GPU voxelization algorithm to compute visibility
- we propose simple yet efficient solutions to screen-space importance sampling and noise-free Monte-Carlo integration.

2 Previous Work

There exists a large body of work in the area of indirect illumination. We will limit ourselves to works that are relevant to our topic: ambient occlusion, interactive global illumination and real time indirect illumination.

2.1 Ambient Occlusion

Ambient Occlusion [30] is a purely geometrical computation, and thus does not approximate physically based global illumination. Although ambient occlusion was precomputed and stored in light maps in early works [30, 7, 14], it is now computed on the fly in image space on the GPU [24, 8]. Screen-space ambient occlusion (SSAO) adapts particularly well to video games [25, 27], because it only performs local texture look-ups and fits well in a deferred shading pipeline. Ritschel extended SSAO by reading the long distance incoming light from an environment map, and the short distance light from nearby objects [23].

2.2 Interactive global illumination

Instant radiosity [11] performs interactive global illumination without any preprocessing by shooting light from virtual point light sources. It is interactive for moderately complex scenes in its original form. Implicit visibility [4] elegantly deals with occlusion: the space of directions is binned and between bins of different points. only the closest source point for each bin of a receiver point is considered, which implicitly accounts for visibility. Antiradiance [3] is a reformulation of the rendering equation in which links transport possibly negative energy to counterbalance occluded contributions.

Coherent Surface Shadow Maps computes multi-bounces indirect illumination for dynamic scenes made of several rigid objects [22]. *Micro-Rendering* [21] combines importance sampling and fast computation of indirect lighting using a collection of micro illumination buffers, using a hierarchical point-based representation of the scene.

Nowrouzezahrai and Snyder proposes an extension of their soft self-shadowing technique for height fields [28] to handle indirect illumination as well [17]. The visibility and radiance at each point are approximated by low order polynomials. This algorithm handles glossy reflexions, although it is limited to low frequency lighting effects.

2.3 Approximate indirect illumination on GPU

Reflective Shadow Maps [1] is inherently a gathering method that uses shadow maps to store direct illumination, as a source of indirect light, while ignoring visibility. Splatting indirect illumination [2] extends this technique: it uses the traditional shadow maps technique to encode which points in the scene actually contain direct illumination. The contribution from these points is then splatted in screen space using deferred shading. This technique performs a single bounce of indirect illumination and neglects visibility during this step, but handles glossy reflectors. Light pyramids [12] are also a fast way of computing close range indirect illumination without occlusion.

Imperfect shadow maps [20] are low-resolution shadow maps that contain approximate visibility information, constructed by splatting a point-based representation of the geometry. They approximately account for visibility in indirect lighting. The radiance volume [10], used in the CryEngine 3 game engine, simulates volumetric light propagation, stored in a 4 channel 3D texture of spherical harmonic distributions. It is completed by local screen-space indirect illumination for color bleeding, and screen-space ambient occlusion.

Multi-resolution Splatting of Indirect Illumination [16] and Hierarchical Image-Space Radiosity [15] compute long-distance indirect illumination. They use virtual point lights (VPL) to represent direct lighting, which they collect with Reflective Shadow Maps (RSM). While the former adaptively splats illumination for all VPLs, the later further organizes the VPLs into a tree structure to allow a fast gathering.

2.4 Comparison with previous work

Our approach fits in the class of approximate GPU-based methods.

The indirect illumination computation proposed by Ritschel [23] is only local. We compute arbitrary distant indirect illumination. Coherent surface shadow maps [22] require static objects, and need to explicitly handle the geometry. This is not suitable for complex environments, and we don't share this limitation.

The cost of using Reflective Shadow Maps (RSMs) depends on the complexity of the direct illumination. A significant number of RSMs might be needed to approximate low frequency sources of direct illumination such as envmaps, or indirect lighting for a second bounce. Methods based on RSMs [1, 2, 16, 15] are thus not well suited to unpredictable lighting conditions, and to more than a single bounce. Since we deal with screen-space illumination, for both input and output, we don't have this limitation.

