ECSE 446/546: Realistic/Advanced Image Synthesis

Assignment 4: Path Tracing
Due: Monday, December 5th, 2022 at 11:59pm EST on myCourses
Final weight: **30%**

Contents

1 Assignment Policies and Submission Process
2 Assignment Overview and Notes
3 Implicit Path Tracing
4 Explicit Path Tracing
5 You're Done: Congratulations!

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`.

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

2 Assignment Overview and Notes

You will implement a path tracer that builds atop the 1-sample progressive rendering framework in Assignment 3. The differences between the Assignment 4 and 3 base codes are much fewer than between Assignments 3 and 2:

- as in A3, your implementation will focus primarily on the `Scene.render` routine, which will generate either a single path (i.e., for the implicit path tracing deliverable) or a single "traced ray" path (i.e., for the ECSE 546-only explicit path tracing deliverable) per pixel; the `Scene.progressive_render_display` routine will then accumulate, average, display and save the results for many such paths;
- the scene and material parameterizations in the code, e.g., how `brdf_params` and `Le` are used to describe BRDF and emitter properties, remains unchanged compared to A3:
  - as in A3, we no longer explicitly define lights in the scene, but rather instantiate emissive scene objects using a non-zero `Le` property; the `Scene.add_geometry` routine continues to additionally call `Scene.add_light` to build a list of light sources for easier access in your `Scene.render` routine, and
  - this assignment will similarly support both diffuse and glossy Phong BRDFs;
- as with A3, you are free to augment your implementation of `Scene.progressive_render_display` (as well as the `__main__` tests, as with all the previous assignments) as you see fit, i.e., to facilitate debugging/logging/etc. We will only evaluate the `Scene.render` routine;
- the most notable, albeit subtle, API changes occur in the `Scene.render` parameter list (and, subsequently, in `Scene.progressive_render_display`, for parameter passthrough purposes):
  - the `sampling_type` parameter now accepts two core enumerations, `IMPLICIT_BRDF_SAMPLING` and `EXPLICIT_LIGHT_BRDF_SAMPLING` corresponding to the implicit and explicit path tracing (ECSE 546-only) code paths, as well as two optional enumerations (`IMPLICIT_UNIFORM_SAMPLING` and `EXPLICIT_UNIFORM_SAMPLING`); similarly to A3, you will branch on this parameter when implementing *your algorithm(s)*,
  - an additional `num_bounces` parameter specifies the maximum number of *light bounces* that your path trace will support; here, your implementation will terminate path lengths in a biased manner as opposed to an unbiased manner (e.g., using Russian Roulette), and you do not need to vectorize over light bounces: use a `for` loop over the number of bounces in your `Scene.render` routine. Crucially, `num_bounces = 0` should yield a direct emission-only rendering, `num_bounces = 1` generates direct illumination (and direct emission), `num_bounces = 2` generates direct illumination and one-bounce of indirect illumination (and direct emission), and so forth.

Be mindful of the points above when copying/adapting code from your previous assignments into the A4 base code.

ECSE 446 students will implement one (1) MC estimator corresponding to an implicit path tracer that uses *BRDF importance sampling* to sample the recursive path vertices; ECSE 546 students will additionally implement an explicit path tracer that uses *spherical light importance sampling* for the direct illumination estimation, and *BRDF importance sampling* to sample the recursive path vertices.

As before, you may choose to begin with a basic, uniform spherical PDF MC estimator(s), motivating our inclusion of the `IMPLICIT_UNIFORM_SAMPLING` and `EXPLICIT_UNIFORM_SAMPLING` enumerates — we will not grade any of your uniform samplers, and so their implementation is completely optional.

> (i) If you are confident in your BRDF and light importance sampling solutions from Assignment 3 then — unlike in A3 — starting with a uniform sampler may **not** actually be a judicious use of your time.

3 Implicit Path Tracing

You will implement a 1-sample/path MC estimator of the (recursive) global illumination equation:

$$L(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_{\mathcal{S}^2} L(\text{trp}(\mathbf{x}, \omega_i), \omega_i) \cdot f_r(\mathbf{x}, \omega_i, \omega_o) \cdot \max(\mathbf{n} \cdot \omega_i, 0) \, d\omega_i \ ,$$

as

$$L(\mathbf{x}, \omega_o) \approx L_e(\mathbf{x}, \omega_o) + \frac{L(\text{trp}(\mathbf{x}, \omega_i), \omega_i) \cdot f_r(\mathbf{x}, \omega_i, \omega_o) \cdot \max(\mathbf{n} \cdot \omega_i, 0)}{p(\omega_i)} \ ,$$

where the ω_i will be drawn using **BRDF importance sampling**, i.e., $\omega_i \sim p(\omega)$ or $f_r(\mathbf{x}, \omega_i, \omega)$.

