ECSE 446/546: Realistic/Advanced Image Synthesis

Assignment 3: Direct Illumination and Importance Sampling

Due: Wednesday, November 14th, 2023 at 11:59pm EST on myCourses

Final weight: **25%**

Contents

1 Assignment Policies and Submission Process

Download and modify the standalone `Python` script we provide on *myCourses*, renaming the file according to your student ID as `YourStudentID.py`

> As usual, every new file you submit on *myCourses* will override the previous submission, and we will only grade the **final submitted file**.

1.1 Late Policy, Collaboration & Plagiarism, Python/Library Usage Rules

For late policy, collaboration & plagiarism, Python language and library usage rules, please refer to the Assignment 0 handout.

2 Assignment Overview and Notes

Building atop the previous assignment, you will implement several Monte Carlo (MC) estimators for a more general direct illumination setting: one with spherical area lights, and diffuse and glossy Phong BRDFs.

This assignment, including the base code and your submitted solution code, will differ in a few subtle — but important — ways, compared to Assignments 1 and 2:

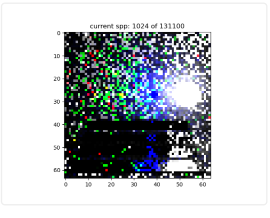
- to simplify your MC integration code, and the progressive rendering code, we will *no longer* distinguish between sample counts *per pass* and total desired sample counts; instead, each render pass (implemented in `Scene.render`) will compute a **1-sample MC estimate**, and the updated progressive rendering-and-display loop (in `Scene.progressive_render_display`) will accumulate, average and display these 1-sample estimates, up to the total desired per-pixel sample count (`total_spp`). Note these functions' updated parameter lists;
- instead of explicitly defining lights in the `Scene`, we will now treat emitters as first-class geometric primitives: scene objects (derived from the `Geometry` class) now utilize the previously unused Le member variable that specifies the object's (spatially- and directionally-uniform) emitted radiance: i.e., a non-zero RGB vector for lights, and zero — the default constructor value — for non-emitting objects. The `Scene.add_geometry` routine was augmented to additionally call `Scene.add_light`, implicitly building the list of light sources (as opposed to explicitly, as in, e.g., Assignment 2) and populating the `Scene.lights` list member variable — having easy access to the subset of scene objects that are lights will be useful, e.g., when looping over lights in your rendering code; and,
- other important changes to the base code will be discussed later in the handout, as needed.

Be mindful of the points above when copying/adapting code from your previous assignments into the A3 base code.

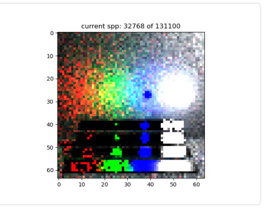
At a high-level, ECSE 446 students will implement *two* (2) MC estimators — one with spherical light importance sampling, and another with diffuse and Phong BRDF importance sampling; ECSE 546 students will additionally implement a multiple importance sampling (MIS) estimator that combines these two sampling techniques.

That being said, we strongly advocate for you to begin by implementing a basic, uniform spherical PDF MC estimator for the more general direct illumination integral we will be exploring. We have left the `UNIFORM_SAMPLING` enumerate available exactly for this reason — we will *not grade* the uniform sampler, and so its implementation is completely optional.

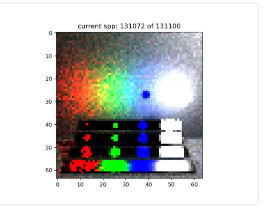
> If you choose to implement the uniform sampler first, you can deploy a low resolution, high `spp` render, while you work on your more efficient MC estimators. The three examples below were rendered at 64×64 using our uniform sampler.



Uniform Sampling (1024 spp)



Uniform Sampling (32768 spp)



Uniform Sampling (131072 spp)

You may also wish to adapt your `Scene.progressive_render_display` routine to output image files more frequently (e.g., on a log scale, such as every power-of-2 accumulated samples) — this way, if you render crashes, or you think it's run long enough (before it reaches `total_spp` samples), you will still have outputed renderings you can refer to.

> Feel free to adapt `Scene.progressive_render_display` as much as you like; we will not grade this function as part of the Assignment 3 evaluation.

Please read the rest of the handout before beginning any of your implementation; as noted above, other notable changes to the base code will be discussed below, and will be useful to be aware of prior to coding.

