GPU-Based Monte-Carlo Volume Raycasting

Christof Rezk Salama
Computer Graphics Group, University of Siegen, Germany

Abstract

This paper presents a practical, high-quality, hardware-accelerated volume rendering approach including scattering, environment mapping, and ambient occlusion. We examine the application of stochastic raytracing techniques for volume rendering and provide a fast GPU-based prototype implementation. In addition, we propose a simple phenomenological scattering model, closely related to the Phong illumination model that many artists are familiar with. We demonstrate our technique being capable of producing convincing images, yet flexible enough for digital productions in practice.

1 Introduction

Volume rendering techniques are important in many scientific areas, such as engineering, computational science and medicine. Throughout the years, GPU-based volume rendering techniques have reached a high stage of maturity, providing high-quality results at interactive frame rates.

In recent years, visual artists who are concerned with the production of educational computer animations, have reported an increasing demand for high quality renditions among their customers. The approaches presented in this paper aim at increasing the visual quality of volume renditions of scanned objects by including multiple scattering effects in a practical way.

Monte-Carlo raytracing techniques are frequently employed whenever photorealistic and physically-based light computation is needed. This paper investigates how GPU-based volume raycasting techniques can be supplemented to support multiple scattering by implementing stochastic raytracing for volumetric objects.

2 Related Work

Many sophisticated techniques to solve the volume rendering integral in real-time have been proposed in the past, including 2D [6] and 3D texture mapping [9]. A detailed overview of GPU-based volume rendering can be found in the book by Engel et al. [1].

The first solely GPU-based implementations of volume raycasting have been published by Krüger and Westermann [5] and Röttger et al. [7]. Hadwiger et al. have proposed a flexible raycasting framework for discrete isosurfaces in large volume data using hierarchical space-leaping and adaptive sampling.

Images generated by the approximation to volume scattering proposed by Kniss et al. [4] were inspirational for this paper. Scattering is approximated by jittered sample position from an attenuation map. Their approach is restricted to illumination from a single point light source or directional light at a time. Hadwiger et al. [2] introduce GPU-based deep shadow maps.

Numerous publications exist on Monte-Carlo-based techniques. As an introduction we refer the reader to the detailed SIGGRAPH course notes [3] or the excellent script by László Szirmay-Kalos [8].

3 Phase Function Model

We use a simple phenomenological phase function model, which is equal to the BSDF at specified isosurfaces and contains a simple forward peak otherwise. The parameters of this phase function model are derived from the underlying scalar field $s(\mathbf{x})$. To keep the model controllable by the user, we restrict scattering events to happen at a fixed set of isosurfaces. Between these isosurfaces the ray direction does not change, but attenuation may still happen. Attenuation is controlled by a traditional transfer function.

At the specified isosurfaces, the gradient magnitude $\nabla s(\mathbf{x})$ is guaranteed to be non-zero. Our phenomonological phase function is illustrated in Figure 1. The reflective part f_r is equal to the specular and diffuse term of the Phong local illumination model,

$$f_r = f_{\text{diff}} + f_{\text{spec}} \tag{1}$$

$$f_{\text{diff}}(\mathbf{v} \leftarrow \omega_i) = k_d \left(\mathbf{n} \cdot \omega_\mathbf{i}\right) \tag{2}$$

$$f_{\text{spec}}(\mathbf{v} \leftarrow \omega_i) = k_s \left(\mathbf{r} \cdot \omega_\mathbf{i}\right)^s \tag{3}$$

$$\text{with } \mathbf{r} = 2\mathbf{n}\left(\mathbf{n} \cdot \mathbf{v}\right) - \mathbf{v}. \tag{4}$$

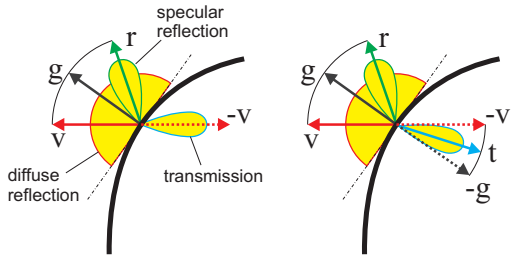

Figure 1. Phase function model with *(right)* and without refraction *(left)*.

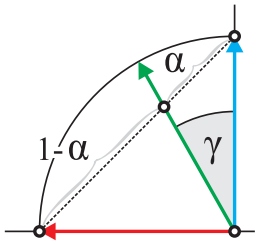

Figure 2. Geometric relationship between the interpolation weight α and the scattering cone angle γ.