Splatting methods [2, 16] can importance sample non diffuse BRDFs on reflectors but not on receivers, because all pixels share the same virtual point lights (VPLs). This also applies to Hierarchical Image-Space Radiosity [15]. Our method performs gathering, thus each pixel has its own sets of lights samples in screen-space.

Imperfect shadow maps [20] require preprocessing the geometry, and a specific treatment for each kind of light source, which makes it incompatible with fully dynamic scenes where the geometry content is not known in advance, such as in multi-player video games.

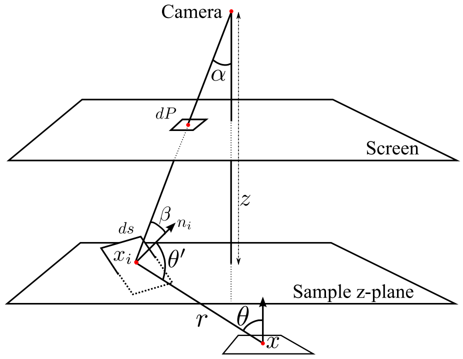


Figure 1: Computation of the indirect illumination in screen space computed at point x. x_i is the current sample, dP the screen differential area and ds the scene differential area of current sample.

3 Screen-space indirect illumination

The indirect illumination at a point x in direction ω is computed as an integral over incoming directions of the hemisphere above x [9]:

$$L'(x,\omega) = \int_\Omega \rho(x,\omega,\omega')L(x',-\omega')\cos\theta d\omega$$

where $\rho(x,\omega,\omega')$ and $L(x,\omega)$ are the BRDF and direct radiance functions at x and directions ω and ω'; x' is the point seen from x in direction ω'; θ is the angle between the normal at x and direction $-\omega'$. This integral can be expressed on all scene surfaces instead of directions. Introducing a visibility term v and the positive-truncated cosine $\cos_+$:

$$L'(x,\omega) = \int_S \rho(x,\omega,\omega')L(x',\omega')\frac{\cos_+\theta\cos_+\theta'}{\|x-x'\|^2}v(x,x')ds$$

We discretize this integral and sum over a large number N of small areas $ds_i = \frac{S}{N}$ across the scene, so that:

$$L'(x,\omega) = \sum_{i=1}^{N} \rho(x,\omega,\omega'_i)L(x_i,-\omega'_i)\frac{\cos_+\theta_i\cos_+\theta'_i}{\|x-x_i\|^2}v(x,x_i)ds_i \qquad (1)$$

In a deferred shading setup, we need to draw samples in image space. We have to re-write ds in terms of the elementary screen-space area dP covered by each screen-space sample (See Figure 1):

$$ds = \frac{z^2 4\tan\frac{f_h}{2}\tan\frac{f_v}{2}dP}{WH}\ \frac{\cos\alpha}{\cos\beta} \qquad (2)$$

In this expression, f_h and f_v are the horizontal and vertical field of view, W and H the width and height of the screen in pixels, and z the depth of the projected pixel. The left quotient in Equation 2 expresses for a pixel region of size dP, the corresponding projected area at depth z on a surface parallel to the screen. The right quotient corrects this area when this surface is not parallel to the screen. Reporting this expression in Equation 1 gives us the indirect illumination of a point x when sampled in screen-space:

$$L'(x,\omega) = \sum_{i=1}^{N} \rho_i L_i v_i \frac{\cos_+ \theta_i \cos_+ \theta_i'}{r_i^2} \frac{z_i^2 4 \tan f_h \tan f_v}{WH} \frac{\cos\alpha}{\cos\beta} dP_i \qquad (3)$$

