Nikolai Hofmann, Marc Stamminger

Erlangen, 22.05.2025

Global Illumination Tutorials (exercise sheet 2)

In this assignment we will extend the shading from the previous assignment with more elaborate materials. To this end, we will implement different *bidirectional reflection distribution functions* (BRDFs). We will also look into a brute-force method to integrate the incoming light over the hemisphere and finally improve convergence by applying *importance sampling* strategies.

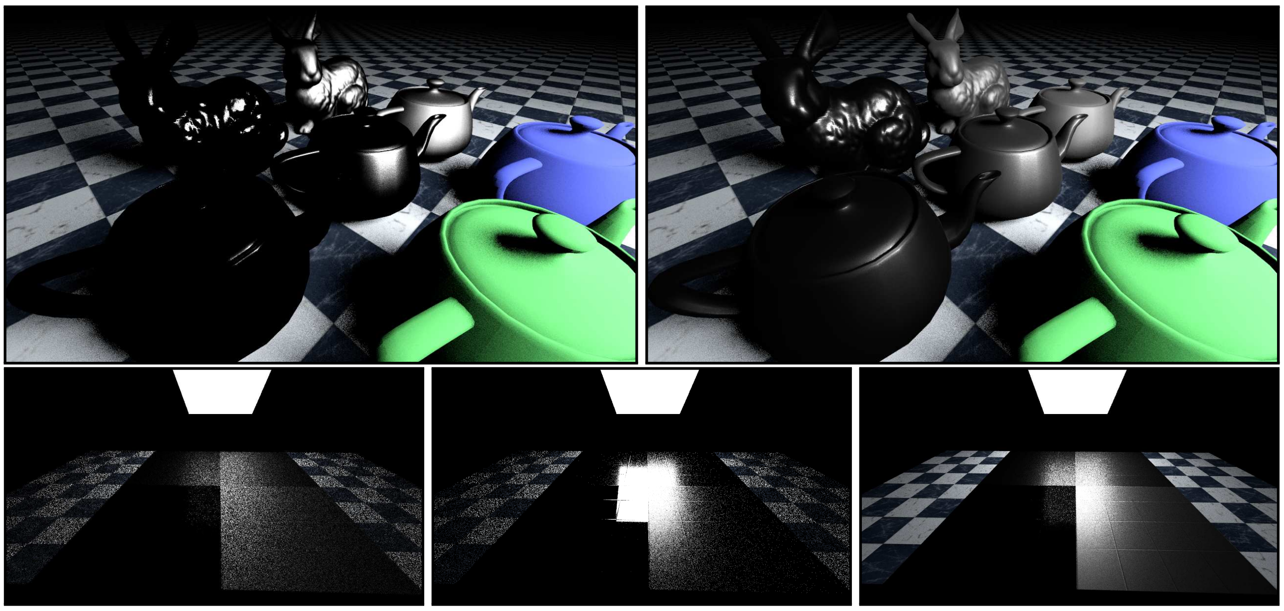

Figure 1: Top row: Direct illumination from an area light source using the Blinn-Phong model (left) and the GGX microfacet model (right) for specular reflection. Bottom row: Monte Carlo integration of the incoming light over the hemisphere using uniform sampling (left), importance sampling according to the BRDF (middle), or light source sampling (right). The textured tiles employ a diffuse BRDF, while the center patches employ a microfacet BRDF with varying roughness.

Submission. To submit, register your group of two on StudOn and submit **one** `.zip` file for your group, containing the files you've worked on, and potentially a `.pdf` or rendered images. In order to be graded, your group needs to upload your solution before the submission deadline **and** check in with us during the grading period.

Lecture: on Tuesdays from 12:15 - 13:45
Tutorials: on Thursdays from 09:00-12:00
Submission deadline: Wednesday 11.06.2025, 23:55
Grading period: Thursday 12.06.2025, 09:00-12:00

First, we will implement a diffuse, phong and microfacet reflection BRDF. Mircofacet reflection models, combined with a proper description of light sources, are often referred to as *physically based rendering* (PBR). Since we are still limited to direct illumination, these techniques are also applicable in a real-time rendering context, such as games for example. The material system consists of: A `Material` class (`src/gi/material.h`), which provides all required data to render a surface, such as the BRDF, parameters, and textures. The `BRDF` class (`src/gi/brdf.h`) provides evaluation and sampling routines for the respective reflection model of the material. Note that there are helper functions available in `src/gi/sampling.h` and by convention, directions always point *away* from the shading point with BRDFs.

Assignment 1 [2 Points] (Lambertian and Phong BRDF)
[Files: `src/gi/brdf.cpp, src/gi/fresnel.h, src/algorithms/direct.cpp`]

We will first look at the BRDF evaluation routine, which is simply called `BRDF::f`. Implement a diffuse Lambertian and a specular Phong BRDF (or Blinn-Phong) in `src/gi/brdf.cpp` according to the lecture. Take care to correctly normalize each BRDF. Since specular highlights are heavily dependent on the incident view angle, use a Fresnel term[1] to scale the contribution of the specular Phong term. In this case, Schlick's approximation is sufficient and you are asked to implement this term in `src/gi/fresnel.h`. In order to actually see your implemented BRDFs in action, adapt the renderer `src/algorithms/direct.cpp` to evaluate the BRDF during shading via the `SurfaceHit` interface. Render the provided config file `configs/a02_phong.json` to validate your implementation against Figure 1 (top, left). Note that the reference uses the Blinn-Phong model.