The transmissive part scatters the transmitted light in an additional Phong lobe centered around the negative viewing vector -$\mathbf{v}$ in case of non-refractive transmission,

$$f_t(\mathbf{v} \leftarrow \omega_i) \quad = \quad k_t \left(-\mathbf{v} \cdot \omega_\mathbf{i}\right)^q. \tag{5}$$

For refractive transmission (Fig. 1, right), $-\mathbf{v}$ is replaced by the refracted vector $\mathbf{t}$ calculated according to Snell's law.

4 GPU-Based Monte-Carlo Raycasting

Existing implementations of GPU-based raycasting sample the volume successively along a viewing ray and calculate a solution of light transfer in the absence of scattering events. If the volume data set is represented by a 3D texture, however, we have the freedom to reflect the ray into an arbitrary direction at any point inside the volume. Random directions can be pre-computed, stored in additional texture images and accessed via randomized texture coordinates.

Such a simple implementation immediately leads to several problems:

- The pre-computed randomized ray directions do not account for the directions with dominant radiance contribution. The convergence is slow, because many rays are wasted in areas with only little contribution.
- Many rays are terminated due to low attenuation and do not contribute to the final image.
- The visual appearance of the volumetric object is hard to control by the artist.

The first problem is tackled by the use of importance sampling, as described in Section 5. The second and third problems are solved by multi-layer rendering technique which controls the reflection and transmission events. Details for the multiple passes are explained in Section 6.

5 Sampling Schemes

Importance sampling is an effective means of increasing the rendering performance by reducing the variance of the Monte-Carlo estimator. More samples are placed in regions of the parameter domain where the function to be integrated is expected to be large, while fewer samples are used in regions where only a small contribution to the integral is assumed. While software implementation have the flexibility to employ arbitrary probability distributions and tailor them to the specific integrand in Equation **??**, our GPU-based implementation must work with the same probability distribution for all scattering events due to the parallel nature of fragment programs.

We decided to use random directions uniformly distributed on the unit sphere as basis and employ simple but effective strategies to omit regions with only little contribution according to the phase function or BSDF. To avoid the necessity to account for different probability distributions $p(x)$, we restrict ourselves to uniform distributions.

For a fast access to randomized direction vectors from within a fragment shader, we use a pre-computed set of random value triplets representing points uniformly distributed on the unit sphere. The pre-computed random vectors are stored in a 3D texture.

The random directions obtained from the texture can directly be used to sample the phase function. For diffuse, surface-like reflection, however, it is necessary to restrict the random directions to a hemisphere centered around a given unit vector $\mathbf{n}$. We can easily generate such samples by negating all random vectors outside the given hemisphere,

$$\mathbf{r}_H(\mathbf{n}) \quad = \quad \mathrm{sgn}(\mathbf{n} \cdot \mathbf{r}_S)\mathbf{r}_S, \tag{6}$$

with sgn being the signum function.

For efficiently sampling a specular Phong lobe, we need to focus the sampling directions to a narrow cone centered around a given direction of reflection. A simple way of focussing ray directions is to compute a weighted sum of the hemispherical random samples $\mathbf{r}_H$ and the direction of perfect reflection $\mathbf{h}$:

$$\begin{aligned} \tilde{\mathbf{r}}_P(\mathbf{h}) \quad &= \quad \alpha \cdot \mathbf{r}_H(\mathbf{h}) \; + \; (1-\alpha)\mathbf{h}. \\ \mathbf{r}_P(\mathbf{h}) \quad &= \quad \frac{\tilde{\mathbf{r}}_P(\mathbf{h})}{\|\tilde{\mathbf{r}}_P(\mathbf{h})\|} \end{aligned} \tag{7}$$