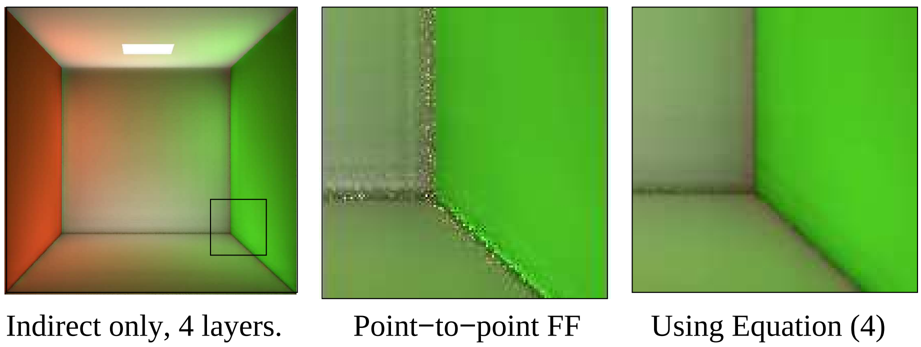


Figure 2: Using the point-to-disc approximation (Equation 4) of the point-to-differential area form factor removes the noise while computing one bounce of indirect illumination with our hierarchical deferred shading algorithm.

The point-to-differential area form-factor $\cos\theta\cos\theta' ds/r^2$ in Equation 3 is a source of high variance, since this term is not bounded as r approaches zero. One solution to this problem was proposed in the Virtual Spherical Lights paper [6]. We propose a simpler workaround: in Equation 3, we replace it by the closed formula for point-to-disc [18]:

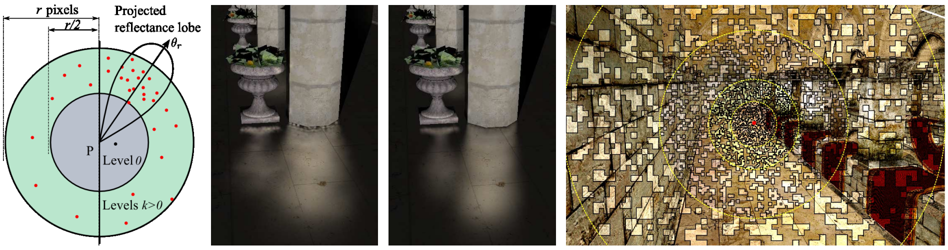
$$\frac{\cos\theta\cos\theta' ds}{r^2} \approx \frac{R^2\cos\theta}{r^2 + R^2} \qquad \text{with} \qquad R^2 = \frac{1}{\pi}\cos\theta' ds \qquad (4)$$

This approximation is valid for small values of ds, and stays bounded. We validate in Figure 2 the gain in using this formulation.

4 GPU screen-space computation

We perform a hierarchical implementation of Equation 3: the depth, normals, visibility and direct illumination buffers (a.k.a G-Buffers) are mipmapped and successively sent to a single local shader. The contribution from all mipmaps are then up-sampled and added together (see Figure 4 for a view of the entire pipeline). Directly using Equation 3 to evaluate arbitrary-distant illumination in screen-space would otherwise result in an unacceptable cost due to long-distance texture lookups in the G-buffers.

Indirect lighting shader In each mipmap level, the shader draws samples in a ring defined by two radii r and $r/2$ (See Figure 3). r is the same for all mipmap levels. For the first level only the full disc of radius r is sampled.



Samples per level | No importance sampling | Importance sampling | Participating pixels for central (red) point

Figure 3: *left:* Placement of samples in the shader. In all mipmaps except the first (largest) one, the same region is sampled. This region is a ring delimited by two circles of radii r and $r/2$. *middle:* our screen-space importance sampling function allows to render glossy reflections of indirect light on the floor without noise (Phong exponent is 128). *right:* position of all samples that eventually participate to the indirect light of the central pixel after upsampling and adding mipmap levels.

This ensures that by combining contributions from all mipmap levels, the entire screen eventually gets sampled without cracks nor overlaps. The optimal value for r is the smallest possible value so that the entire screen is eventually sampled when combining contributions from all mipmap levels. A typical setting is to use 4 levels and $r = 40$ pixels for a 1280×720 screen resolution.

We keep a constant number of samples across mipmap levels. This way, the sampling density in the resulting image is such that farther regions get less sampled. Because at step k the shader works on a mipmap at scale 2^k, the collected energy must be scaled by a factor 2^{2k}.

