EFFICIENT RENDERING OF REALISTIC METALS

by

Lazányi István Zsolt
Department of Control Engineering and Information Technology
Budapest University of Technology and Economics

supervisor

Prof. Szirmay-Kalos László, Ph.D.
Department of Control Engineering and Information Technology
Budapest University of Technology and Economics

2008
Introduction

The main objective of rendering is to provide the computer user with the illusion of watching real objects on the computer screen [SKAC03, SP92]. This illusion requires the brightness of the pixels be identical to the brightness of the point visible in the direction defined by the pixel. Thus image synthesis should identify the visible points, and then compute the brightness of these points. To query the point that is visible through a pixel, we have to start a line, called ray, from the eye position through the center of the given pixel, and determine the intersection of the ray with the objects of the virtual world. The visible point, also called shaded point, is defined by the intersection that is closest to the eye position.

Assuming that the shaded point itself does not emit light, the brightness of the shaded point can be determined by considering the illumination received from other surface points. That is, we have to consider all possible incoming directions at that point and summarize their contributions to the brightness of the shaded point. The resulting integral is called the rendering equation [Kaj86]. To determine the surface points that are visible from the shaded point, new visibility queries are needed. Furthermore, to calculate their contributions, the brightness of those surface points is also needed. This will recursively generate new surface points and new visibility queries. Finally, the solution of the problem can also be explained as the identification of those light paths that connect the light sources to the eye via single or multiple reflections or refractions. When these multiple reflections or refractions are taken into account, the method belongs to the category of global illumination methods [Ant04, SK08, SKSS08]. On the other hand, if indirect illumination is ignored, i.e. only single reflections and refractions are considered, we talk about local illumination methods.

Interactive applications

Computer games and real-time systems must maintain an interactive frame rate (that is, rendering at least 20 images per second) while rendering very complex virtual environments. Thus, precomputations and other simplifications are necessary that significantly reduce the rendering time without destroying image quality. Taking into account that pre-computation works well for static geometry (and especially for diffuse surfaces), static and dynamic objects are usually separated. The self illumination of the static scene is usually rendered in a pre-processing phase by a non real-time global illumination algorithm (e.g. radiosity).

On the other hand, the illumination falling onto dynamic objects, as well as the illumination reflected by these dynamic objects must be determined at run-time. The effect of dynamic objects onto the static environment is usually handled by shadow mapping, that is, the direct lighting of static objects is modulated by the presence of dynamic ones. Since dynamic objects are usually much smaller than the static environment, this is usually acceptable. However, because of these size differences, the indirect illumination from static objects onto dynamic ones cannot be ignored, but must be computed with sufficient accuracy. Representing the environment with images can significantly speed up these calculations. This technique is called environment mapping [BN76].

Rendering metals

In a typical computer game, the scene is surrounded by diffuse, textured walls that reflect the colorful lights emitted by dynamic objects. These colorful reflections not only provide a pleasing visual result but also may help to hide some artifacts introduced during rendering. Furthermore, reflective, metallic objects such as weapons and armors are also often included. The reflections caused by these objects help to increase the contrast of the scenes and also...
help to establish the atmosphere of the game, especially for dark scenes (e.g. “knives glinting in the moonlight”). However, to save precious rendering time, metals such as steel, silver and aluminum are rendered in games so that they look equally “gray”, although realistic silver could be easily distinguished from steel due to its yellowish look.

Metals have several unique material properties that are worth studying from rendering aspects. For example, the refraction index for metals is a complex number, where the imaginary part describes how the electromagnetic waves die away in the material. Thus, metals do not let electromagnetic waves travel inside the material, as also demonstrated in the Faraday cage experiment. Furthermore, the refraction index for metals varies significantly on different wavelengths. Thus the Fresnel term, which is derived from the refraction index and determines the amount of reflected light for a given incidence angle, will also depend on wavelength, resulting in colored specular reflections (the specular highlight for most materials, e.g. plastics is white).

Depending on their optical quality, metals can have different reflective properties. While optically smooth (polished) metals can be treated as ideal mirrors, optically rough (non-polished, oxidized) metals act as glossy mirrors, or as diffuse surfaces in the extreme case. Thus, to render metals in real time, accurate but simple reflection models and rendering methods are needed for both ideal and non-ideal (glossy or diffuse) reflections. Furthermore, instead of considering direct illumination only, at least one indirect bounce of light should be calculated or at least approximated to achieve realistic result.

