Modeling and Simulation of Appearance

Lab #3 - Monte Carlo Based Materials

Julio Marco - juliom@unizar.es
Néstor Monzón - nmonzon@unizar.es

2024 - 2025

Disclaimer

- This lab **builds upon** what we did in **the previous ones**.

- You need to install a patch (`patch_03.zip`) from Moodle, and run **CMake** again.

    - **You will need to add include/nori/reflectance.h and src/reflectance.cpp in the CMakeLists.txt file**

About this lab

- We are implementing more complex **appearance** models.

    - *Microfacet-based materials.*

- We are implementing some Monte Carlo integrators based on **sampling the material properties** (BRDF sampling)

About this lab

- We are implementing more complex **appearance** models.
    - *Microfacet-based materials.*

- We are implementing some Monte Carlo integrators based on **sampling the material properties** (BRDF sampling)

- We are finally using **Multiple Importance Sampling (MIS)** to combine:
    - BRDF sampling.
    - Emitter sampling.

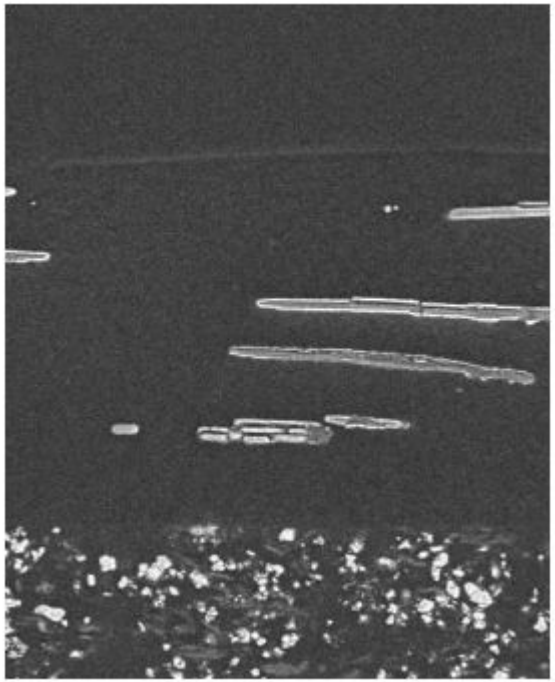
Microfacet-based BRDF (40%)

- You are implementing **two** types of BRDFs:
    - Rough conductor.
    - Rough substrate.

Microfacet-based BRDF (40%)

- You are implementing **two** types of BRDFs:
    - Rough conductor.
    - Rough substrate.

The surface is assumed to be formed of tiny mirrors.



Guillen et al. 2020

Microfacet-based BRDF (40%)

- You are implementing **two** types of BRDFs:
    - Rough conductor.
    - Rough substrate.

You must work on `microfacet.cpp`, and implement, for both of them:

- `RoughXXXX::eval()`
- `RoughXXXX::pdf()`
- `RoughXXXX::sample()`

Microfacet-based BRDF (40%)

You must work on `microfacet.cpp`, and implement, for both of them:

- `RoughXXXX::eval()`
- `RoughXXXX::pdf()`
- `RoughXXXX::sample()`

You have many functionalities available at `Reflectance.h`:

- `Reflectance::fresnel()` → $F(\cdot)$
- `Reflectance::G1()` → $G1(\cdot)$

You may need to reuse `Warp::squareToBeckman`.

