BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2024
http://excel.fit.vutbr.cz

Realistic Simulation of Complex Materials

Matěj Toul*

Abstract

The aim of this paper is to simulate and render complex real world materials (liquids, gases, wax and other materials), often referred to as participating media. The priorities of this simulation are both physical plausibility and computational efficiency. The simulation is implemented using a path tracer for results as close to photorealism as possible. Inside the path tracer, light interactions are handled using BSSRDF and all the calculations are highly parallelized using GPU and Vulkan API. Using these techniques, a wide range of materials can be accurately simulated based purely on their measured light scattering properties. The output of this work can be used to plausibly simulate complex light transport across different scenes and materials, be it for scientific reasons or for visual entertainment.

*xtoulm00@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

With light-speed advancements in computer capabilities, the expectations for what looks realistic are getting higher and higher as well. The methods for fast, but still accurate simulations are required today, more than ever. This paper presents one of the possible ways to meet these expectations.

Light transport simulation in computer graphics was, for the longest time, quite difficult – even for simple materials. Nowadays, more computationally intensive methods are becoming viable for efficient rendering and advanced phenomena, such as *subsurface scattering*, can be simulated in a matter of seconds.

Path tracing [1] is generally the go-to approach for any kind of light transport simulation. The beauty is in its simplicity – the core algorithm derives from the behavior of light itself and at a high-level, light transport can be described by simple laws of reflection and transmission. Going further, there are more phenomena, mostly coming from the unusual ray-wave nature of light. Still, the main problem is usually the volume of computations, not their complexity.

One can imagine the path tracing algorithm as a wrapper, as its job is barely to send light rays into the 3D scene and follow them across interactions. Everything that happens during those interactions, but also between them, is highly adaptable and varies field to field.

Bidirectional scattering distribution functions (BSDF) are used as means to resolve light interactions at the boundaries of materials. There are many implementations. Some draw from the original *Phong model* [2], others follow more advanced theories, such as microfacets [3]. But all of them solve the same core problem – where the light should continue after interaction and how it should be affected by the interaction.

The method presented in this paper uses path tracing with simple opaque interactions handled using the Lambert model. For the simulation of more complex materials, subsurface scattering is taken into account, resulting into a method commonly known as **BSSRDF**. This is one of many steps that can be taken to improve photorealism.

Inside the medium, four phenomena occur – absorption, out-scattering, emission and in-scattering, but the method only takes into account the two former. While this is a limitation, emission is rare in natural materials and in-scattering can be closely approximated with enough out-scattering samples.

2. Path tracer

The path tracer implementation consists of several parts that can be seen in the Figure 1. At first, a light ray is sent from the camera, looking for something to interact with – a regular solid surface, transmissive participating medium or nothing.

Solid surface interactions are handled with a simple Lambert BSDF, represented by the term $L(\mathbf{x}_s, \vec{\omega})$ in the part of Equation 1 behind the integral. The value T_r here represents the *transmittance* – simply put, it's the product of all the medium interactions.

If a medium is hit, we first need to determine if the ray enters it. That can be done using **Fresnel equations**. They give the percentage of light that is reflected and transmitted, based on the angle of incoming light and medium properties. We then pick one of the options on random. The same interaction repeats upon leaving the medium.

3. In the Media

The transmitted rays enter the medium under the angle given by the well-known Snell's law. Inside, a different kind of interactions occur. First, a path to the light source is traced, to see if the sampled point is in shadow or not.

Next, the integral part of Equation 1 needs to get resolved. To do so, absorption and out-scattering phenomena, among other things, need to be taken into account.

Absorption Figure 2a causes the incoming light to lose some of its properties (mostly color) to the material. The absorbance of media is given by *absorption coefficient*, normally of spectral value, for the purpose of rendering defined as RGB value. Absorption affects how the medium itself is rendered onto the screen. It follows a simple principle – high red absorption results into cyan medium.

Out-scattering Figure 2b in some ways is a counterpart to absorption. The *scattering coefficient* defines how much of each color component gets scattered out of the point. While this has the opposite effect to absorption – high red scattering results into red medium, the scattering coefficient also determines how much light gets through the medium.

Highly scattering media, such as milk, can end up almost completely opaque. This happens because scattering coefficient also affects how often medium interactions happen inside the media. As more interactions occur, the incoming light ray is changed multiple times. Anything inside or behind the media is therefore blurry or completely invisible.

Phase function Figure 2c is a probability function that determines the new ray direction after each of the medium interactions. Anisotropy of the medium is taken into account here, weighing the phase function forwards or backwards. In this simulation, the Henyey-Greenstein phase function [4] is used.

4. Implementation

The whole method is implemented on the GPU, using low-level and highly efficient **Vulkan API**. In graphics programming using the GPU, memory transfers are often the most expensive processes. That's why the implementation follows a simple schema shown in the Figure 3.

First, Vulkan is set up. To speed up this process, wrapper functions included in Nvidia's **nvpro_core** library are used. The setup includes parsing input files in Wavefront OBJ format into specialized acceleration structures, optimized for fast ray traversal and also parsing of JSON file defining the media. One can use a provided script to create these files, or create them manually, as they follow a very simple structure.

The acceleration structure data, along with the media definitions get transferred from host (CPU) to the device (GPU). On the GPU, the whole path tracer, including the submodules explained above, is implemented.

While the implementation is used for single renders, the output is stored in a device-only texture that then gets copied back over to host. This way, the original output texture never leaves device, minimizing GPU downtime and allowing for simple future extensions to support video and on powerful machines perhaps even real-time rendering.

5. Conclusions

While the solution produces very plausible results quickly even on medium tier hardware, the room for extensions and improvements is big. For example, *heterogeneous media*, where absorption and scattering coefficients spatially vary, can be implemented. For even more accurate results with current media, one could implement various BSDFs for different kinds of materials (dielectric, metal, etc.).

The renders seen in Figures 4a-c were rendered using the laptop variant of Nvidia GeForce RTX 3060 graphics card, set up to render at 2048 samples and tracing up to 32 interactions each.

Acknowledgements

I would like to thank my supervisor Ing. Michal Vlnas for his invaluable help in proper understanding of all the aspects of methods used in this paper.

References

[1] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, aug 1986.

[2] Bui Tuong Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, jun 1975.

[3] Alexander V Prokhorov and Leonard M Hanssen. Algorithmic model of microfacet brdf for monte carlo calculation of optical radiation transfer. In *Optical Diagnostic Methods for Inorganic Materials III*, volume 5192, pages 141–157. SPIE, 2003.

[4] Louis G Henyey and Jesse Leonard Greenstein. Diffuse radiation in the galaxy. *Astrophysical Journal, vol. 93, p. 70-83 (1941).*, 93:70–83, 1941.