15668 Final Project: SGGX Microflake Distribution and Layered Material with Statistical Operator

WEISHENG LI

Fig. 1. Rendering Competition Submission (1024 spp, render time: 1h 40min)

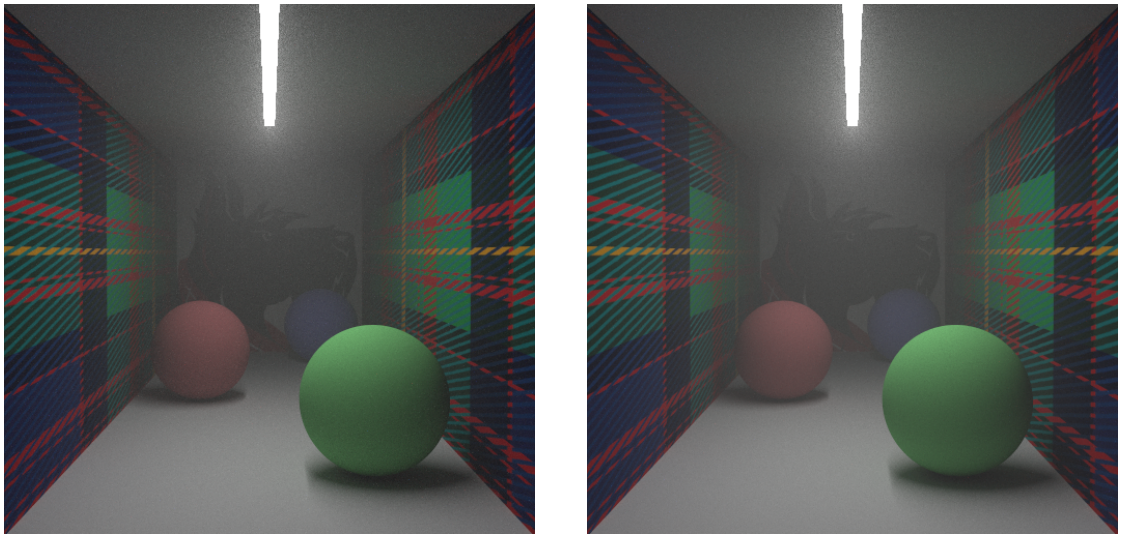
1 INTRODUCTION

Two 6-points intermediate features are implemented as the final project: Anisotropic volume rendering using symmetric GGX distribution and layered material with statistical operator. Both features are implemented on DIRT. The source code is hosted on GitHub (https://github.com/Weisheng-Li/PBR_Final_Project). All test scenes and images can be found on the release. The README file in the GitHub repo details all code I wrote for the project.

2 SGGX MICROFLAKE DISTRIBUTION

This feature is implemented based on the SGGX Microflake distribution paper[2]. My implementation only supports uniform anisotropic medium with surface-like normal distribution. For simplicity of phase function evaluation, all microflakes are assumed to be specular reflectors.

Author's address: Weisheng Li, weishenl@andrew.cmu.edu.



(a) Anisotropic-cornell-box.json with 1024 spp (b) Isotropic-cornell-box.json with 1024 spp

Fig. 2. Comparison of rendering result of homogeneous isotropic medium with same density, albedo, and number of samples (a) The rendering result of SGGX distribution. The corresponding JSON file (anisotropic-cornell-box.json) is included in the submission. (b) The reference image from PA4, isotropic-cornell-box-1024-spp.png

2.1 Implementation Details

Due to the homogeneous nature of the medium, the S matrix that statistically defines the orientation distribution of microflakes is a constant matrix instantiated during JSON parsing. Users can either specify a normal direction to create a surface-like normal distribution or leave the normal field empty to create a uniform normal distribution, which becomes an isotropic medium. The S matrix will be instantiated accordingly. For the isotropic case, S is an identity matrix. For the surface-like NDF, S is defined by equation (14) in the original paper.

Using the S matrix, I can then compute σ_t and σ_s, both now dependent of ω_i. They are computed by first evaluating the projection onto ω_i using S, and multiplying the projected area by density and optionally the albedo.

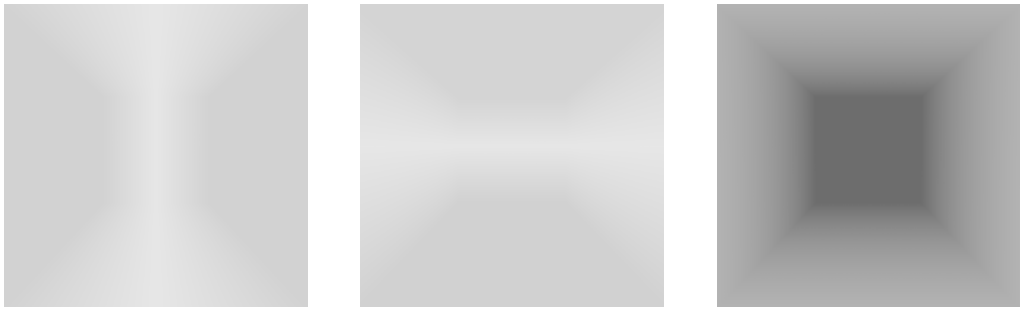
Additionally, a special phase function needs to be defined to work with the SGGX-based medium. Its sampling and evaluation procedure is described in Algorithm 1 and Equation (24) of the original paper.

