Graphics and Imaging Lab

Assignment #3

Monte Carlo-Based Materials

Modeling and Simulation of Appearance

Master on Robotics, Graphics and Computer Vision

Important: Before you start, download the provided patch to the code (`patch_P3.zip`) and extract it on the root folder of NORI. It includes some helper functions, as well as a few new scenes.

Overview

In this assignment, you will depart from the diffuse shading you used in previous assignments, and implement more complex appearance models, which you will use for rendering scenes under **direct illumination**. After completing this assignment, you will be able to efficiently render microfacet-based materials. For that, in addition to new appearance models, you will implement a set of Monte Carlo integrators based on sampling the material, which you will finally combine with the emitter sampling integrator from Assignment 2 using multiple importance sampling (MIS).

Similar to Assignment 2, your new integrators will compute the local illumination integral discussed in class:

$$\begin{aligned} L_o\left(\mathbf{x}, \omega_o\right) &= L_e\left(\mathbf{x}, \omega_o\right) + L_r\left(\mathbf{x}, \omega_o\right) \\ &= L_e\left(\mathbf{x}, \omega_o\right) + \int_\Omega L_i\left(\mathbf{x}, \omega_i\right) f_r\left(\mathbf{x}, \omega_i, \omega_o\right) \cos\theta_i \, \mathrm{d}\omega_i, \end{aligned} \tag{1}$$

where L_o is the outgoing radiance at a surface position $\mathbf{x}$ in direction ω_o as the sum of L_e and L_r the emitted and reflected radiance at $\mathbf{x}$ respectively. The reflected radiance is defined as the integral on the hemisphere Ω of the incoming radiance L_i at $\mathbf{x}$ from direction $\omega_i \in \Omega$, times the BRDF f_r. The angle θ_i is the angle between ω_i and the surface normal at $\mathbf{x}$. Generally, L_i and L_o are related to each other using the ray tracing operator $\mathrm{r}(\mathbf{x}, \omega)$, which returns the nearest surface position visible along the ray with origin $\mathbf{x}$ and direction ω, i.e.

$$L_i(\mathbf{x}, \omega) = L_o(\mathrm{r}(\mathbf{x}, \omega), -\omega), \tag{2}$$

and the above integral is thus defined recursively. In this assignment, we focus on direct illumination only and therefore truncate the recursion after the first light bounce. This means that the integral is now given by

$$L_o\left(\mathbf{x}, \omega_o\right) = L_e\left(\mathbf{x}, \omega_o\right) + \int_\Omega L_e\left(\mathrm{r}\left(\mathbf{x}, \omega_i\right), -\omega_i\right) f_r\left(\mathbf{x}, \omega_i, \omega_o\right) \cos\theta_i \, \mathrm{d}\omega_i. \tag{3}$$

During rendering, NORI will approximate this integral with its Monte Carlo estimate. Note that the summation and averaging is taken care of by NORI itself, and you do not need

to write code for this. The integrators you will implement in this assignment only need to compute the values of a single sample at a time.

Once again, we have prepared a set of scenes for you to test, though feel free to use new ones. All the scenes can be found in `./scenes/assignment-3`, which you can download from Moodle. In the rest of the assignment, we refer to the scenes in that particular folder; we skip the full path for simplicity.

1 Microfacet-based BRDF (40%)

Here you will implement two types of BRDFs, a rough conductor and a rough dielectric substrate (e.g. plastic). Both materials will be based on microfacet theory, so that the interface of the surface will be assumed to be formed by a set of tiny mirrors. Implement both of them in `microfacet.cpp`. Note that we have prepared a set of auxiliary functions in `include/nori/reflectance.h`, which you can use in your code.

1.1 Rough conductor (20%)

Follow the PBR textbook (Fourth Edition, Chapter 9.6) to implement a microfacet BDRF for conductors in the class `RoughConductor` (`microfacet.cpp`). For that, use the Beckmann distribution as your normal distribution function (NDF). Modify `RoughConductor::eval` to account for the following formula:

$$f_r\left(\omega_i, \omega_o\right) = \frac{D\left(\omega_h\right) F\left(\left(\omega_h \cdot \omega_i\right), R_0\right) G\left(\omega_i, \omega_o, \omega_h\right)}{4 \cos\theta_i \cos\theta_o}, \tag{4}$$

where the term F is the Fresnel reflection coefficient following the Schlick's approximation, and R_0 is the reflection at normal incidence. You may use the `Reflectance::fresnel()` function from `reflectance.h` to compute this term. The distribution function D is the Beckmann normal distribution function defined as

