# Real-time subsurface scattering on the GPU

Antoine Bouthors        Eric Bruneton        Fabrice Neyret        Nelson Max

**Figure 1:** *Marble statue without* (left) *and with* (right) *subsurface scattering.*

## Abstract

We present a GPU algorithm that computes subsurface light transport in real time on arbitrary animated meshes. We evaluate both single scattering and multiple scattering, by using piecewise linear and ring-based approximations of the surface in the fragment shader. We demonstrate our technique on animated meshes at 60 fps.

**Keywords:** scattering, subsurface, GPU, dipole approximation

## 1 Introduction

Several approaches have been presented to bring [Jensen et al. 2001] to real-time. However, all of these approaches either require some precomputation (such as scattering links [Carr et al. 2003], discretized precomputed integrals [Hao and Varshney 2004], mesh-based hierarchy [Mertens et al. 2003b]) or do not reproduce all the possible features (*e.g.,* only local scattering and image-based irradiance maps [Mertens et al. 2003a], planar surface assumption [Mertens et al. 2003a; François et al. 2006], single scattering only [François et al. 2006]). We get rid of precomputations by computing light transport on the fly. We avoid the planar surface assumption by sampling the surface at rendering time at chosen places. We compute multiple scattering using an approximation suited for the GPU.

Before each frame, we compute a depth map from the light point of view, recording the depths $z_l$ of points on the lit surface. In the following, we are dealing with the shading of the point $p$ on the visible surface, for each pixel to render.

## 2 Single scattering

Geometry has a great influence on single scattering, especially in highly curved areas (*e.g.,* face features) and on shadow boundaries. To account for it, we approximate it by a piecewise linear surface. We take exponentially spaced samples $p_i$ inside the volume along the view ray and compute the lengths $l_i$ of the exit rays towards the light by reading $z_l(p_i)$ (fig. 2, left). We assume the surface is planar between each sample. We integrate the single scattering equation analytically on the GPU for each segment.

## 3 Subsurface scattering

To evaluate subsurface scattering, we need to integrate $\int_x I(x)R_d(\|p - x\|)dx$ over the neighboring surface of $p$, where $I$ is the irradiance function and $R_d$ the subsurface light transport function [Jensen et al. 2001]. This can also be done on the GPU, in a Monte-Carlo manner: we evaluate $I(x_i)$ using the shadow map $z_l(x_i)$ and compute $R_d(\|x - x_i\|)$ on the GPU, for
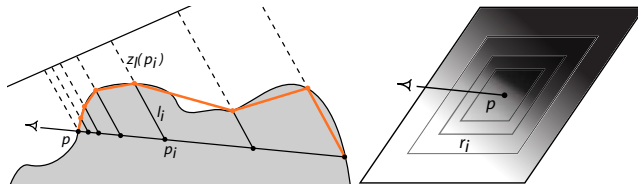


**Figure 2:** Left: *Single scattering.* Right: *Multiple scattering. Integration over the rings smoothes the shadow boundaries.*

a set of samples $x_i$ chosen using importance sampling. However, while this is feasible on the latest graphics hardware, it requires too many texture accesses on low-end GPUs.

To accelerate the process, we rely on the fact that $R_d(\|p - x\|)$ is constant for all $x$ on a ring around $p$. We compute $R_d(r_i)$ for a set of concentric rings $i$ of radius $r_i$ around $p$ (fig. 2, right). We evaluate $I_i$ for each ring $i$ by taking advantage of MIP-mapping: the integral of a value over a square ring is the difference between two MIP-map levels. Here we need to MIP-map the result of a shadow map comparison. We rely on Variance Shadow Maps [Donnelly and Lauritzen 2006].

## 4 Implementation and Results

Figure 1 and the accompanying videos show the effect of using our method vs. standard BRDF rendering on a marble model. One can see that the light passes through the thin parts of the model and shadow boundaries are smoother. We implemented both the sample-based and the ring-based technique. Framerate varies between 30 and 100 FPS on 1K-20K triangles models on an nVidia G80 in 800x600 (75% of the screen filled) for the ring-based method and 50% slower for the sample-based method with 30 samples per pixel. The main limitation is on quality. Due to the use of shadow maps, one can sometimes see aliasing artifacts and Mach bands on the ring-based method. The sample-based method is less prone to these problems but is subject to noise if too few samples are used.

## 5 Conclusion and Future Work

Our technique shows subsurface scattering on arbitrary animated meshes in real-time and allows for the animation of the light source(s) and viewpoint. This work is fully compatible with textures and the next obvious step is to apply the texturing method from [Jensen et al. 2001], which is straightforward. Using multilayered materials and textures would definitely improve realism. The quality of the rendering (less aliasing) should also be improved.

## References

CARR, N. A., HALL, J. D., AND HART, J. C. 2003. GPU algorithms for radiosity and subsurface scattering. In *Graphics Hardware'03*, 51–59.

DONNELLY, W., AND LAURITZEN, A. 2006. Variance shadow maps. In *I3D'06*, 161–165.

FRANÇOIS, G., PATTANAIK, S., BOUATOUCH, K., AND BRETON, G. 2006. Subsurface texture mapping. In *SIGGRAPH Sketches'06*, 172.

HAO, X., AND VARSHNEY, A. 2004. Real-time rendering of translucent meshes. *ACM Trans. Graph. 23*, 2, 120–142.

JENSEN, H. W., MARSCHNER, S. R., LEVOY, M., AND HANRAHAN, P. 2001. A practical model for subsurface light transport. In *SIGGRAPH'01*, 511–518.

MERTENS, T., KAUTZ, J., BEKAERT, P., REETH, F. V., AND SEIDEL, H.-P. 2003. Efficient rendering of local subsurface scattering. In *Pacific Graphics'03*, 51.

MERTENS, T., KAUTZ, J., BEKAERT, P., SEIDELZ, H.-P., AND REETH, F. V. 2003. Interactive rendering of translucent deformable objects. In *EGRW'03*, 130–140.