**Organization of the dissertation**

In the dissertation, Section 1.1 defines the basic concepts of illumination calculations and introduces the rendering equation, while Section 1.2 discusses the capabilities of the programmable graphics hardware. Then, Chapter 2 reviews the state-of-the-art of the dissertation.

The next four chapters present the new scientific results for rendering metals with hardware support. Namely, Chapter 3 proposes a simplification of the Fresnel function for metals, Chapters 4 and 5 present approximation techniques for rendering ideal and non-ideal reflections, respectively, and Chapter 6 presents a CPU-based global illumination method that also exploits the capabilities of the graphics hardware. Finally, Chapter 7 concludes the results and provides a list of publications.
New scientific results

1 Rendering metals

Colored reflections are governed by the Fresnel term, which can be expressed from the refraction index of the material [Gla95]. The evaluation of this function is rather computation intensive. A simple and accurate rational approximation of the Fresnel term has been proposed by Schlick [Sch93]. However, for metals where the refraction index becomes a complex number, this approximation cannot be applied directly (Figure 1.1).

![Figure 1.1. Refraction indices for different metals, at different wavelengths: (a) real part and (b) imaginary part.](image)

Thesis I: Fresnel term approximations for metals

Related chapter in dissertation: Chapter 3

Related publications: [P1]

By modifying (rescaling) the formula proposed by Schlick, I have proposed an accurate simplification $F^*$ of the Fresnel function $F$:

$$F^*(n, k, \cos \theta) := F_\perp^* + (1 - F_\perp^*)(1 - \cos \theta)^5,$$

where

$$F_\perp^* = \frac{(n - 1)^2 + k^2}{(n + 1)^2 + k^2}.$$

The error of the approximation can be further reduced by applying a rational approximation to the residual error. I proposed the following approximation: $ax(1 - x)^\alpha$, where parameters $a$ and $\alpha$ are obtained from simple parameter constraints.

The resulting formula can also cope with complex refraction indices, thus it can present realistic metals and is simple enough to be implemented on the graphics hardware, and used in games.
2 Rendering specular reflections

When rays originate at a given object, environment mapping takes images about the environment from the center of the object, and then the environment of the object is replaced by a cube, or by a sphere, which is textured by these images. When the incoming illumination from a direction is needed, instead of sending a ray we can look up the result from the images constituting the environment map (Figure 2.1).

![Figure 2.1](image)

Figure 2.1. Steps of environment mapping. (a) Finding the center of the object, (b) taking images from the center, and (c) looking up illumination from the images.

A fundamental problem of environment mapping is that the environment map is the correct representation of the direction dependent illumination only at a single point, the reference point of the object. For other points, accurate results can only be expected if the distance of the point of interest from the reference point is negligible compared to the distance from the surrounding geometry. However, when the object size and the scale of its movements are comparable with the distance from the surrounding surface, errors occur, which create the impression that the object is independent of its illuminating environment.

**Thesis II: Rendering reflective surfaces using approximate raytracing on the GPU**

**Related chapter in dissertation: Chapter 4**

**Related publications: [P2, P3]**

I have proposed a fast approximation method to localize environment mapped reflections. The method uses an iteration process to trace a ray from point \( x \) in direction \( \vec{R} \), i.e. to solve the ray equation \( \vec{q} = \vec{x} + \vec{R} \cdot d \) using the distance values stored in environment map texels. For example, an initial guess with parameter \( d_p \) can be refined as follows:

\[
d_{l} = d_{p} + |\vec{r}| \cdot (1 - |\vec{p}|/|\vec{p}'|),
\]

where \( \vec{r} \) is the hit point proposed by the “classical” environment mapping method, \( \vec{p} \) is the current guess, and \( |\vec{p}|/|\vec{p}'| \) describes the accuracy of the current guess. Guesses with parameters \( d_l \) and \( d_p \) can be refined using the following interpolation formula:

\[
d_{new} = d_{l} + (d_{l} - d_{p}) \cdot \frac{1 - |\vec{l}|/|\vec{l}'|}{|\vec{l}|/|\vec{l}'| - |\vec{p}|/|\vec{p}'|},
\]

where \( \vec{l} \) and \( \vec{p} \) are the corresponding guesses.