3 General Direct Illumination

In Assignment 2, you implemented a multi-sample MC estimator of the ambient occlusion equation, a simplified instance of direct illumination, where the scene comprises only diffuse BRDFs and a single, distant and uniform environment light.

This assignment instead treats the generic direct illumination equation:

$$L_o(\mathbf{x}, \omega_o) = \int_{S^2} L_e(\mathbf{x}, \omega_i)\, V(\mathbf{x}, \omega_i)\, f_r(\mathbf{x}, \omega_o, \omega_i)\, \max(\mathbf{n} \cdot \omega_i, 0)\, d\omega_i \,.$$

As we will be treating *both* diffuse and Phong BRDFs in this assignment, we have adapted the base code in another subtle — but essential — way: the `brdf_params` property of the `Scene` geometries now uses the previously dormant 4th element. When `brdf_params[3]` is 1, then the object's BRDF is diffuse and the first three parameters (`brdf_params[0:3]`) correspond to the *diffuse reflectance* ρ_d; otherwise, the object's BRDF is Phong, `brdf_params[3]` corresponds to the Phong exponent α, and the first three parameters (`brdf_params[0:3]`) correspond instead to the *specular reflectance* ρ_s.

As such, the BRDF in our new direct illumination equation is

$$f_r(\mathbf{x}, \omega_o, \omega_i) = \begin{cases} \rho_d / \pi, & \text{if } \alpha = 1\,, \\ \left(\rho_s(\alpha+1)/(2\pi)\right) \max(0, (\omega_r \cdot \omega_i)^\alpha), & \text{if } \alpha > 1\,, \end{cases}$$

where the reflected view-direction $\omega_r = 2(\mathbf{n} \cdot \omega_o)\, \mathbf{n} - \omega_o$, and ω_o is oriented *from* the shading point *to* the viewer (i.e., the opposite direction of your eye rays).

Note here that, when shading with the Phong BRDF, there are **two** cosine terms in your direct illumination equation: the foreshortening term about the normal, and the Phong cosine (exponent) reflection lobe; in the diffuse setting, the latter — view-dependent — cosine is no longer present. Appropriately taking this into account will be necessary for *all* your estimators, with additional attention required when implementing the second assignment deliverable (BRDF Importance Sampling).

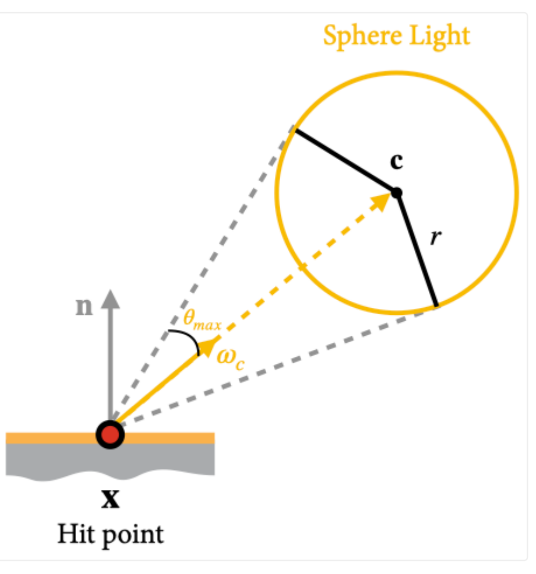
4 Light Importance Sampling

We will be working with a variant of Veach's MIS scene, which includes four (4) spherical light sources of increasing size (and one additional off-screen fill light), four metallic plates of increasing specularity, and a diffuse floor and back wall.

> You can disable various scene elements (e.g., lights, geometry) — by commenting out the appropriate lines in the scene definition code — as well as reducing the render resolution, when isolating and debugging the behaviour of your estimators.

The first deliverable will require that you implement a light importance sampling strategy for these spherical emitters. We discussed three such approaches in class, and you will implement the last one: spherical solid angle sampling. As such, it will remain most natural to discuss your estimator — and the sampling PDF — in terms of the solid angle form of the direct illumination equation.