In the implicit path tracing scenario (with biased maximum path length), the incremental path construction terminates under only a few conditions:

1. $\text{trp}(\mathbf{x}, \omega_i) = \mathbf{y}$ lies on a light source, yielding a (non-recursive, deterministic) value for $L(\text{trp}(\mathbf{x}, \omega_i), \omega_i) = L_e(\mathbf{y} \to \omega_i)$,
2. $\text{trp}(\mathbf{x}, \omega_i)$ exits the scene (i.e., does not intersect any object), or
3. you have reached the `num_bounces` path length limit before hitting a light.

As in A3, you will treat both diffuse and Phong BRDFs, with the `brdf_params` property of the `Scene` geometry comprising either the diffuse albedo ρ_d or the glossy albedo and Phong exponent (ρ_s, α), depending on the value of `brdf_params[1]`:

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

where $\omega_o = 2(\mathbf{n} \cdot \omega_i)\,\mathbf{n} - \omega_i$ with ω_o oriented from the currently sampled path vertex to the previous path vertex, i.e., $\omega_o = -\omega_{i-1}$: note an abuse of notation with ω_{i-1} referring to the direction sampled during the **previous bounce** of the iterative path construction process.

> (i) A common design pattern in implicit path tracing implementations is to incrementally update the path `throughput`, starting with an initial value of $(1, 1, 1)$ and updated according to `throughput` $\leftarrow \left(f_r(\mathbf{x}, -\omega_{i-1}, \omega_i) \cdot \max(\mathbf{n} \cdot \omega_i, 0) \right) / p(\omega_i)$; then, upon termination, the final 1-sample/path estimate is set as either `L = throughput * L(y, -ω_i)` or zero (e.g., when the recursive ray does not hit an emitter).
>
> Since your vectorized path tracer will incrementally build a **set of paths**, you may find it helpful to additionally track a vector of booleans that specify whether each per-pixel path is still "active".

> **Deliverable 1 [30 points]**
>
> - Implement an implicit path tracer under a `sampling_type == IMPLICIT_BRDF_SAMPLING` conditional branch in `Scene.render`;
> - Sample recursive scattering directions using BRDF importance sampling;
> - Note that, unlike A3, it is simpler to allow the indefinite path construction process to implicitly treat multiple light sources, as opposed to looping over them in your estimator.
> - Your estimator code should leverage all of the subtleties (e.g., proper shadow acne treatment) you have developed in prior assignments.

We include three variants of the popular Cornell Box test scene in the `__main__` routine, which you can augment/edit as you see fit: one base scene with only diffuse objects and a large spherical light, another set of test scenes with medium- and small-sized lights, and a final scene with a glossy back wall and near-specular large box.

The images on the right illustrate the noticeable (but arguably marginal) impact of doubling the sampling rate, even in the lower-variance setting of only one indirect bounce.

`num_bounces = 2, total_spp = 512` | `num_bounces = 2, total_spp = 1024`

Increasing the number of bounces, with fixed sampling rate, increases variance but also (and expectedly) the brightness due to indirect illumination.

`num_bounces = 3, total_spp = 1024` | `num_bounces = 4, total_spp = 1024`

As we shrink the size of the light (whilst keeping its power constant), implicit path tracing has greater difficulty "finding the light", resulting in higher variance results (at fixed sampling rates).

`num_bounces = 2, total_spp = 1024` *Medium-sized light (r = 38)* | `num_bounces = 2, total_spp = 1024` *Small-sized light (r = 18)*

These three final (purposefully larger) images illustrate glossy reflections — of varying degree — on the back wall and large box. Notice how, when artificially biasing the maximum path length, the near-specular reflection of the small box and ground on the large box both appear to have one less bounce of light than when they are viewed directly by the camera. Can you reason as to why this is?

