ECSE 446/546: Realistic/Advanced Image Synthesis

Assignment 3: Advanced Direct Illumination

Due: Sunday, November 24th, 2024 at 11:59pm EST on myCourses

Final weight: **25%**

Contents

1 Assignment Policies

All future assignments, including this one, will build on top of Assignment 1.

1.1 Assignment Submission

Gather all the `python` *source files* within the **`taichi_tracer/`** folder (i.e., everything *except* the `scene_data_dir/` folder) and compress them into a single `zip` file. Name your `zip` file according to your student ID, as:

`YourStudentID.zip`

For example, if your ID is **234567890**, your submission filename should be **234567890.zip**.

DO NOT ADD ANYTHING BEFORE OR AFTER THE MCGILL ID.

> Every time you submit a new file on *myCourses*, your previous submission will be overwritten. We will only grade the **final submitted file**, so feel free to submit often as you progress through the assignment.

In accordance with article 15 of the Charter of Students' Rights, students may submit any written or programming components in either French or English.

1.2 Late policy

All the assignments are to completed *individually*. You are expected to respect the late day policy and collaboration/plagiarism polices, discussed below.

> **Late Day Allotment and Late Policy**
>
> Every student will be allowed a total of **six (6)** late days during the entire semester, without penalty. Specifically, failure to submit a (valid) assignment on time will result in a late day (rounded **up** to the nearest day) being deducted from the student's late day allotment. Once the late day allotment is exhausted, any further late submissions will obtain a score of **0%**. Exceptional circumstances will be treated as per McGill's Policies on Student Rights and Responsibilities.

If you require an accomodation, please advise McGill Student Accessibility and Achievement (514-398-6009) as early in the semester as possible. In the event of circumstances beyond our control, the evaluation scheme as detailed on the course website and on assignment handouts may require modification.

1.3 Collaboration & Plagiarism

Plagiarism is an academic offense of misrepresenting authorship. This can result in penalties up to expulsion. It is also possible to plagiarise *your own work*, e.g., by submitting work from another course without proper attribution. **When in doubt, attribute!**

You are expected to submit your own work. Assignments are individual tasks. This does not need to preclude forming an environment where you can be comfortable discussing **ideas** with your classmates. **When in doubt, some good rules to follow include**:

- fully understand every step of every solution you submit,
- only submit solution code that was *written* (not copy/pasted/modified, not `ChatGPT`'ed, etc.) by you, and
- never refer to another student's code — if at all possible, we recommend that you avoid *looking* at another classmates code.

McGill values academic integrity and students should take the time to fully understand the meaning and consequences of cheating, plagiarism and other academic offenses (as defined in the Code of Student Conduct and Disciplinary Procedures — see these two links).

> Computational plagiarism detection tools are employed as part of the evaluation procedure in ECSE 446/546. Students may only be notified of potential infractions at the end of the semester.

Additional policies governing academic issues which affect students can be found in the Handbook on Student Rights and Responsibilities.

2 Mesh Lights

The most notable change from the previous assignment is the inclusion of **mesh lights**, rather than environmental emitters, to light our scene.

Mesh lights are treated as "first-class" physical objects in the scene, meaning they are associated with scene geometry (just like every other non-emitting object) and with emissivity associated to their *material* property `Material.Ke`.

Mesh Lights.

3 BRDF Importance Sampling

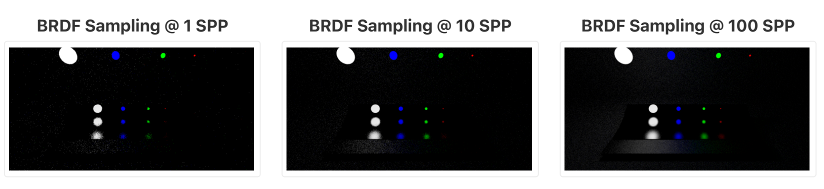
You will begin by modifying your BRDF Importance Sampling direct illumination estimator to support mesh lights.

Recall the formula for this MC estimator:

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

where you previously evaluated $L_e(\mathbf{x}, \boldsymbol{\omega})$ as (spatially-invariant) environment map emission $L_{env}(\boldsymbol{\omega})$.

With mesh lights, however, you will now need to check if your shadow rays intersects an emissive object when evaluating V and L_e. Concretely, your visibility check $V(\mathbf{x}, \boldsymbol{\omega})$ now checks for a successful hit with an emissive object, rather than a miss (i.e., an environmental "hit").