Screen-space importance sampling We render glossy reflexions of indirect light. Because we're gathering light energy, we are able to deal with glossy BRDFs for the last bounce before the camera. For this we importance-sample the screen-space samples according to the position of the projected lobe of the BRDF (See Figure 3). For the Phong and Lafortune models we successfuly used the following importance function:

$$f(\theta) = \alpha + e^{-s\frac{(\theta_r - \theta)^2}{2\pi}}$$

This importance function is essentially a small constant *alpha* (0.1 in our experiments) plus a gaussian with variance inversely proportional to the square root of the exponent s of the lobe. We took inspiration from Rammamorthi's angular frequency estimate for the Phong model [19].

Visibility computation For each frame, we perform a solid voxelization of the scene restricted to the view frustum, as proposed by Eisemann and Decoret [5]. We compute the visibility term v_i in Equation 3 by ray-marching in this grid.

Constructing the voxel grid needs an additional rendering pass, with specific rendering parameters: the z-buffer is desactivated, and blending is done with

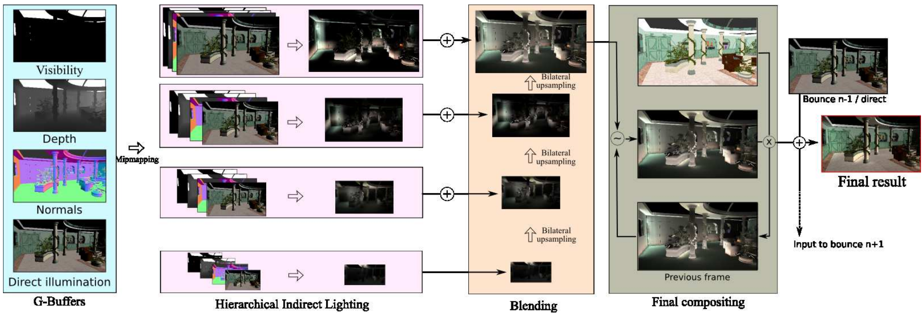


Figure 4: Complete pipeline for our screen-space indirect illumination algorithm. At each frame, the direct illumination, normals, voxels and depth buffers coming from the rendering loop are mipmapped. Each set of mipmap is sent to the indirect lighting shader, and the resulting indirect illumination contributions are up-sampled and added together. The newly computed values are averaged with the values of the previous frame (for compatibility-tested warped pixels only). The obtained illumination is multiplied by the albedo buffer and added to the direct illumination.

a XOR [5]. This produces a binary grid (one bit per voxel) stored in 4 render targets of 32bit each. The size of the grid is thus 128 in depth and equal to the size of the screen in x and y.

We always perform ray marching in the current mipmap level of the grid, thus computing v_i requires only local texture look-ups in the visibility buffer as well. For each ray, we add a bias to the start and end points, to avoid mistaking the start and end voxels for occluders.

Filtering of mipmap levels We can't interpolate normals and depths across mipmap levels because it produces inconsistent results, and appearance/disappearance of small objects causes temporal aliasing. To filter the depth and normals, we thus use a voting scheme among sub-pixels of the next mipmap level. To favor large objects, the filtered normal (resp. depth) at level k is the normal (resp. depth) for which sub-pixel p_i in level $k+1$ has the largest score $s(p_i)$, using:

$$s(p_i) = \sum_{|p_i - p_j|_{L_1} \leq 1} g_1(|p_i - p_j|) g_2(1 - n_j.n_i) g_3(z_j - z_i) \quad (5)$$

In this expression, z_j and n_j are the depth and normal at pixel p_j, and g_1, g_2, g_3 are gaussians. This expression automatically keeps larger areas of constant depth and normals and discards small inconsistent regions. We show in Figure 5 how this filtering method acts on normals across mipmap levels.

We can't interpolate the voxel grid through mipmap levels either, because it is a binary representation (1 bit per voxel). We thus conservatively downsample it by applying an AND operation between all sub-voxels in the next mipmap level, in the x and y dimensions only.