The geometric scenario we will be treating when sampling directions towards the subtended solid angle of a spherical light is summarized in the diagram, below:



Sampling directions toward a spherical light source

Given a shading point $\mathbf{x}$ and a sphere light in the scene (with center $\mathbf{c}$ and radius r), we wish to sample directions uniformly within the solid angle subtened by the light, Ω_e. This solid angle can be parameterized by its central axis ω_c and its half-angle θ_{max}, due to the spherical symmetry of the light, and is

$$\Omega_e = 2\pi(1 - \cos(\theta_{max}))\,.$$

Given these two parameters (which you will have to compute at every shading point, and for every light[1]), you can draw directions ω_j proportional to $p_{light}(\omega) = 1/\Omega_e$ in two steps:

1. sample uniform directions $\bar{\omega}$ about the z-axis (i.e., for $\omega_c = z$) within the cone defined by $\bar{\omega} \in (\theta = [0, \theta_{max}], \phi = [0, 2\pi])$ and so with density $1/\Omega_e$, and
2. rotate these $\bar{\omega}$ directions into a coordinate frame aligned about the *actual* central axis towards the light ω_c, obtaining the final sampling directions ω_j.

[1] Be mindful of floating point precision issues when computing inputs and outputs of (inverse) trigonometric functions; you may need to appropriately bracket your computations, e.g., of θ_{max}.

For step 1 above, we can sample the canonical $\bar{\omega} = (\bar{\omega}_x, \bar{\omega}_y, \bar{\omega}_z)$ using two canonical uniform random numbers ξ_1, ξ_2 — computed using `np.random.rand` — as follows:

$$\bar{\omega}_z = 1 - \xi_1(1 - \cos(\theta_{max})) \qquad r = \sqrt{1 - \bar{\omega}_z^2} \qquad \phi = 2\pi\xi_2$$

$$\bar{\omega}_x = r\cos\phi \qquad \bar{\omega}_y = r\sin\phi\,.$$

Strategies for performing step 2 above were discussed in lecture. Feel free to implement either/both of these stages as standalone routines, if you wish to (optionally) compartmentalize your code.

You can now proceed with evaluating your light importance sampling MC estimator by tracing these sampled shadow rays — from your hit point towards $\omega_j \sim p_{light}(\omega)$ — as:

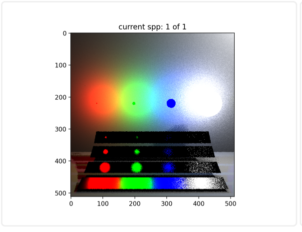
$$L_o(\mathbf{x}, \omega_o) \approx \frac{1}{N} \sum_{j=1}^{N} \frac{L_e(\mathbf{x}, \omega_j)\, V(\mathbf{x}, \omega_j)\, f_r(\mathbf{x}, \omega_o, \omega_j)\, \max(\mathbf{n} \cdot \omega_j, 0)}{p_{light}(\omega_j)}\,.$$

Note that, the only major differences with this light importance-sampled MC estimator and a simpler uniform spherical MC estimator are the (uniform) values of $p(\omega_j)$ and the sampling routine for drawing $\omega_j \sim p_{light}(\omega)$. Thus, a uniform spherical sampler can be readily modified to perform this light importance sampling, i.e., without changing much/any of the implementation of the MC estimator's numerator.

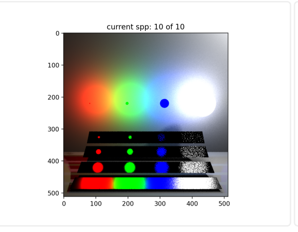
Keep in mind, now that emitters are first-class scene geometries, light sources can both emit *and occlude* (but not reflect) light.

> Your estimator does not have to vectorize over lights — you will not be penalized for using a `for` loop over emitters; vectorizing over Rays remains the highest-throughput (and only **required**) performance gain in our rendering system.

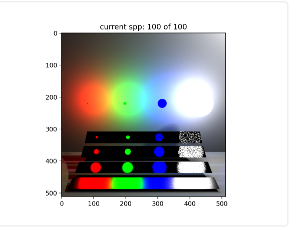
> **Deliverable 1 [15 points]**
>
> - Use branching logic that treats the `sampling_type == LIGHT_SAMPLING` case in `Scene.render` and implement the light importance sampling scheme described above.
> - Note that you should evaluate the MC estimator above *for every light in the scene*, accumulating their (1-sample) estimates into your final ("1-sample") render pass MC estimate.
> - Your estimator code should gracefully treat shadow ray acne and support both BRDF types.