> **Deliverable 1 [10 points]**
>
> - Modify your `A2Renderer`'s `render()` routine to support mesh lights with BRDF importance sampling.

> Your `A3Renderer` instantiates an `A2Renderer` object. You do not need to re-implement BRDF importance sampling, you can just call your previous method.

4 Light Importance Sampling

The next Monte Carlo estimator that you will implement is one that conducts mesh light importance sampling.

A sampled emissive triangle (green) on the mesh light.

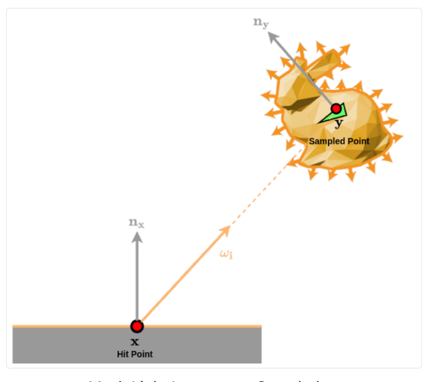
To do so, we will proceed in two steps:

1. first, you must choose an emissive triangle from your mesh by sampling according to a probability distribution function $p_{\text{triangles}}(t)$ over triangles; to do so, build a (1D) cumulative distribution function over all emissive triangle areas, and perform *inversion sampling* using 1D binary search over the (inverted) CDF, using a canonical uniform variate $\xi_{\text{triangle}} \in [0, 1]$; and,
2. after choosing your emissive triangle (i.e., sampling a triangle i proportional to the marginal density $i \sim p_{\text{triangle}}$), you can sample a point uniformly over the surface area of the (marginally) sampled triangle according to the (conditional) density $p(\mathbf{y}|i)$:
   - given canonical uniform random variates $\xi_0, \xi_1 \in [0, 1]$, compute barycentric coordinates b_0 and b_1 as follows:

$$\text{if } \xi_0 < \xi_1: \quad b_0 = \frac{\xi_0}{2} \quad \text{and} \quad b_1 = \xi_1 - b_0$$

$$\text{otherwise:} \quad b_1 = \frac{\xi_1}{2} \quad \text{and} \quad b_0 = \xi_0 - b_1$$

   - with the third barycentric coordinate as $b_2 = 1 - b_0 - b_1$, you arrive at the sampled surface point on the mesh $\mathbf{y} = b_0\mathbf{v}_0 + b_1\mathbf{v}_1 + b_2\mathbf{v}_2$, where $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2$ are triangle i's vertices.



Mesh Light Importance Sampled $\omega_\mathbf{i}$

The probability of your sample is the product of the marginal probability of choosing the emissive triangle from the mesh light p_{triangle} propotional to its area, times the conditional probability $p(\mathbf{y}|t)$ of sampling a point uniformly on the surface of that triangle:

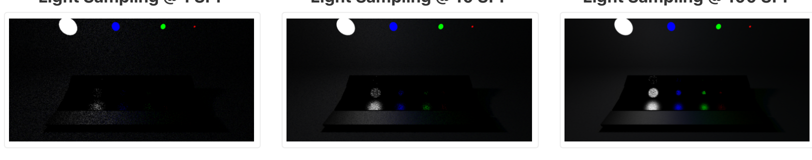
$$p_{\text{light}}(\mathbf{y}) = p_{\text{triangle}}(t)\, p(\mathbf{y}|t) = \left(A_t \Big/ \sum_i^{N_{\text{triangles}}} A_i\right)\left(\frac{1}{A_t}\right) = \frac{1}{\sum_i^{N_{\text{triangles}}} A_i} = \frac{1}{\text{total emissive area}},$$

where A_i denotes the area of a triangle i and t is the index of the sampled triangle.

The light importance sampled Monte Carlo estimator is thus:

$$L_o(\mathbf{x}, \boldsymbol{\omega}_o) \approx \frac{1}{N} \sum_{j=1}^{N} \frac{L_e(\mathbf{x}, \boldsymbol{\omega}_j)\, V(\mathbf{x}, \boldsymbol{\omega}_j)\, f_r(\mathbf{x}, \boldsymbol{\omega}_o, \boldsymbol{\omega}_j)\, \max(\mathbf{n}_\mathbf{x} \cdot \boldsymbol{\omega}_j, 0)\, \max(\mathbf{n}_{\mathbf{y}_j} \cdot -\boldsymbol{\omega}_j, 0)}{p_{\text{light}}(\mathbf{y}_j)\, ||\mathbf{x} - \mathbf{y}_j||^2},$$

where $\boldsymbol{\omega}_j = (\mathbf{y}_j - \mathbf{x}) / ||(\mathbf{y}_j - \mathbf{x})||$ and the integration points $\mathbf{y}_j \sim p_{\text{light}}$ are drawn over emissive surfaces.