Continuous combination of all levels We filter and blend illumination contributions from all mipmap levels using a joint bilateral up-sampling filter [13, 21]: mipmaps levels of indirect illumination are computed starting from

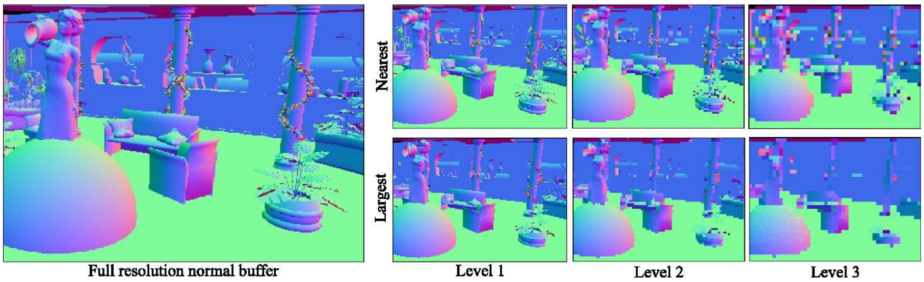


Figure 5: Filtering of normals across mipmap levels using the closest pixel produces inconsistent results. Instead, we filter by choosing at level k the normal of the sub-pixels in level $k+1$ that corresponds to the most important object, according to the score defined by Equation 5

the smallest, and added into a buffer. At the end of each step, this buffer is up-sampled to the size of the next mipmap. This way, the illumination at level k is filtered several times in the final result. This produces smooth gradients in the final indirect lighting results, low noise and no edge artifact. Figure 6 presents an example where one can separately see the filtered contribution of each mipmap level.

We don't need to perform the computation starting at the full screen resolution. This gives us an efficient way to trade-off speed for quality in the algorithm (We mainly use half and quarter start resolutions). It suffices indeed that the mipmapped results are properly up-sampled to the final image resolution. Figure 10 shows the effect fo variing this parameter, which is a loss of high frequencies in indirect light.

Temporal filtering Using a smaller number of samples in the shader increases the speed, but is also a source of artifacts. We reduce the variance due to too few Monte-Carlo samples –at no extra cost– by blending the currently computed indirect illumination I_i with that of the previous frame I_{i-1}. For this we warp the previous frame to fit the current camera parameters before blending (More general solutions to this problems are presented e.g. in [26]):

$$F_i = a\mathcal{W}(I_{i-1}) + (1-a)I_i \tag{6}$$

In this expression $\mathcal{W}$ represents the warping function that gives for the current pixel the position of this pixel as seen with the camera settings of the previous frame. For moving objects, $\mathcal{W}$ accounts for both the camera and the moving object. Once found, the depth and normal of this transformed position are compared to the depth and normal of the corresponding pixel in frame $i-1$, to lower the chances of blurring irrelevant pixels together. Adapting parameter a to both the moving speed and shininess of the point's brdf helps finding a good compromize between variance and ghosting.

Multi-bounce computation Computing more than one bounce of indirect light with our technique is easy because the input and output illumination of our pipeline have the same format. We achieve this by using illumination of bounce $n-1$ as an input of bounce n.

Direct illumination | Indirect (level 1 only) | Indirect (level 2 only)

Indirect (level 3 only) | Indirect (level 4 only) | Final image (1+2+3+4)

Figure 6: Using our technique, the indirect illumination is computed at various scales in screen space, to account for all-ranges indirect illumination. From left to right, top to bottom: the direct illumination component from a vertical spotlight, the indirect illumination computed by level 1 to 4, and the combined result of our shader. In levels 2 and 3, one can see the shape of the sampling area covered by the shader. Illumination from the different levels blend smoothly and continuously.

No indirect visibility | With indirect visibility | Difference (scaled)

Figure 7: A comparison between not using visibility (*left*) and using visibility (*middle*) for indirect lighting shows that additional soft indirect shadows are captured by our algorithm. *Right:* scaled difference of the two images shows indirect light was removed.

5 Results

On Figure 7 we illustrate the effect of computing indirect visibility with our technique. The difference images shows that our method can simulate indirect soft shadows with an acceptable visual quality. On Figure 8 we show an example of computing indirect illumination in the Sponza atrium with our technique. The same figure also shows the detailed computation times for this scene.