$$D\left(\omega_h\right) = \frac{e^{\frac{-\tan^2\theta_h}{\alpha^2}}}{\pi\alpha^2 \cos^4\theta_h}, \tag{5}$$

with α the parameter defining the roughness of the surface. Finally, G models the Beckmann's shadowing-masking term under the Smith approximation, defined as $G\left(\omega_i, \omega_o, \omega_h\right) = G_1\left(\omega_i, \omega_h\right) G_1\left(\omega_o, \omega_h\right)$, with

$$\begin{gathered} G_1\left(\omega, \omega_h\right) = \chi^+\left(\frac{\omega \cdot \omega_h}{\omega \cdot \mathbf{n}}\right) \begin{cases} \frac{3.535b + 2.181b^2}{1 + 2.276b + 2.577b^2} & b < 1.6 \\ 1 & \text{otherwise} \end{cases} \\ b = \left(a \tan\theta_v\right)^{-1}, \chi^+(c) = \begin{cases} 1 & c > 0 \\ 0 & c \leq 0 \end{cases}. \end{gathered} \tag{6}$$

where θ_v is the angle between the surface normal $\mathbf{n}$ and the ω_v argument of G_1. You can use the function `Reflectance::G1()` to compute G_1. Then, implement the sampling routines `RoughConductor::sample` and `RoughConductor::pdf` that you will use in the following part of the assignment. Use `Warp::squareToBeckmann` and its associated pdf in `Warp::squareToBeckmannPdf` from the previous assignment for sampling. Remember that similar to the emitters, the materials' sampling routiness return the **weight** of the sample $w(\omega_i) = f_r(\omega_i, \omega_o) / p(\omega_i) \cos\theta_i$.

You can test your implementation using the scene `./cbox/cbox_conductors_ems.xml`; the reference, computed for 1024 samples per pixel, can be found in `./references/`.

1.2 Rough substrate (20%)

Follow the PBR textbook (Third Edition, Chapter 8.5) to implement a microfacet BDRF for substrates in the class `RoughSubstrate` (`microfacet.cpp`). Again, use the Beckmann distribution as your normal distribution function (NDF). Modify `RoughSubstrate::eval` to account for the following formula:

$$f_r(\omega_i, \omega_o) = f_{\text{diff}}\ (\omega_i, \omega_o) + f_{\text{mf}}(\omega_i, \omega_o), \tag{7}$$

where the first term f_{diff} is the diffuse component of the substrate due to subsurface transport, and the second term f_{mf} is the contribution of the rough dielectric boundary. Compute the first term following the Ashikhmin and Shirley's model as

$$f_{\text{diff}}\ (\omega_i, \omega_o) = \frac{28 k_{\text{d}}}{23\pi}\left(1 - \left(\frac{\eta_{\text{ext}} - \eta_{\text{int}}}{\eta_{\text{ext}} + \eta_{\text{int}}}\right)^2\right)\left(1 - (1 - 0.5\cos\theta_i)^5\right)\left(1 - (1 - 0.5\cos\theta_o)^5\right), \tag{8}$$

with k_{d} the albedo of the surface and η_{ext} and η_{int} the incident and transmited indices of refraction. This models results from fitting a diffuse reflection taking into account the Fresnel term; you can identify that the three rightmost terms are very similar to the Schlick's approximation of the Fresnel equation. The second term in Equation 7 implements the reflection at the interface of the dielectric surface, following the same microfacet model as in the rough conductor:

$$f_{\text{mf}}(\omega_i, \omega_o) = \frac{D(\omega_h)\, F\left((\omega_h \cdot \omega_i), \eta_{\text{ext}}, \eta_{\text{int}}\right) G(\omega_i, \omega_o, \omega_h)}{4\cos\theta_i \cos\theta_o}. \tag{9}$$

The main difference is the Fresnel reflection coefficient F, which is now dependent on the indices of refraction (i.e. not using the Schlick's approximation); you may use the appropriate `Reflectance::fresnel()` function from `reflectance.h` to compute this term. Note that in this case the Fresnel term is assumed to be monochromatic (a scalar instead of a tristimulus); this is because the change on index of refraction of dielectrics is relatively small for dielectrics in the visible spectrum, and we can assume no wavelength dependence.

