CS667 Homework 5
(due 13 November)

For this homework you will implement global illumination and participating media in the ray tracer from Homework 4. You can work from my framework code, which is posted on CMS, or from your own Homework 4 (but you will still want the framework code for this assignment, which contains a number of added classes related to path tracing and media).

The provided code includes a rudimentary path tracer that reflects from surface to surface choosing random directions on the hemisphere—basically, a recursive ProjSolidAngleIlluminator.

1. Implement KajiyaPathTracer, which does Kajiya-style path tracing by sampling direct and indirect illumination separately. You can use the existing DirectIlluminator mechanism for direct illumination and sample the indirect using BRDF-proportional sampling.

2. Implement MIPathTracer, which does multiple-importance based path tracing. That is, it sends two rays at each intersection, one by luminaire sampling and one by BRDF sampling, and uses both to do MI for direct illumination while using the second for indirect.

The provided code also includes a direct-lighting renderer called DirectOnlyMediumRenderer that handles a participating medium in the context of direct lighting and single scattering. It relies on a Medium to handle sampling ray segments and a DirectIlluminator to handle sampling incident light at an interaction point. Out of the box, HomogeneousMedium just samples rays uniformly, and ProjSolidAngleIlluminator (yes, its name is now outdated) supports scattering, but only by uniform sampling over the sphere.

3. Improve the implementation of HomogeneousMedium so that it performs importance sampling according to attenuation.

4. Implement the directScattering method in LuminairesIlluminator, BRDFIlluminator, and MultipleIlluminator, analogously to how you implemented directIllumination in the previous assignment. LuminairesIlluminator samples the sources by area; BRDFIlluminator (another outdated name) samples according to the phase function; MultipleIlluminator chooses between the two and uses multiple importance sampling to combine the results.

The final touch is to merge DirectOnlyMediumRenderer with KajiyaPathTracer to end up with a path tracer that seamlessly handles surface inter-reflection, volume multiple scattering, and surface–volume and volume–surface illumination.

5. Implement KajiyaMediumPathTracer, which is a Kajiya-style path tracer that supports a participating medium. Being a path tracer, it should not branch, but it should compute direct illumination at each surface or volume interaction point. Like DirectOnlyMediumRenderer, for each ray it casts it should randomly choose between sampling volume scattering and computing surface reflection, letting the Medium determine the probabilities according to attenuation. Like KajiyaPathTracer, it should rely on a DirectIlluminator to handle the direct illumination (and now direct scattering) computation.

This last part requires little new code once the previous parts are working. If you like, you can go ahead and implement MIMediumPathTracer; it should be fairly straightforward once you've done everything else.

Implementation

This assignment is mostly about implementing importance sampling schemes in one domain after another. My best advice is to change one thing at a time, verifying that you can still compute the same answers after each change. Write down the domain, the integrand, and the probability density for each integral and you can't go far wrong.

There are a number of test scenes on the web site, mainly centered around the Cornell Box, a common test scene for indirect illumination.