5.1 Analysis of cost

Our algorithm allows a wide range compromise between accuracy and speed. One can easily increase the framerate by reducing the number of samples per level. Another efficient cost reduction technique is to apply the indirect illumi-

Direct illumination only

Indirect Illumination only

Direct and indirect

Figure 8: Computed screen-space indirect illumination in the Sponza atrium with our technique.

Resolution	Quarter		Quarter		Half		Half	
Mipmap levels	4		4		4		5	
Samples per level	64		256		64		64	
Use visibility	No	Yes	No	Yes	No	Yes	No	Yes
Voxel grid		4.5		4.5		4.5		4.5
SSIL level 1	4.74	27.9	17.5	103	19.1	78.2	19.12	78.21
SSIL level 2	1.25	11.78	4.61	43.3	5.06	29.3	5.06	29.3
SSIL level 3	0.41	6.18	1.46	24.8	1.34	11.8	1.34	11.8
SSIL level 4	0.21	0.74	0.74	16.1	0.43	6.28	0.43	6.27
SSIL level 5							0.23	4.3
Total SSIL (1 bounce)	6.61	50.24	24.3	187.9	25.9	125	26.2	129
Upsampling + TC	3.23	3.24	3.23	3.23	3.68	3.68	3.72	3.72
Total (ms)	**9.84**	**58**	27.58	194	29.6	131	29.9	136

Figure 9: *Bottom left:* Computation times in milliseconds for different parameter sets, using a NVidia GTX260 card, for a display resolution of 1280×720 pixels. The sampling radius is $r = 40$. TC stands for Temporal Coherence.

nation shader only starting from half or quarter resolution mipmaps. Figure 8 shows GPU computation times for the differents steps of our method, with different quality settings for the Sponza atrium scene. These measurements were done on a nVidia GTX260 card, rendering at a resolution of 1280×720. The sampling radius for the shader is $r = 40$ pixels. The varying parameters are the starting resolution (half or quarter resolution), the number of mipmap levels, and the number of samples used at each level. We observe that, for a same cost, we get a better quality by starting from a lower resolution but using more samples (entry 2 and 4). Local indirect illumination has less details but noise and variance is much more noticeable while animated, and better effect range (see Figure 10). Note that the cost of temporal filtering is independent from these parameters (it only depends on the rendering resolution). The last entry shows that adding one indirect lighting level only adds a very small computation overhead (¡ 1%) whereas it greatly increases the effect range in the final result (see Figure 6).

5.2 Integration in a Game Engine pipeline

In order to validate our technique in an industry constrained game engine, we have implemented our technique in the engine which is used in the game Alone In The Dark "Near Death Investigation". We believe this rendering engine is typical of modern game rendering pipelines based on deferred shading. Integrating our indirect illumination algorithm in this engine proves its scalability to real world situations. Images of Figure 10 were produced with our technique in this game engine.

5.3 Discussion and limitations

A common problem to screen space computation methods is that only visible geometry can hold sources for indirect light, which theoretically means that some parts of the indirect illumination can be missing. From our experiments we conclude that such artifacts are most of the time barely noticeable, and do not cause frame coherency problems.

Very specular surfaces can produce ghosting effects when used in conjunction to temporal filtering. Our solution is to adapt parameter a in Equation 6 to the glossiness of surfaces.

Although successive bounces of indirect illumination share the mipmapping of G-Buffers, the light gathering shader must be applied as many times. Fortunately, there are several ways of limiting this cost: one is to suppress the visibility computation after bounce 1, which results in acceptable artifacts [29]. An other one is to compute exactly two bounces interleaved between odd/even frames, and add them as we do for controlling temporal coherence. This works because there's a lot of temporal coherence in indirect lighting.

As we're using a simple G-Buffer to represent the input illumination (previous bounce, or direct illumination), we are limited to a diffuse approximation of this light. Nevertheless, we make the assumption that in glossy scenes, accounting for glossiness is mostly important for the last bounce to the camera, as it reveals the glossy nature of objects.

