15-468 Final Project Report

Konwoo Kim
Carnegie Mellon University
konwook@andrew.cmu.edu

Abstract

We discuss the implementations, tests, and results of the features for our final project which include the GGX BSDF, decomposition tracking, spectral tracking, and spectral MIS. For each feature, we explain its implementation and integration within DIRT as well as tests and results to validate correctness. A final rendered image showcasing all of the implemented features is presented at the end. Code to reproduce all of the results in this report is available at `https://github.com/cmu-15-468/dirt-s22-konwook` *under the release titled Final Project.*

1. GGX

1.1. Implementation

Our GGX implementation is largely based on [3] and [4]. In particular, the distribution of normals we sample from is defined in [4] as:

$$D(_m) = \frac{\alpha^2}{\pi((\omega_g \cdot \omega_m)^2(\alpha^2 - 1) + 1)^2}$$

We importance sample this distribution of normals by analytically computing the inverse CDF. This gives spherical coordinates of the form:

$$\theta_m = \arccos \sqrt{\frac{1-\xi_0}{\xi(\alpha^2-1)+1}} \text{ and } \phi = 2\pi\xi_1$$

We additionally use the Smith masking-shadowing function as defined in [3] when computing the BRDF. To integrate this within DIRT, we create a new `Material` called `Material::GGX` which supports `GGX::scatter`, `GGX::sample`, `GGX::eval`, `GGZ::pdf`, `GGX::shlick`, `GGX::maskingShadowing`. The first four functions are necesasry to implement as part of the `Material` API while the last two are helper functions to compute the Shlick Fresnel approximation and Smith masking-shadowing function. We importance sample the normal distribution inside of `GGX::sample`, compute the probability distribution inside of `GGX::pdf`, and evaluate the reflectance in `GGX::eval` as:

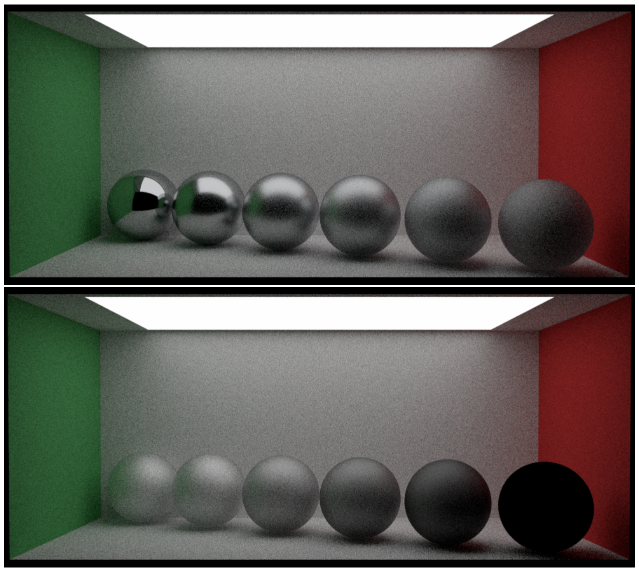

Figure 1. Top: GGX spheres with linearly increasing amounts of roughness from 0 to 0.75. Bottom: GGX spheres with fixed roughness of 0.5 and linearly increasing amounts of specular and albedo from $(0, 0, 0)$ to $(0.95, 0.95, 0.95)$.

$$\frac{F(\omega_i, \omega_m)G_2(\omega_i, \omega_o, \omega_m)|\omega_o \cdot \omega_m|)}{|\omega_o \cdot \omega_g||\omega_m \cdot \omega_g|}$$

where F is the Shlick Fresnel approximation, G_2 is the masking shadowing term, and ω_m, ω_g are the microfacet and geometric normals.