Use available resources wisely!

$$f_r(\omega_i, \omega_o) = f_{\text{diff}}(\omega_i, \omega_o) + f_{\text{mf}}(\omega_i, \omega_o), \tag{9}$$

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

$$f_{\text{diff}}(\omega_i, \omega_o) = \frac{28k_{\text{d}}}{23\pi}\left(1 - \left(\frac{\eta_{\text{ext}} - \eta_{\text{int}}}{\eta_{\text{ext}} + \eta_{\text{int}}}\right)^2\right)\left(1 - (1 - 0.5\cos\theta_i)^5\right)\left(1 - (1 - 0.5\cos\theta_o)^5\right), \tag{10}$$

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 10 implements the reflection at the interface of the dielectric surface, following the same microfacet model as in the rough conductor:

$$f_{\text{mf}}(\omega_i, \omega_o) = \frac{D(\omega_h)\, F\left((\omega_h \cdot \omega_i), \eta_{\text{ext}}, \eta_{\text{int}}\right)\, G(\omega_i, \omega_o, \omega_h)}{4\cos\theta_i \cos\theta_o}. \tag{11}$$

Our report

PBRT book →

$$F_{\text{r}}(\cos\theta) = R + (1 - R)(1 - \cos\theta)^5,$$

where R is the reflectance of the surface at normal incidence.

Given this Fresnel term, the diffuse term in the following equation successfully models Fresnel-based reduced diffuse reflection in a physically plausible manner:

$$f_{\text{r}}(\text{p}, \omega_{\text{i}}, \omega_{\text{o}}) = \frac{28R_{\text{d}}}{23\pi}(1 - R_{\text{s}})\left(1 - (1 - \frac{(\mathbf{n}\cdot\omega_{\text{i}})}{2})^5\right)\left(1 - (1 - \frac{(\mathbf{n}\cdot\omega_{\text{o}})}{2})^5\right).$$

We will not include the derivation of this result here.

<<FresnelBlend Public Methods>>=

```
Spectrum SchlickFresnel(Float cosTheta) const {
    auto pow5 = [](Float v) { return (v * v) * (v * v) * v; };
    return Rs + pow5(1 - cosTheta) * (Spectrum(1.) - Rs);
}
```

<<BxDF Method Definitions>>+=

```
Spectrum FresnelBlend::f(const Vector3f &wo, const Vector3f &wi) const {
    auto pow5 = [](Float v) { return (v * v) * (v * v) * v; };
    Spectrum diffuse = (28.f/(23.f*Pi)) * Rd *
        (Spectrum(1.f) - Rs) *
        (1 - pow5(1 - .5f * AbsCosTheta(wi))) *
        (1 - pow5(1 - .5f * AbsCosTheta(wo)));
    Vector3f wh = wi + wo;
    if (wh.x == 0 && wh.y == 0 && wh.z == 0) return Spectrum(0);
    wh = Normalize(wh);
    Spectrum specular = distribution->D(wh) /
        (4 * AbsDot(wi, wh) *
         std::max(AbsCosTheta(wi), AbsCosTheta(wo))) *
         SchlickFresnel(Dot(wi, wh));
    return diffuse + specular;
}
```

BRDF Sampling (30%)

- Render `serapis/serapis_[rough | smooth]_ems.xml`.

As roughness decreases, variance increases

BRDF Sampling (30%)

- Render `serapis/serapis_[rough | smooth]_ems.xml`.

- Direct emitter sampling struggles when rendering materials with directional behavior.

BRDF Sampling (30%)

- Render `serapis/serapis_[rough | smooth]_ems.xml`.
- Direct emitter sampling struggles when rendering materials with directional behavior.
- We are now **sampling the material BRDF** rather than the light.
    - Trace a ray from the material and check whether it hits a light source.

BRDF Sampling (30%)

- We are now **sampling the material BRDF** rather than the light.
    - Trace a ray from the material and check whether it hits a light source.

- Work on `direct_mats.cpp`.

Multiple Importance Sampling (30%)

- Direct **emitter** sampling works better for diffuse materials or small lights.

- Direct **material** sampling works better for larger lights and directional materials.

- **Multiple Importance Sampling** allows to **combine** both of them.

- Work on `direct_mis.cpp`.

Submission

Include:

- `README.txt` with the names of the authors and consulted references.
- A folder `./figures` with the generated stuff.
- A folder `./src` with all the source files you modified or added.