Light Sampling (1 spp)



Light Sampling (10 spp)



Light Sampling (100 spp)

5 BRDF Importance Sampling

The next MC estimator you will implement will employ BRDF importance sampling. Here, the sampling PDF you will implement will depend on whether you are on a diffuse or glossy Phong shading point, as:

$$p_{brdf}(\omega) = \begin{cases} (1/\pi) \max(0, (\mathbf{n} \cdot \omega)), & \text{if } \alpha = 1\,, \\ \left((\alpha+1)/(2\pi)\right) \max(0, (\omega_r \cdot \omega)^\alpha), & \text{if } \alpha > 1\,. \end{cases}$$

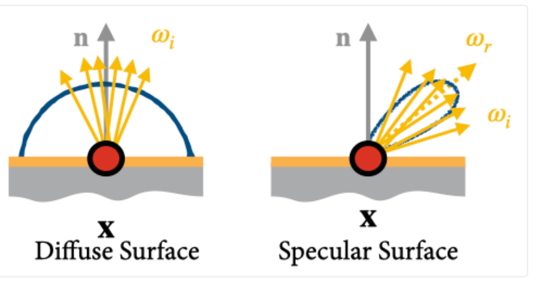
Much like with light importance sampling, you can draw samples $\omega_j \sim p_{brdf}(\omega)$ in two stages, first drawing them in a canonical orientation aligned with the z-axis, before rotating them into an appropriate coordinate system at the shade point.

For diffuse surfaces, you will be drawing samples according to a (normalized) cosine lobe aligned about the shading normal $\mathbf{n}$ and, for glossy Phong surfaces, a (normalized) cosine-power lobe aligned about the reflected outgoing viewing directions ω_r.

Conveniently, the sampling routine for the canonical orientation is parameterized by α and can generate both cosine and cosine-power distributions, as:

$$\bar{\omega}_z = \xi_1^{(1/(\alpha+1))} \qquad r = \sqrt{1 - \bar{\omega}_z^2} \qquad \phi = 2\pi\xi_2$$

$$\bar{\omega}_x = r\cos\phi \qquad \bar{\omega}_y = r\sin\phi\,.$$



Sampled distributions (yellow arrows) and BRDFs (blue outline) for diffuse and glossy Phong surfaces.

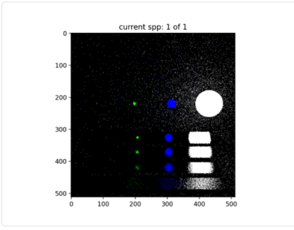
You can now similarly proceed with evaluating your BRDF importance sampling MC estimator by tracing these sampled shadow rays — from your hit point towards $\omega_j \sim p_{brdf}(\omega)$ — as:

$$L_o(\mathbf{x}, \omega_o) \approx \frac{1}{N} \sum_{j=1}^{N} \frac{L_e(\mathbf{x}, \omega_j)\, V(\mathbf{x}, \omega_j)\, f_r(\mathbf{x}, \omega_o, \omega_j)\, \max(\mathbf{n} \cdot \omega_j, 0)}{p_{brdf}(\omega_j)}\,,$$

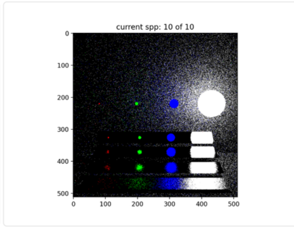
keeping in mind that, depending on the type of surface, the PDF will "cancel out" different terms, above: for diffuse surfaces, $f_r(\mathbf{x}, \omega_o, \omega_j)\, \max(\mathbf{n} \cdot \omega_j, 0) / p_{brdf}(\omega_j)$ will simplify to ρ_d and, for glossy Phong surfaces it will simplify to $\rho_s \max(\mathbf{n} \cdot \omega_j, 0)$. While you *can* implement these optimizing simplifications to reduce superfluous computation, simply substituting into the MC estimator expression above will yield the same result.

Unlike the light importance-sampled MC estimator, the conditioning on surface type *and* the non-uniform value of the PDF will lead to *slightly* more complicated code.

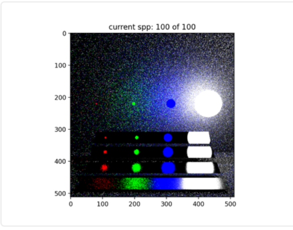
> **Deliverable 2 [15 points]**
>
> - Use additional branching logic in `Scene.render` that treats the `sampling_type == BRDF_SAMPLING` and implement the BRDF importance sampling scheme described above.
> - Note that you should evaluate the MC estimator above *for every light in the scene*, accumulating their (1-sample) estimates into your final ("1-sample") render pass MC estimate.
> - Your estimator code should gracefully treat shadow ray acne and support both BRDF types.