Finally, for the solid voxelization to work, the scene geometry needs to be water tight. This is an acceptable limitation [5].

6 Conclusion and future work

We have presented a deferred shading algorithm for computing indirect illumination; our algorithm performs all computations in screen space, and computes indirect lighting hierarchically: illumination from distant points is computed with less precision then illumination from nearby points. In addition, we have shown that temporal filtering drastically removes flickering, at a very low additional cost. Our algorithm is both really fast and robust: for typical settings, we are able to compute indirect illumination in real time. Because there are no precomputations and we rely only on screen-space information, our algorithm was easily combined with a deferred shading pipeline, in a typical industry-strength video-game engine.

In the future, we would like to explore ways of exploiting the coherence of visibility queries between the different mipmap levels. Indeed, close range queries for a given mipmap level give information about blocked directions in all successive (longer distance) mipmap levels.

References

[1] Carsten Dachsbacher and Marc Stamminger. Reflective shadow maps. In *Symposium on Interactive 3D Graphics and Games*, pages 203–231, 2005.

[2] Carsten Dachsbacher and Marc Stamminger. Splatting indirect illumination. In *Symposium on Interactive 3D Graphics and Games*, pages 93–100, 2006.

[3] Carsten Dachsbacher, Marc Stamminger, George Drettakis, and Fredo Durand. Implicit visibility and antiradiance for interactive global illumination. *ACM Transactions on Graphics*, 26(3), 2007.

[4] Zhao Dong, Jan Kautz, Christian Theobalt, and Hans-Peter Seidel. Interactive global illumination using implicit visibility. In *Pacific Conference on Computer Graphics and Applications*, 2007.

[5] Elmar Eisemann and Xavier Décoret. Single-pass gpu solid voxelization for real-time applications. In *Graphics Interface 2008*, pages 73–80, 2008.

[6] Miloš Hašan, Jaroslav Křivánek, Bruce Walter, and Kavita Bala. Virtual spherical lights for many-light rendering of glossy scenes. *ACM Transactions on Graphics*, 28(5), 2009.

[7] Andrey Iones, Anton Krupkin, Mateu Sbert, and Sergey Zhukov. Fast, realistic lighting for video games. *IEEE Computer Graphics and Applications*, 23(3):54–64, May/June 2003.

[8] V. Kajalin. Screen-space ambient occlusion. In *Shader X7*. Wolfgang Engel, Ed., Charles River Media, 2009.

[9] James T. Kajiya. The Rendering Equation. *Computer Graphics (SIGGRAPH '86 Proceedings)*, 20(4), 1986.

[10] Anton Kaplanyan. Light propagation volumes in CryEngine 3. In *SIGGRAPH 2009 Course Notes - Advances in Real-Time Rendering in 3D Graphics and Games*, 2009.

[11] Alexander Keller. Instant radiosity. In *SIGGRAPH '97*, 1997.

[12] Hyunwoo Ki and Kyoungsu Oh. A gpu-based light hierarchy for real-time approximate illumination. *The Visual Computer*, 24(7-9):649–658, July 2008.

[13] Johannes Kopf, Michael Cohen, Dani Lischinski, and Matt Uyttendaele. Joint bilateral upsampling. *ACM Transactions on Graphics*, 26(3), 2007.

[14] Alex Mendez, Mateu Sbert, and Jordi Cata. Real-time obscurances with color bleeding. In *Spring Conference on Computer Graphics (SCCG 2003)*, 2003.

[15] Greg Nichols, Jeremy Shopf, and Chris Wyman. Hierarchical image-space radiosity for interactive global illumination. In *Eurographics Symposium on Rendering 2009*, 2009.

[16] Greg Nichols and Chris Wyman. Multiresolution splatting for indirect illumination. In *Symposium on Interactive 3D Graphics and Games*, 2009.

[17] Derek Nowrouzezahrai and John Snyder. Fast global illumination on dynamic height fields. In *Eurographics Symposium on Rendering 2009*, 2009.