> **Deliverable 2 [10 points]**
>
> - Modify your `A3Renderer`'s `render()` routine to support mesh light importance sampling.
>   - Implement the `compute_triangle_area()` function in the `MeshLightSampler` class, which returns the area of a triangle given its three vertices.
>   - Implement the `compute_cdf()` function in the `MeshLightSampler` which populates the `self.cdf` class variable (in place); this is the CDF of your emissive triangles, from which you will sample.
>   - Implement the `sample_emissive_triangle()` function in the `MeshLightSampler` class, which randomly samples an emissive triangle, using the precomputed CDF and the inversion method.
>   - Implement the `evaluate_probability()` function in the `MeshLightSampler` class, which returns the probability of your sampled direction ω_j (converted from a sampled point $\mathbf{y}_j$).
>   - Implement the `sampled_mesh_lights()` function in `MeshLightSampler`, which returns both a direction ω_j but also the ID of your sampled emissive triangle.

> - You will need the ID of the sampled emissive triangle during your visibily check.
> - You do not need to implement every helper function as described: these helpers are put in place to help guide you towards the solution; you can feel free to implement the entirety of this deliverable as you see fit.

`Taichi` loops are highly optimized, and are not guaranteed to run in sequential order. In fact, they **almost never** will. If you want to perform an operation that requires specific sequential processing (e.g., computing a CDF), you can force `taichi` to run a specific loop in "serialized" mode, which will force the order of operations to be respected, by decorating the loop as follows:

```
sum = 0.0
ti.loop_config(serialize=True)
for i in range(emissive_triangles):
      sum += pdf[i]
      cdf[i] = sum
```

This obviously runs slower than a regular loop, however this is the only way to guarantee a sequential operation. You will need to be mindful of `taichi`'s optimization when performing certain tasks in this assignment.

5 Multiple Importance Sampling

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.

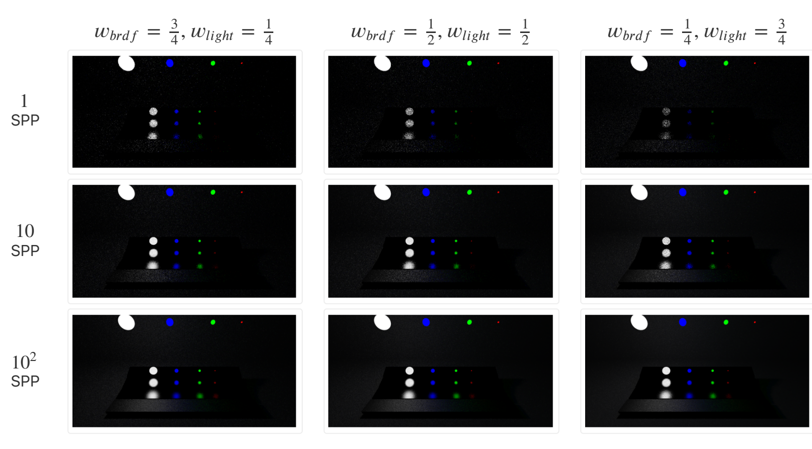
While this general MIS formulation is suitable, in the context of our 1-sample per-render-iteration progressive rendering setting, it poses a problem: with the smallest setting of $n_f = n_g = 1$, each iteration will generate two samples.

Since we absolutely wish to maintain the 1-sample-per-pass property 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, in the $w_{brdf} = 1/2$ and $w_{light} = 1/2$ setting. More generally, for weighins that favour one of the strategies more than the other (i.e., where $w_{brdf} \neq w_{light}$), you can employ the 1-sample strategy discussed during during the lectures, where you first stochastically choose a strategy and then sample according to it (with an appropriate 1-sample MIS weight). In these cases, you can explore the design space of trade-offs between pure-BRDF vs. pure-Light Importance Sampling, and every combined strategy in between them (as illustrated, below).



> **Deliverable 3 [10 points]**
>
> - Update your `A3Renderer`'s `render()` routine to support (1-sample) MC estimation with an MIS estimate that combines BRDF and Light sampling distributions.