`num_bounces = 2, total_spp = 1024`, *Glossy Scene*

`num_bounces = 3, total_spp = 1024`, *Glossy Scene* | `num_bounces = 4, total_spp = 1024`, *Glossy Scene*

ECSE 546 Students Only

4 Explicit Path Tracing

ECSE 546 students will additionally implement an explicit path tracer, separating out the direct (non-recursive) and indirect (recursive) lighting contributions as:

$$L(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + L_{\text{direct}}(\mathbf{x}, \omega_o) + \underbrace{\int_{\mathcal{S}^2(\mathbf{x})} L(\text{trp}(\mathbf{x}, \omega_i), \omega_i) \cdot f_r(\mathbf{x}, \omega_i, \omega_o) \cdot \max(\mathbf{n} \cdot \omega_i, 0) \, d\omega_i}_{L_{\text{indirect}}(\mathbf{x}, \omega_o)} \ ,$$

where the (non-recursive) direct illumination contribution $L_{\text{direct}}(\mathbf{x}, \omega_o)$ will be computed using **light importance sampling** and with explicit for loops over light sources (as ECSE 446 and the (recursive) indirect term $L_{\text{indirect}}(\mathbf{x}, \omega_o)$ is computed using **BRDF importance sampling**. Here, unlike the implicit case above, recursive path vertices should be rejected if they contribute (locally) to direct illumination, adding a certain excitedness to the logic of when the vectorized path construction should terminate; this is denoted mathematically, in part, by the exclusion of points $\mathbf{y} \in \mathcal{M}_e$ on the light from the integration domain of the recursive indirect contribution, i.e., where $L(\text{trp}(\mathbf{x}, \omega_i), \omega_i) = L_e(\mathbf{y} \to \omega_i) \neq 0$.

Validation against the implicit path tracing results can be used to confirm that the explicit path tracer converges to the same result (albeit, usually, with far fewer accumulated samples per pixel). As discussed in class, recall that — unlike the implicit case — each 1-sample estimate that your `Scene.render` explicit path tracer generates may potentially correspond to a contribution from more than one energy carrying light path.

> **Deliverable 2 [10 points]**
>
> - Implement an explicit path tracer under a `sampling_type == EXPLICIT_LIGHT_BRDF_SAMPLING` conditional branch in `Scene.render`;
> - Evaluate direct illumination explicitly by sample directions towards the spherical light using light importance sampling, explicitly iterating over all light sources (i.e., instead of stochastically selecting light sources).
> - Sample recursive scattering directions using BRDF importance sampling, rejecting those samples that strike an emitter.
> - Your estimator code should leverage all of the subtleties (e.g., proper shadow acne treatment) you have developed in prior assignments.

The images on the right illustrate the significant variance reduction — in this scene — that explicit path tracing brings compared to implicit path tracing. Note the increased sample efficiency.

`num_bounces = 2, total_spp = 1` | `num_bounces = 2, total_spp = 18`

While increasing the path length still leads to higher variance, comparatively, explicit path tracing still outperforms implicit path tracing in this scene; here, even at prohibitively low sampling rates, the overall transport is arguably discernible.

`num_bounces = 3, total_spp = 1` | `num_bounces = 4, total_spp = 1`

Increasing the sampling rate leads to magnified gains, compared to implicit tracing, since explicit direct contributions compound with each added per-pixel "sample" and each additional bounce: recall that direct contributions at path vertices past primary shading points contribute to indirect transport from the camera's point of view; explicit light sampling — especially on diffuse surfaces (recall conclusions from MIS in the Veach scene, as in A3) — helps to reduce indirect lighting variance, too.

`num_bounces = 3, total_spp = 18` | `num_bounces = 4, total_spp = 18`

Shrinking the light still increases variance, however to a much lower degree compared to implicit path tracing. This is in large part due to the effectiveness of light importance sampling with diffuse reflectance. Can you predict whether the strength of this contrast would remain as we increase the specularity of surfaces?

`num_bounces = 3, total_spp = 18` *Medium-sized light (r = 38)* | `num_bounces = 4, total_spp = 18` *Small-sized light (r = 18)*

`num_bounces = 2, total_spp = 1024`, *Glossy Scene*

As above, the three last images use the glossy variant of our Cornell Box. Apart from the clear variance reduction benefits, note the difference in the structure of high-variance regions in these results compared to their implicit path tracing counterparts. Can you reason about why we are, e.g., noticing fireflies on the right side of the ceiling?

`num_bounces = 3, total_spp = 1024`, *Glossy Scene* | `num_bounces = 4, total_spp = 1024`, *Glossy Scene*

5 You're Done: Congratulations!

Congratulations, you've completed the 5th **and final** 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.

Generated by Markdown | XL