[18] O. S. Pianykh, J. M. Tyler, and Jr. W. N. Waggenspack. Improved Monte Carlo form factor integration. *Computers & Graphics*, 22(6), 1998.

[19] Ravi Ramamoorthi and Pat Hanrahan. A signal-processing framework for reflection. *ACM Transactions on Graphics*, 23(4):1004–1042, 2004.

[20] T. Ritschel, T. Grosch, M. H. Kim, H.-P. Seidel, C. Dachsbacher, and J. Kautz. Imperfect shadow maps for efficient computation of indirect illumination. *ACM Transactions on Graphics*, 27(5), 2008.

[21] Tobias Ritschel, Thomas Engelhardt, Thorsten Grosch, Hans-Peter Seidel, Jan Kautz, and Carsten Dachsbacher. Micro-rendering for scalable, parallel final gathering. *ACM Transactions on Graphics*, 28(5), 2009.

[22] Tobias Ritschel, Thorsten Grosch, Jan Kautz, and Hans-Peter Seidel. Interactive global illumination based on coherent surface shadow maps. In *Graphics Interface 2008*, May 2008.

[23] Tobias Ritschel, Thorsten Grosch, and Hans-Peter Seidel. Approximating dynamic global illumination in image space. In *Symposium on Interactive 3D Graphics and Games*, 2009.

[24] Perumaal Shanmugam and Okan Arikan. Hardware accelerated ambient occlusion techniques on GPUs. In *Symposium on Interactive 3D Graphics and Games*, 2007.

[25] Oles Shishkovtsov. Deferred shading in s.t.a.l.k.e.r. In *GPU Gems 3*, chapter 9. Addison-Wesley, 2005.

[26] Pitchaya Sitthi-amorn, Jason Lawrence, Lei Yang, Pedro V. Sander, Diego Nehab, and Jiahe Xi. Automated reprojection-based pixel shader optimization. *ACM Transactions on Graphics*, 27(5), 2008.

[27] Niklas Smedberg and Daniel Wright. Rendering techniques in Gears Of War 2, March 2009. GDC 2009.

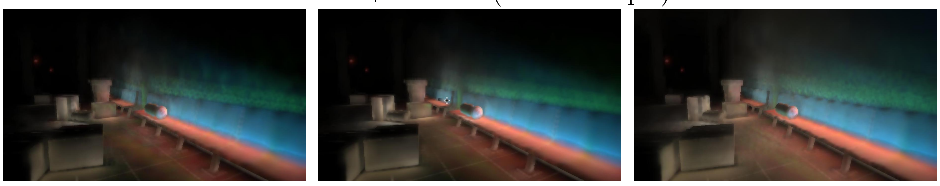
[28] John Snyder and Derek Nowrouzezahrai. Fast soft self-shadowing on dynamic height fields. In *Eurographics Symposium on Rendering*, June 2008.

[29] Insu Yu, Andrew Cox, Min H. Kim, Tobias Ritschel, Thorsten Grosch, Carsten Dachsbacher, and Jan Kautz. Perceptual influence of approximate visibility in indirect illumination. In *Symposium on Applied Perception in Graphics and Visualization*, 2009.

[30] Sergei Zhukov, Andrej Iones, and Grigorij Kronin. An ambient light illumination model. In *Eurographics Rendering Workshop*, pages 45–56, 1998.

Direct illumination only

Direct + indirect (our technique)



Indirect, $\frac{1}{2}$ res., 64 sp. Indirect, $\frac{1}{2}$ res., 256 sp. Indirect, $\frac{1}{4}$ res., 64 sp.

Figure 10: Screen-shots of the game Alone In the Dark, into which we have plugged our technique for computing the indirect illumination (Note the glossy indirect reflexions on the bin). The bottom row shows the incident indirect illumination computed with 3 different setups corresponding to the computation times of Figure 8. Note how much, on the top-right image, does the indirect illumination add realism and immersion, revealing regions of the scene not directly illuminated.

Centre de recherche INRIA Grenoble – Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Centre de recherche INRIA Futurs : Parc Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399