ECSE 546 Students Only

6 Environment Light Importance Sampling

The final deliverable (for ECSE 546 students) for this assignment will be environment light importance sampling. Rather than rendering a scene directly, we will in the environment map space to validate that our importance sampling routine generates appropriate sampling distributions — we leave the exercise of integrating these samples into a renderer to those interested students.

To perform our experiments, we provide you with a custom visualization pipeline, `EnvISRenderer`. This visualizer samples points on the environment map, and displays a normalized distribution of your points on a latitude-longitude image map.

In order to importance sample an enivronment map, you will need to implement the following marginal-conditional sampling procedure, as discussed in class:

- Step 1: Initializing the joint probabilty distribution $p(\phi, \theta)$

  The first step is to convert the RGB environment map into a scalar luminance map. There are many ways to do so (e.g., `max` or average RGB value) but we will perform the following **luminance** conversion:

$$\text{Luminance}(RGB) = 0.2126R + 0.7152G + 0.0722B\,.$$

  Once you have your luminance map, you then need to scale it by $\sin(\theta)$ to arrive at the joint probabilty distribution $p(\phi, \theta)$ we will aim to importance sample:

$$p(\phi, \theta) = \text{Luminance}\big(L_{env}(\phi, \theta)\big)\ \sin(\theta)$$

- Step 2: Precomputing the marginal probabilty distribution $p(\theta)$

  Next, we need to compute and store the marginal distribution $p(\theta) = \int_0^{2\pi} p(\phi, \theta) d\phi$, which you can generate by summing and normalizing over each row of your joint distribution.

- Step 3: Conditional probabilty distribution $p(\phi|\theta)$

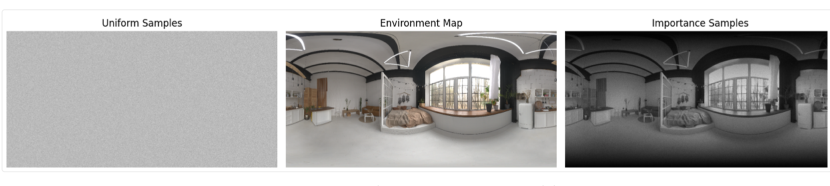
  When it comes time to evaluate the 1D conditional probability distribution $p(\phi|\theta) = \frac{p(\phi,\theta)}{p(\theta)}$, you already have the joint and marginal PDFs at your disposal.

- Step 4:

  The final step is to actually perform the (importance) sampling! To sample proportional to your luminance map, first sample a $\theta_i \sim p(\theta)$ and then sample $\phi_i \sim p(\phi|\theta_i)$. You'll do this by building 1D CDFs for the marginal (one) $p(\theta)$ and (many) conditional $p(\phi|\theta)$ PDFs, and then sample them using (discrete) inversion sampling.

Importance Sampled Environment Map @ 1 Million Samples

Importance Sampled Environment Map @ 10 Million Samples



Importance Sampled Environment Map @ 100 Million Samples

> **Deliverable 4 [10 points]**
>
> - Implement the `precompute_scalar()` function in the `Environment` class which computes the luminance PDF $p(\phi, \theta)$ and stores it in the `self.image_scalar` field
> - Implement the `precompute_marginal_ptheta()` function in the `Environment` class to precompute $p(\theta)$ and store it the `self.marginal_ptheta` field
> - Implement the `precompute_conditional_p_phi_given_theta()` function in the `Environment` class which computes the conditional PDFs $p(\phi|\theta)$ and stores them in the `self.conditional_p_phi_given_theta` field
> - Implement the `precompute_cdfs()` function in the `Environment` class which computes the marginal and conditional CDFs and stores them in the `self.cdf_ptheta` and `self.cdf_p_phi_given_theta` fields
> - Implement the `sample_theta()` and `sample_phi()` functions in the `Environment` class which will sample a ϕ and θ using inversion sampling of the CDFs
> - Implement the `importance_sample_envmap()` function in the `Environment` class which will return a normalized $[u, v]$ texture coordinate for your sample on the environment map's latitude-longitude coordinate system

> Since you are sampling from a discretized CDF, when computing the normalized $[u, v]$ coordinates, you will need to linearly interpolate (i.e., using a `lerp()`) the θ and ϕ values that you sampled.

7 You're Done!

Congratulations, you've completed the 3rd assignment. Review the submission procedures and guidelines at the start of this handout before submitting your assignment solution.