Then, implement the sampling routines `RoughSubstrate::sample` and `RoughSubstrate::pdf` that you will use in the following part of the assignment. `Use Warp::squareToBeckmann` and its associated pdf in `Warp::squareToBeckmannPdf` from the previous assignment for sampling the boundary, and use cosine-weighted hemispherical sampling for the diffuse part. To choose between the diffuse and microfacet terms, you can use Russian roulette based on the Fresnel term $F\left(\left(\mathbf{n}\cdot\omega_i\right),\eta_{\text{ext}},\eta_{\text{int}}\right)$, so that the probability of choosing the microfacet-based lobe is $p\left(f_{\text{mf}}\right)=F\left(\left(\mathbf{n}\cdot\omega_i\right),\eta_{\text{ext}},\eta_{\text{int}}\right)$, while the diffuse lobe has $p\left(f_{\text{diff}}\right)=1-p\left(f_{\text{mf}}\right)$. Note that this blending is done by computing the Fresnel term over the surface's normal, and not the microfacet normal. Once again, remember that similar to the emitters, the materials' sampling routiness return the **weight** of the sample $w\left(\omega_i\right)=f_r\left(\omega_i,\omega_o\right)/p\left(\omega_i\right)\cos\theta_i$.

You can test your implementation using the scene `./cbox/cbox_substrate_ems.xml`; the reference, computed for 1024 samples per pixel, can be found in the folder `./references/`.

2 BRDF Sampling (30%)

Once you implemented your material models, lets try them in a new scene: Render both `./serapis/serapis_rough_ems.xml` and `./serapis/serapis_smooth_ems.xml`, that contain a rough and smooth conductor illuminated by an environment map, respectively. As the roughness decreases, the amount of variance significantly increases. While the `DirectEmitterSampling` integrator worked just fine with diffuse materials, it struggles rendering materials with directional behaviour.

In the second part of this assignment, you will implement a direct lighting integrator based purely on BRDF sampling. Rather than generating points on emitters, you will sample the BRDF at the intersection point and trace a ray to check if it intersects a light source.

Create a `direct_mats.cpp` file, and create a new integrator called DirectMaterialSampling. Use the `NORI_REGISTER_CLASS` to register the new integrator as `"direct_mats"` as:

```
NORI_REGISTER_CLASS(DirectMaterialSampling, "direct_mats");
```

This integrator should use `BSDF::sample` to generate a new ray direction, and trace a ray to find the next intersection. You can find the BSDF at the intersection by using `Mesh::getBSDF()`. If the next intersection hits a light source, it should use the emitter interface you've designed to evaluate the contribution from the light source. Note that your interface for emitters might start to look very similar to the BSDF interface.

The estimate computed by this integrator corresponds to

$$L_o\left(\mathbf{x},\omega_o\right)\approx L_e\left(\mathbf{x},\omega_o\right)+\frac{1}{N}\sum_{k=1}^{N}\frac{L_e\left(\mathbf{r}\left(\mathbf{x},\omega_i^{(k)}\right),-\omega_i^{(k)}\right)f_r\left(\mathbf{x},\omega_o,\omega_i^{(k)}\right)\cos\theta_i^{(k)}}{p_\Omega\left(\omega_i^{(k)}\right)},\tag{10}$$

where $\omega_i^{(k)}$ was generated using BRDF sampling with pdf $p_\Omega\left(\omega_i^{(k)}\right)$. Note that here the pdf is defined over solid angle, and therefore there is no need for transforming measurements.

You can test your implementation using the scene `./serapis/serapis_smooth_mats.xml` and `./serapis/serapis_smooth_mats.xml`. For a reference solution, render the cbox scenes at `./cbox/cbox_conductors_mats.xml` and `./cbox/cbox_substrate_mats.xml` with a high number of samples; they should converge to the reference solution for the emitter sampling integrator in `./references/cbox_conductors_ems.exr` and `./references/cbox_substrate_ems.exr`.

3 Optional: Multiple Importance Sampling (30%)

As you have seen in the previous points, the implemented integrators (and therefore sampling routines) excel at different types of light and reflectances. Diffuse materials and small lights perform better with `DirectEmitterSampling`, and larger lights and directional materials are better sampled with `DirectMaterialSampling`. You can validate that behaviour rendering `./veach_mi/veach_ems.xml` and `./veach_mi/veach_mats.xml`, where from the roughness of the reflectors increases from top to bottom, and the size of the light source increases from left to right. As the final integrator in this assignment, you will implement multiple importance sampling (MIS). This integrator combines both emitter and BRDF sampling into a robust integrator.