1.2. Results

We use a scene of a wide Cornell box with multiple spheres made of incrementally differing GGX materials. In particular, `Material::GGX` requires a `albedo, specular, roughness` where the first two terms are 3D vectors and the third term is a float. We ablate these parameters to produce a smoothly varying series of spheres and test if their visual appearance is as expected. The top image of Figure 1 shows that linearly varying the roughess of the sphere while keeping the albedo and specular terms fixed produces increasingly rough and less shiny spheres. The bottom image of Figure 1 shows that linearly varying the specular and albedo parameters while fixing the roughness produces different kinds of metallic appearances. Both image align with the expected result of varying different GGX paramters.

2. Analog Decomposition Tracking

2.1. Implementation

Our implementation of decomposition tracking follows that of [1] where it was originally introduced. Given an arbitrary heterogeneous media to render, we decompose it into a homogeneous control component and heterogeneous residual component. Unless otherwise specified, we define our control volume coefficients to be 10% of the corresponding coefficients in the original volume. We initially sample the control component as we can analytically compute the free-flight time as

$$t_c \leftarrow \frac{-\log(1-\zeta)}{\mu_t^c}$$

where μ_t^c is the extinction coefficient of the original volume. Then, we repeatedly sample a free path in the residual component according to:

$$t_r \leftarrow t_r - \frac{-\log(1-\psi)}{\bar{\mu} - \mu_t^c}$$

where $\bar{\mu}$ is the free-path sampling coefficient i.e. the sum of the null-collision coefficient and extinction coefficient of the original volume. Note that decomposition tracking relies on the free-path sampling coefficient being a majorant on the original extinction coefficient. We continue sampling the residual component until we pass the control sample. Then, we probabilistically classify the collision like in weighted delta tracking according to whether it corresponds to a absorption, scattering, or null event. We repeat this until we get a real collision. Unlike [1], we don't sample a new ray direction from the phase function in the case that we don't have a true collision.

Finally, to integrate this within DIRT, we refactor the `Medium` interface. We create a templated `Sigma` struct to store the various coefficients and so that we can easily extend it for spectral tracking. We also create a `DecompMedium` which inherits from `Medium` and has separate `Sigma` members for the control and residual components. Apart from the constructor, `DecompMedium` also supports `DecompMedium::Tr`, `DecompMedium::Sample`, `DecompMedium::density`. These methods compute the transmittance, sampled medium interaction, and density respectively.

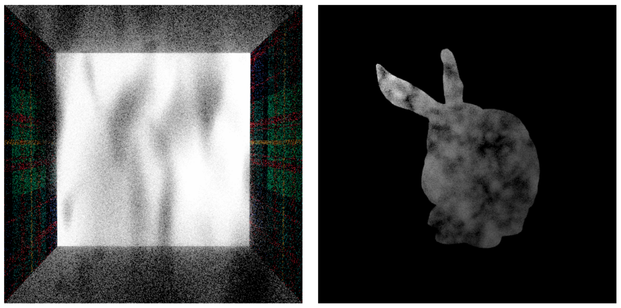

Figure 2. Cornell box and bunny scene rendered with 0% control ratio to verify equivalence of decomposition and delta tracking in the limit.

2.2. Results

To test decomposition tracking, we first verify that it can replicate the behavior of normal delta tracking by setting the proportion of the control component to be 0%. Figure 2 shows that we are able to do this as we recreate the Cornell Box and bunny scenes from the last programming assignment. Following this, we show results with using a non-zero control proportion and varying parameters like the control ratio as well as the null-scattering coefficient. Figure 3 shows two scenes with different control proportions and two scenes with different null-scattering coefficients. As expected, as we decrease the control proportion, the density of the visual media increases and as we increase the null-scattering coefficient, it decreases.

Finally, we compare the efficiency of decomposition tracking to delta tracking. We find that on average across 5 scenes rendered at 16 spp, decomposition tracking with a 10% control proportion and 1% null-scattering coefficient runs in 19.12 seconds while delta tracking runs in 20.27 seconds, so decomposition tracking does provide a small improvement in efficiency.