I also put some extra effort into implementing a few 3×3 matrix operations (inverse, determinant etc) due to the lack of support for the 3×3 matrix in DIRT.

2.2 Correctness Validation

The first part of the correctness validation is using SGGX distribution to render an isotropic medium and compare the result to the same scene we rendered using the isotropic method in programming assignment 4. As shown in Figure 2, the results look indistinguishable.

The second part of the correctness validation is for surface-like normal distribution. In my experiment, it tends to generate unnatural visual artifacts that are difficult to reason about, so I designed a very simple scene for testing (see Figure 3). This scene consists of a Cornell box with emissive surfaces and anisotropic medium inside. The max bounce



(a) x-axis aligned normal (horizontal) (b) y-axis aligned normal (vertical) (c) z-axis aligned normal (parallel to view direction)

Fig. 3. Emissive Cornell-box with different surface-like normal distribution

is set to 0 to prohibit any light bounce. I set the normal of surface-like normal distribution to align with x, y, and z axis respectively. The expected behavior is that when the normal is parallel to the viewing direction, I should observe maximum projected areas of microflakes along the viewing direction, thus the scene would be uniformly darker (Figure 3 (c)). When the normal is orthogonal to the viewing direction, the projected area should be 0. As the angle becomes smaller, the projected area would become larger. This behavior corresponds to the vertical and horizontal bright bar in Figure 3 (a) and (b) and the darker regions near the sides.

3 LAYERED MATERIAL WITH STATISTICAL OPERATOR

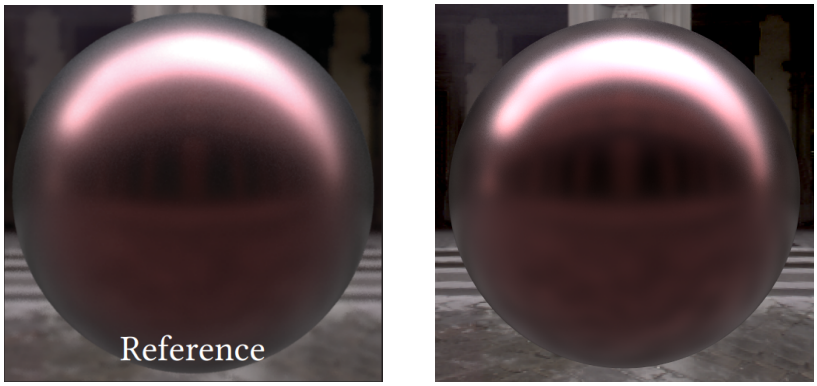
This feature is implemented based on the paper with the same name [1]. I also use the Mitsuba implementation that comes along with the paper as a reference. My implementation can support stacking multiple layers of dielectric coating on top of a conducting surface, but it doesn't support participating media between interfaces, fully dielectric layers, or layers with different depths due to the limited time. Total internal reflection is also ignored, so my implementation is not energy-conserving.

3.1 Implementation Details

The first component required by this feature is a precomputed 4-D FGD texture. I use the FGD texture released along with the paper and write an FGD texture class to support interpolated FGD texture lookup.

The second component is the adding-doubling computation, which takes in the z-component of ω_i (in local coordinate) and the layered material specification. Only the z-component is needed since all layers are assumed to be isotropic. The layered material specification consists of the roughness and complex IOR of each layer, where the real part (eta) represents the IOR and the imaginary part (kappa) accounts for albedo. The output of the adding-doubling procedure is a vector of BRDF lobes, each representing the statistic of radiance exiting the surface after traveling through the first n layers. Theoretically, the adding-doubling computation can be precomputed for each layered material as well. It should be efficient to compute and store given that it's parameterized on only the z-component, but I didn't include it in my implementation.

The last component is writing the sampling, evaluation, and pdf procedure. These procedures are relatively simple and clearly articulated in the 6.1 Forward Model section of the paper. The most time-consuming part of implementing these



(a) Reference image from the paper, computed using stochastic evaluation of the layered structure

(b) Rendered by my implementation

Fig. 4. Emissive Cornell-box with different surface-like normal distribution

functions is writing procedures for sampling and evaluating Beckmann distribution and its corresponding shadowing term. I cannot reuse the Beckmann material we implemented before since the algorithm requires directly sampling and evaluating the normal rather than the outgoing direction. I copied the Beckmann distribution implemented in Mitsuba since it appears to be both efficient and numerically stable.

3.2 Correctness Validation

Correctness validation for this feature is very simple, as the author releases the specification of a few sample scenes shown in the paper. I verify the correctness of my implementation by using the same material parameters and compare the rendered image from my implementation to the reference image shown in the paper. (see Figure 4).

REFERENCES

[1] Laurent Belcour. 2018. Efficient Rendering of Layered Materials using an Atomic Decomposition with Statistical Operators. *ACM Transactions on Graphics* 37, 73 (2018), 1–15. https://doi.org/10.1145/3197517.3201289

[2] Eric Heitz, Jonathan Dupuy, Cyril Crassin, and Carsten Dachsbacher. 2015. The SGGX microflake distribution. *ACM Transactions on Graphics* 34, 48 (2015), 1–11. https://doi.org/10.1145/2766988