BRDF Sampling (1 spp)



BRDF Sampling (10 spp)



BRDF Sampling (100 spp)

ECSE 546 Students Only

6 Multiple Importance Sampling

ECSE 546 students will additionally implement a multiple importance sampling estimator that combines light and BRDF sampling distributions.

Recall that, given two sampling distributions p_f and p_g used to estimate an integral $F = \int f(x)\, dx$, we can express a general MIS MC estimator that uses both strategies, as

$$\frac{1}{n_f} \sum_{j=1}^{n_f} \frac{f(x_j)\, w_f(x_j)}{p_f(x_j)} + \frac{1}{n_g} \sum_{k=1}^{n_g} \frac{f(x_k)\, w_g(x_k)}{p_g(x_k)},$$

where n_f and n_g are the number of samples ($x_j \sim p_f$ and $x_k \sim p_g$) drawn from the p_f and p_g distributions, and w_f and w_g are special weighing functions chosen so that the expectation of the estimator is the desired integral F. One provably good choice of weighing functions follows the *balance heuristic*:

$$w_s(x) = \frac{n_s p_s(x)}{\sum_{i \in S} n_i p_i(x)},$$

where $s \in S = \{f, g\}$, in our two-strategy setting, above.

While this general MIS formulation is suitable, in the context of our 1-sample per-render-iteration progressive rendering setting, it poses a problem: with the smallest setting of $n_f = n_g = 1$, each iteration will generate two samples.

Since we absolutely wish to maintain the 1-sample-per-pass property[2] instead, we can exploit an important property of the balance heuristic: when using an equal number of samples per strategy, samples drawn in the MIS estimator with the balance heuristic weights are — in aggregate — proportional to **the average of all the strategies**.

In other words, in our two-strategy setting, if you draw samples ω_j according to the average of the light and BRDF PDFs, $\omega_j \sim p_{mis}(\omega) = \frac{p_{light}(\omega)}{2} + \frac{p_{brdf}(\omega)}{2}$ and use them in a *standard* MC estimator, as

$$L_o(\mathbf{x}, \omega_o) \approx \frac{1}{N} \sum_{j=1}^{N} \frac{L_e(\mathbf{x}, \omega_j)\, V(\mathbf{x}, \omega_j)\, f_r(\mathbf{x}, \omega_o, \omega_j)\, \max(\mathbf{n} \cdot \omega_j, 0)}{p_{mis}(\omega_j)}\,,$$

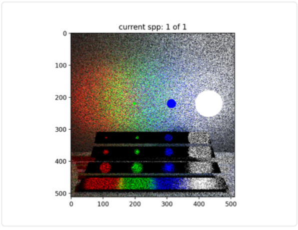
then your estimator will be statistically equivalent to the more general MIS-with-balance-heuristic estimator, above.

We discussed a similar two-strategy average PDF sampling scenario in lecture, and you will adapt and implement it to the light and BRDF sampling strategies in this assignment in order to realize (what ends up equating to) your MIS estimator. Note that, with this *average PDF* methodology, you are now able to stochastically draw a single sample $\omega_j \sim p_{mis}(\omega)$ per progressive render pass.

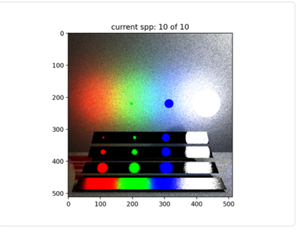
It is your responsibility to devise the PDF sampling and evaluation logic for this method.

[2] One good reason for maintaining this property consistently *across* all the approaches we implement, is that it will allow us to more easily perform *apples vs. apples* comparisons between light-only, BRDF-only, and MIS estimators.

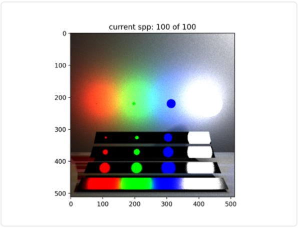
> **Deliverable 3 [10 points]**
>
> - Add one final branch in `Scene.render` to treat the `sampling_type == MIS_SAMPLING` scenario, and implement your MIS estimator using the aforementioned *average PDF* methodology.



MIS Sampling (1 spp)



MIS Sampling (10 spp)



MIS Sampling (100 spp)

7 You're Done!

Congratulations, you've completed the 4th (real) assignment. Review the submission procedures and guidelines at the start of the Assignment 0 handout before submitting the `Python` script file with your assignment solution.