3. Spectral Tracking and MIS

3.1. Implementation

Our implementation of spectral tracking and MIS follows that of [1] and [2]. For spectral tracking, we extend the `Sigma` interface using a template so that spectrally varying coefficients can be specified. Then, spectral tracking proceeds like delta tracking but we iteratively update a vector of weights as we sample events:

$$\hat{w} \leftarrow \hat{w} \times \frac{\hat{\mu}_s(x)}{\bar{\mu} P_s(x)}$$

where the subscript refers to a scattering event and we use an analogous update for the null event. For the absorption event, we scale the weights similar and return an element-wise multiplication of the weights with the

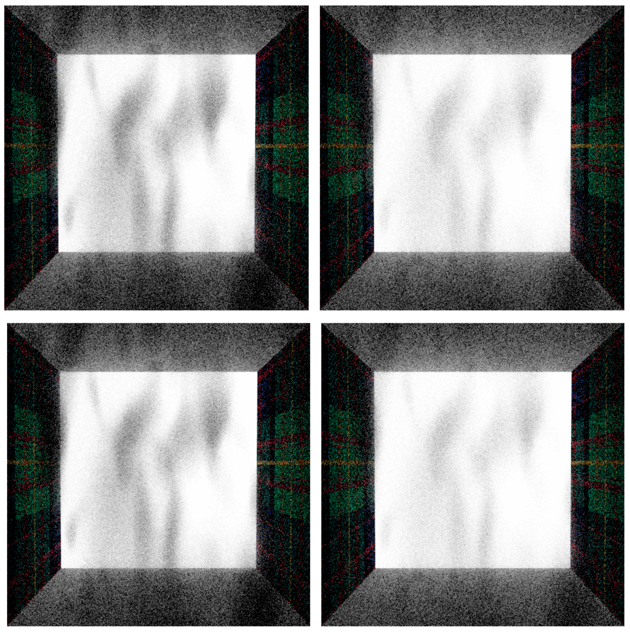

Figure 3. Top: Decomposition tracking with null extinction coefficient set to 1% and 10%. Bottom: Decomposition tracking with control proportion set to 10% and 50%.

ratio of the scattering and extinction coefficients of the original volume. We integrate this within DIRT by creating a `SpectralMedium` which inherits from `Medium` and supports the same functions that `DecompMedium` does as well as `SpectralMedium::SpectralTr`, `SpectralMedium::SpectralSample` which return colors instead floats. We integrate these with `SpectralVolpathTracer` and `SpectralVolpathTracerNEE`. Finally, we compare this method to a naive spectral tracking method which simply runs delta tracking for each channel and concats the results together. We denote this as `NaiveSpectralMedium`.

For MIS, we follow the method in [2] and implement the unidirectional spectral multiple importance sampling method. Similar to spectral tracking, we maintain per-component path contributions but we initially select a random color component before sampling medium interactions. This is implemented in `SpectralMISMedium`. Spectral MIS wasn't fully debugged so there are no results.

3.2. Results

To verify correctness, we show that we can render heterogeneous media of different colors using spectrally varying coefficients. Figure 4 shows a comparison between the naive delta tracking method and spectral tracking across three different scenes with different spectral coefficients. Spectral tracking is able to represent the color in the media much more effectively compared to naive delta tracking. However, spectral tracking does exhibit fireflies and darker artifacts on the edges of the media. Spectral tracking is also much more efficient than delta tracking. On average, spectral tracking takes 22.37 seconds to render the scene in Figure 4 while delta tracking takes 57.72 seconds. This speedup is due to the vectorized weight update while delta tracking has to run 3 separate iterations for each channel.