Create a `direct_mis.cpp` file, and inside it create a new integrator called `DirectMIS`. Use the `NORI_REGISTER_CLASS` to register the new integrator as `"direct_mis"` as:

```
NORI_REGISTER_CLASS(DirectMIS, "direct_mis");
```

This integrator should take one sample using emitter sampling, and a second sample using BRDF sampling. Add the contribution of both samples, but weight them each by their MIS weights as in the lecture.

The estimate computed by this integrator can be expressed as

$$
\begin{aligned}
L_o\left(\mathbf{x}, \omega_o\right) &\approx L_e\left(\mathbf{x}, \omega_o\right) + w_{\text{em}} \frac{\mathscr{F}\left(\mathbf{x}, \omega_o, \vec{\omega}_{\text{em}}\right)}{p_{\text{em}}\left(\vec{\omega}_{\text{em}}\right)} + w_{\text{mat}} \frac{\mathscr{F}\left(\mathbf{x}, \omega_o, \vec{\omega}_{\text{mat}}\right)}{p_{\text{mat}}\left(\vec{\omega}_{\text{mat}}\right)}, \\
w_{\text{em}} &= \frac{p_{\text{em}}\left(\vec{\omega}_{\text{em}}\right)}{p_{\text{em}}\left(\vec{\omega}_{\text{em}}\right) + p_{\text{mat}}\left(\vec{\omega}_{\text{em}}\right)}, \\
w_{\text{mat}} &= \frac{p_{\text{mat}}\left(\vec{\omega}_{\text{mat}}\right)}{p_{\text{em}}\left(\vec{\omega}_{\text{mat}}\right) + p_{\text{mat}}\left(\vec{\omega}_{\text{mat}}\right)},
\end{aligned}
\tag{11}
$$

where the subindices $(.)_{\text{mat}}$ and $(.)_{\text{em}}$ account for material- and emitter-based sampling routines, and $\mathscr{F}\left(\mathbf{x}, \omega_o, \vec{\omega}_k\right) = L_i\left(\mathbf{x}, \omega_k\right) f_r\left(x, \omega_o, \omega_k\right) \cos\theta_k$. To accomplish this, you will need

to extend your implementation in a number of ways. Firstly, you will need a way to evaluate the PDF of some strategy A (e.g. emitter sampling), using a sample generated from a different strategy B (e.g. BSDF sampling). Also, when computing MIS weights, keep in mind that you will need to convert both PDFs to the same measure before adding them (you should not add or compare a PDF over solid angle to one defined over surface area, and vice versa).

For texting, use `./veach_mi/veach_mis.xml`, and compare the three integrators in the same scene. All three should converge to the same solution.

4 References

You may find the following general references useful:

- "Physically Based Rendering, Fourth Edition: From Theory To Implementation" by Matt Pharr, Wenzel Jakob, and Greg Humphreys. The MIT Press, 4th edition, March 2023. http://www.pbr-book.org/
- "Global Illumination Compendium - The Concise Guide to Global Illumination Algorithms", Philip Dutre, 2003. https://people.cs.kuleuven.be/philip.dutre/GI/
- "Robust Monte Carlo Methods for Light Transport Simulation", PhD Thesis by Eric Veach, Stanford University, December 1997. https://graphics.stanford.edu/papers/veach_thesis/thesis-bw.pdf
- Online Nori docs: https://www.cs.cornell.edu/courses/cs6630/2012sp/nori/api/annotated.html/

Feel free to consult additional references when completing projects, but remember to cite them in your writeup. When asked to implement feature X, we request that you don't go and read the source code of the implementation of X in some other renderer, because you will likely not learn much in the process. The PBRT book is excluded from this rule. If in doubt, get in touch with the course staff.

5 What to submit?

You should submit all your rendered figures and the source code you generated in this assignment. The submission should be a a `zip` file with name `p3_NIP1_NIP2.zip`, and with the following structure:

- A file `README.txt` including the name of the authors (as sanity check) and all references consulted during the development of the assignment.

- A folder `./figures` including all generated figures (both `exr` and `png`); the figures should be named with the name of the `xml` scene, appended with the number of samples used to generate these figures.
- A folder all source files you modified or added in your implementation.

In case your file gets too big for the submission system, you can just include a link to a shared folder (Google Drive) with additional figures in `README.txt`.

The **deadline** for this task will be just before the first session of Assignment #4 corresponding to the lab group you are enrolled on:

- **Group 1 (Tuesdays B): November 18, 2024.**
- **Group 2 (Thursdays A): November 13, 2024.**