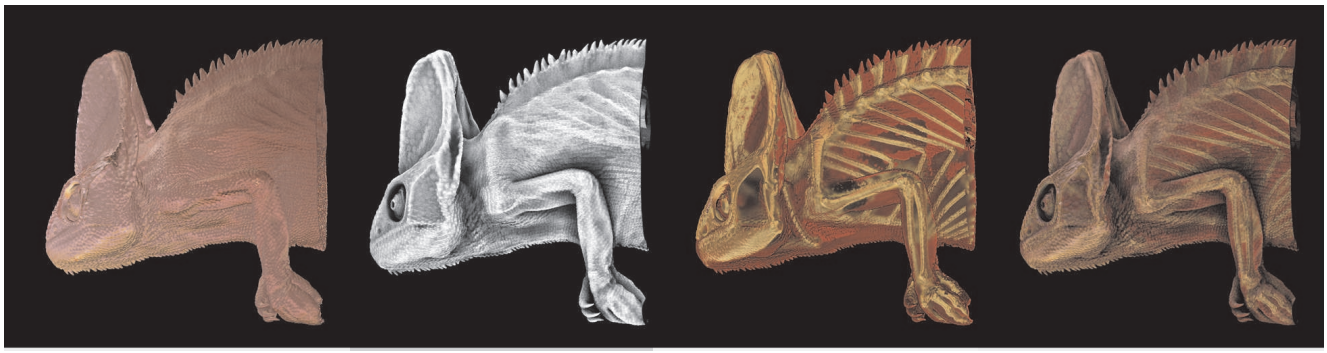

Figure 3. Results of the rendering passes. *Far left:* **Ambient occlusion pass for the first isosurface.** *Left:* **Beauty pass for the first isosurface.** *Right:* **Scattering Pass below the first isosurface** *Far right:* **Final Composite.**

The scalar weight α determines the maximum cone angle of scattering around the direction $\mathbf{h}$. A value $\alpha = 1$ means scattering in all directions on the hemisphere, while a value of $\alpha = 0$ results in the (non-randomized) ray direction perfectly focused into direction $\mathbf{h}$.

To determine an appropriate value of α for a given specular exponent s, we calculate the maximum reflection angle $\gamma_{\max}$, at which the specular term falls below a user-specified threshold T (say 0.1),

$$\gamma_{\max}(s) \;=\; \max\{\gamma \,|\, \cos(\gamma)^s > T\}. \tag{8}$$

Solving this equation yields

$$\gamma_{\max} \;=\; \arccos({}^s\sqrt{T}). \tag{9}$$

Figure 2 illustrates the relationship between the focus weight α and the angle γ. It is easy to derive an equation for α as function of the maximum angle $\gamma_{\max}$:

$$\alpha \;=\; \frac{1 + tan(\gamma_{\max} - \frac{\pi}{4})}{2} \tag{10}$$

6 Practice

For digital productions it turns out to be advantageous to decompose the final rendering into different independent layers. Although this often leads to physically incorrect solutions, it allows the artist to adjust contrast and brightness of different layers without the necessity to recompute a single frame of footage.

First Isosurface Pass: In practice, we calculate the first hit of the ray with the isosurface. The first pass is a beauty pass which calculates local illumination for the first isosurface. We estimate the gradient vector using central differences and sample the environment cube map multiple times to calculate local diffuse and specular illumination.

The leftmost image in Figure 3 shows the result of the beauty pass for the UTCT Veiled Chameleon data set illuminated by the LDR-version of the Grace cathedral environment map (courtesy of Paul Debevec, www.debevec.org). Since light is additive, for maximum flexibility in the compositing step, we can as well render separate diffuse and specular passes. This allows us to adjust the specular and diffuse reflection coefficients later in the compositing step.

Ambient Occlusion: For rendering soft shadows, ambient occlusion [10] is an essential way to fake global illumination effects, resulting in images similar to radiosity calculations, but at considerably lower computational cost.

For an ambient occlusion pass we need the position and gradient vector generated in the previous passes. For each pixel we cast several rays across the hemisphere centered around the gradient direction. The rays are traced a few steps only using a large step size to determine whether they hit the isosurfaces again or not. The percentage of rays which do not hit the geometry are stored as grayvalue in the frame buffer. The seconf image in Figure 3 shows the result of the ambient occlusion pass.

Scattering: In practice, a single scattering pass turned out to be sufficient in most cases (Figure Figure 3, 3rd image). We start a transmissive ray at the hit point with the first isosurface. The direction is scattered within a Phong lobe around the refracted vector. We trace this ray by accumulating the attenuation factors until it hits the second isosurface. At this second hit point, the gradient vector is estimated and the ray is reflected randomly into the hemisphere centered around the gradient direction. The ray is traced with attenuation but without further scattering until it leaves the volume.