The proposed approximation method can be used to localize environment mapped reflections, that is, to make them depend on where they occur. The method is fast and accurate if the scene consists of larger planar faces, when the results are similar to that of ray-tracing. Furthermore, the method suits very well the GPU architecture and can render ray-tracing and global illumination effects interactively.
3 Rendering diffuse and glossy reflections

In case of non-ideal (diffuse and glossy) reflections, multiple directions around the ideal reflection direction have to be considered, making glossy and diffuse reflections quite expensive to calculate (Figure 3.1).

Figure 3.1. Indirect illumination with sampling rays starting from point $\hat{x}$.

Usually, these effects are calculated in a preprocessing phase and the result is stored in a diffuse or specular environment map [KVHS00, KM00]. However, if the rendered object is moving, these maps have to be frequently recalculated to avoid popping artifacts. Alternatively, multiple environment maps can be applied, with significant memory requirements [GSHG98].

Thesis III: Approximate rendering of glossy and matte surfaces on the GPU

Related chapter in dissertation: Chapter 5
Related publications: [P4, P5, P6]

I have proposed a fast approximation method to localize non-ideal illumination effects on a dynamic object. Instead of using multiple environment maps, the calculations are based on a single environment map that records incoming illumination and also stores geometric information of the scene. The environment map is downsampled, which corresponds to clustering lights and replacing a cluster by an area light source. When calculating the contributions of the light sources, the geometric information can be used to cope with the difference between the reference point and the shaded points. Namely, a solid angle considering the reference point can be localized as follows:

$$\Delta \omega^* \approx \Delta \omega \cdot \frac{r^2}{r^*^2},$$

where $\Delta \omega$ and $\Delta \omega^*$ are the solid angles measured from the reference point and from the shaded point, while $r$ and $r^*$ are the distances from the environment, measured from the reference point and the shaded point, respectively. Using area-to-point approximations, the formula can be extended to also deal with the case when the shaded point is close to the environment:

$$\Delta \omega^* \approx 2\pi - \frac{2\pi}{\sqrt{1 + \left(\frac{2\pi - \Delta \omega}{2\pi - \Delta \omega^*} - 1\right) \cdot r^2/r^*^2}}.$$

The method computes indirect illumination although approximately, but providing very pleasing visual quality. The method suits very well the GPU architecture, and can render diffuse and specular interreflections interactively.
4 Rendering multiple interreflections

In Thesis II and Thesis III, only single reflections were considered. In order to calculate multiple reflections I also proposed a GPU friendly global illumination method.

To speed up global illumination computations, we can try to reuse light paths or subpaths generated for a pixel for the computation of other pixels as well. A particularly simple and effective approach based on this idea is the virtual light sources method [Kel97], which enables us to reuse complete light paths at the cost of tracing a single shadow ray per pixel, thus it can solve the rendering equation efficiently (Figure 4.1).

![Figure 4.1. The virtual light sources algorithm. (a) Generating virtual light sources and (b) determining the contributions of the virtual light sources to visible point $\bar{x}$.](image)

Thesis IV: Speeding up the virtual light sources algorithm with the GPU

Related chapter in dissertation: Chapter 6

Related publications: [P7]

I proposed two techniques to speed up the classical, CPU-based virtual light sources algorithm. Firstly, the contribution of the virtual light sources is calculated with the support of the graphics hardware, without taking care of the visibility factor. Secondly, by establishing a priority scheme, the evaluation of a sample is only completed (by tracing the shadow ray) if the sample can have a significant effect on the result.

Since the contributions can be computed independently of the visibility tests, we can utilize the computing power of the graphics card to increase performance.
Application of the results

The primary application area of the proposed methods is the introduction of global illumination effects in computer games. The proposed methods were not only implemented as separate test applications but also integrated into an existing rendering engine as a part of the GameTools FP6 (IST-2-004363) project (www.gametools.org). Thus, some of the proposals may be included in future game engines.
Related publications


Other publications


Bibliography