Assignment 2 [3 Points] (GGX microfacet BRDF)
[Files: `src/gi/brdf.cpp`]

Whereas the Phong BRDF is a rather simple approximation of the specular reflection lobe, microfacet models manage to more closely resemble physical material properties. We now ask you to implement a microfacet BRDF based on the GGX distribution. All necessary (and more) information is available in the paper "Microfacet models for refraction through rough surfaces" [2].
The basic form of the microfacet model is the following (see Equations 20 and 23 in the paper):

$$f(w_o, w_i) = \frac{FDG}{4(w_i \cdot n)(w_o \cdot n)}\ , \tag{1}$$

where w_o, w_i, n are the outgoing, incoming and surface normal vectors, respectively. F is the Fresnel term, D is a probability density of how the microfacets are angled, and G is a self-shadowing term between microfacets. You may again rely on Schlick's approximation for the Fresnel term, but use the GGX distribution for both the D and G terms (see Equations 33 and 34 in the paper).
You can either compute all terms in the `MicrofacetReflection::f` function directly, or spread your implementation to the helper functions `GGX_D` and `GGX_G1`. For reference, render the provided config file `configs/a02_ggx.json` and compare your result to Figure 1 (top, right). You may also render `configs/a02_spz_ggx.json`, if you want to look at something else for a change.
By the way, these reflection models also do not match *the real world*™ perfectly. They are merely a more sophisticated model than the Phong model, primarily designed to match empirical measurements more closely, while still remaining easily controllable (via roughness) and invertable.

[1] `https://en.wikipedia.org/wiki/Fresnel_equations`
[2] `https://www.cs.cornell.edu/~srm/publications/EGSR07-btdf.pdf`

Assignment 3 [2 Points] (Monte-Carlo integration of incoming light)
[Files: `src/gi/sampling.h, src/algorithms/brdf_imp.cpp`]

A simple (brute-force) method to numerically integrate incoming light over the hemisphere is to choose uniform random directions over the hemisphere, shoot a ray and accumulate radiance where a light source was hit. To this end, we first need a mapping of random samples to (tangent-space) directions over the hemisphere and their respective PDF in terms of solid angle, which should be implemented in the following two functions in `src/gi/sampling.h`: `uniform_sample_hemisphere()` and `uniform_hemisphere_pdf()`.
Now, to compute the radiance $L(\omega_o)$ reflected back towards the camera in direction ω_o, we numerically solve the following integral using Monte Carlo integration:

$$L(\omega_o) = \int_{\Omega} L_e(\omega_i) f(w_o, w_i) \cos(w_i)\, dw_i = \frac{1}{N} \sum_{N} \frac{L_e(w_i) f(w_o, w_i) \cos(w_i)}{pdf(w_o, w_i)}\,, \tag{2}$$

where $f(w_o, w_i)$ is the BRDF, $L_e(w_i)$ the emission from the light source which was hit from direction w_i and $pdf(w_o, w_i)$ the PDF of the sampled direction w_i. Thus, in `src/algorithms/brdf_imp.cpp`, generate a (tangent-space) direction on the hemisphere, transform it into world-space, setup a secondary ray in the sampled direction and intersect the ray with the scene. If you happen to hit a light source, compute the outgoing radiance via the above equation. Render with the provided config file `configs/a02_imp.json` and compare your result to Figure 1 (bottom, left) for reference. Note that in Figure 1 we chose $N = 1$ for comparison with light source integration, but you are free to choose other values for N as well.

Assignment 4 [3 Points] (BRDF importance sampling)
[Files: `src/gi/sampling.h, src/gi/brdf.cpp, src/algorithms/brdf_imp.cpp`]

You might notice that the results exhibit a large amount of noise, especially noticeable with specular highlights. To improve upon this, we need to more closely match our sampling scheme to the BRDF, i.e. shoot more rays into directions where the value of the BRDF is large. To this end, we now let the BRDF pick a sampling direction instead, via the `BRDF::sample` functions for both the Lambertian and Microfacet BRDFs. This function is expected to return the evaluated BRDF (`brdf`), the sampled direction (`w_i`, in world-space), and the respective PDF (`pdf`).
As the Lambertian BRDF is constant, we sample the cosine term in the rendering equation instead. Thus, implement `cosine_sample_hemisphere` and `cosine_hemisphere_pdf` analogous to the uniform case, and use your implementation to complete `LambertianReflection::sample`.
In order to sample the GGX BRDF, we rely on the sampling strategy as proposed in the paper. First, implement drawing a (tangent space) micro normal from a random sample using Equations 35 and 36. Second, compute the corresponding PDF using Equations 14, 24 and 38. Complete the function `MicrofacetReflection::sample`, similar to the `LambertianReflection::sample`. Hint: to get the outgoing direction, reflect around the sampled (world space) micro normal.
Finally change the sampling in the `BRDFImportance` renderer to use `SurfaceInteraction::sample` instead of uniform sampling (switchable via `UNIFORM`), render and compare the results with Figure 1 (bottom, middle) for reference. Note the differences between rendering with uniform sampling, importance sampling and light source integration, as is shown in Figure 1 (bottom row). During grading, you should be able to explain the reason behind these differences, which technique performs best for which type of BRDF and why.

Happy Hacking! :)