A faster, but less accurate version of the scattering pass is based on the assumption that the attenuation which is accumulated from the eye point to the second hit point is an appropriate estimate for the attenuation from the hit point

Pass	Samples	Time/msec
First Hit pass	-	45–50
Iso Beauty (P)	16 spec., 8 diff.	80–170
Amb.Occ. (P)	32 rays, 10 steps	120–170
Sub.Scat. (P)	4 prim., 4 sec.	210–470
Final (P)	see individual passes	598–690
Iso Beauty	128 spec., 64 diff.	274–349
Amb.Occ.	128 rays, 20 ray steps	1450–1876
Sub.Scat.	64 prim., 16 sec.	5968–8771
Final	see individual passes	7156–9203

Table 1. Performance measurement for different passes in preview (P) and final quality (512^3, $16bit$ data).

back to the outside. Hence, we can square the accumulated attenuation and directly sample the environment map in the reflected direction without tracing the ray any further. The final composite is displayed in the rightmost image in Figure 3

7 Results and Conclusion

The presented Monte-Carlo volume raycasting approach has been implemented using Cg (nv40 profile) and OpenGL on an NVidia Geforce 8800 GTX graphics board with 768 MB video memory. The GPU-based implementation of the Monte-Carlo raycaster does not maximize the efficiency using stratified sampling or importance sampling, like many software implementations do. Nevertheless, the high parallel architecture of the GPU generates images at considerable speed.

The convergence of the proposed technique greatly depends on the phase function used. We have experimented with different phase functions and found that allowing scattering events to happen at every point inside the volume leads to extremely slow convergence and is thus computationally not feasible. The resulting images are hardly predictable for the artist, which leads to visual parameters being impossible to control in practice. This is the reason, why we suggest restricting the scattering events to a limited set of isosurfaces.

The performance of the different passes is shown in Table 1. Performance was measured both for preview (P) and final quality. The results show that preview renderings of the single passes are possible at interactive frame rates, which is important to adjust visual parameters in practice. The results of the preview passes are still noisy, but the image quality is good enough to get the visual impression of the final result.

The first conclusion we draw is that, in practice, the rendering of volumetric objects with observable internal structures, such as the tomographic scans used throughout this paper, requires considerably different strategies than rendering typical translucent materials, such as clouds, milk and skin, where scattering is rather homogenous. The method described in this paper is meant as a proof of concept.

Another conclusion we draw from our experiments is that scattering passes must be tailored to the desired visual effect. Multiple scattering layers may be used and viewing rays may be transmitted through the entire volume to account for translucency effects for backlit objects and the like. The described techniques integrate well in the layered shading framework known to visual artists. Since the rendering parameters are derived from the Phong model, they are intuitive to control and already familiar to most artists.

References

[1] K. Engel, M. Hadwiger, J. Kniss, C. Rezk-Salama, and D. Weiskopf. *Real-Time Volume Graphics*. AK Peters, Ltd., 2006.

[2] M. Hadwiger, A. Kratz, C. Sigg, and K. Bhler. Gpu-accelerated deep shadow maps for direct volume rendering. In *Proc. Graphics Hardware*, 2006.

[3] H. W. Jensen, J. Arvo, P. Dutre, A. Keller, A. Owen, M. Pharr, and P. Shirley. Monte carlo ray tracing. In *ACM SIGGRAPH Course Notes 44*, 2003.

[4] J. Kniss, S. Premoze, C. Hansen, and D. Ebert. Interative Translucent Volume Rendering and Procedural Modeling. In *Proceedings of IEEE Visualization*, 2002.

[5] J. Krüger and R. Westermann. Acceleration Techniques for GPU-based Volume Rendering. In *Proceedings of IEEE Visualization 2003*, pages 287–292, 2003.

[6] C. Rezk-Salama, K. Engel, M. Bauer, G. Greiner, and T. Ertl. Interactive Volume Rendering on Standard PC Graphics Hardware Using Multi-Textures and Multi-Stage Rasterization. In *Proceedings of ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware*, 2000.

[7] S. Röttger, S. Guthe, D. Weiskopf, and T. Ertl. Smart Hardware-Accelerated Volume Rendering. In *Procceedings of EG/IEEE TCVG Symposium on Visualization VisSym '03*, pages 231–238, 2003.

[8] L. Szirmay-Kalos. Monte-carlo methods in global illumination. http://www.iit.bme.hu/ szirmay/script.pdf, 1999. Script, Institute of Computer Graphics, Vienna University of Technology.

[9] O. Wilson, A. V. Gelder, and J. Wilhelms. Direct Volume Rendering via 3D-textures. Technical Report UCSC-CRL-94-19, Univ. of California, Santa Cruz, 1994.

[10] S. Zhukov, A. Iones, and G. Kronin. An ambient light illumination model. In *Proc. Eurographics Rendering Workshop*, 1998.