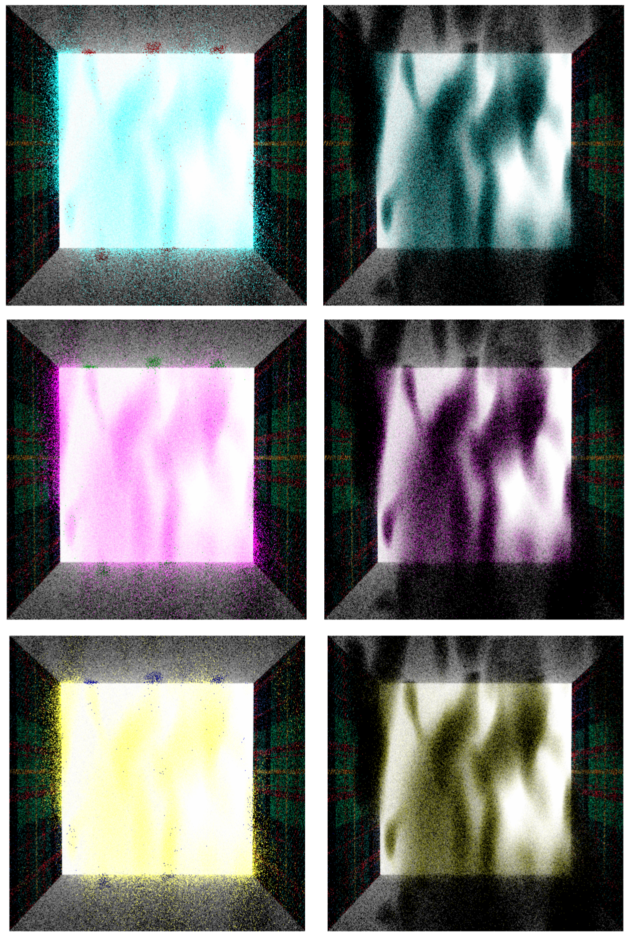

Figure 4. Spectral tracking (left) vs naive delta tracking (right) on different scenes with spectrally varying absorption, scattering, and extinction coefficients.

4. Final Image

Our final image shown in Figure 7 incorporates all of the features described above. We create an axe made of GGX material, position it in a room with textured walls and a diffuse light on the ceiling, and create a light, blue-green heterogeneous media in front of it which we render with spectral tracking and multiple importance sampling. An earlier version of this image shown at the final presentation utilized decomposition tracking and no spectral tracking.

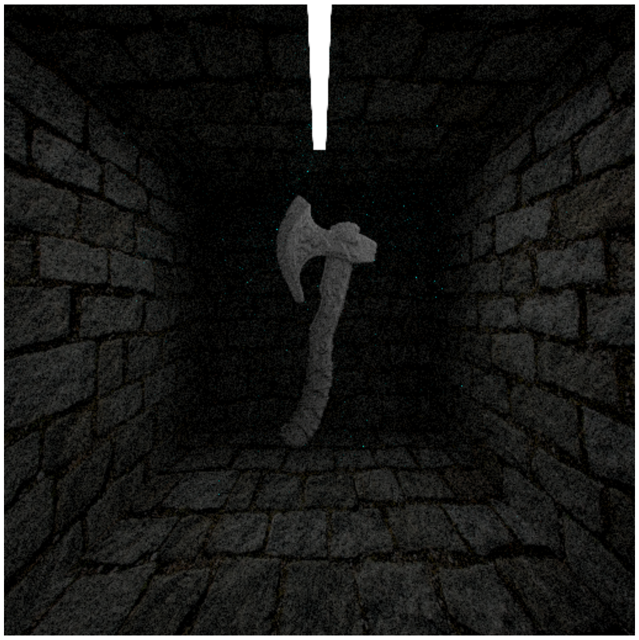
Figure 5. Axe mesh positioned in a textured room with a sparse, lightly colored heterogeneous media in front. Rendered at 1024 spp.

References

[1] Peter Kutz, Ralf Habel, Yining Karl Li, and Jan Novak. Spectral and decomposition tracking for rendering heteregenous volumes, 2017.

[2] Bailey Miller, Iliyan Georgiev, and Wojciech Jarosz. A null-scattering path integral formulation of light transport, 2019.

[3] Joe Schutte. Importance sampling techniques for ggx with smith masking-shadowing, 2018.

[4] Bruce Walter, Stephen Marschner, Hongsong Li, and Kenneth Torrance. Microfacet models for refraction through rough surfaces, 2007.