Week 4

The key subject this week is Monte Carlo integration, that is, to solve integrals by sampling. With sampling comes the ability to render soft shadows and indirect illumination. We will focus on soft shadows in the following exercises.

Learning Objectives

- Apply Monte Carlo integration in rendering.
- Sample points on a triangle mesh.
- Sample directions using a cosine-weighted hemisphere.
- Render soft shadows including the effects of ambient occlusion.

Ray Tracing

In the exercises for Week 1, the Lambertian shading for area lights was only an approximation. Monte Carlo techniques can eliminate this inaccuracy.

- Load the Cornell box (`CornellBox.obj`) and the blocks inside it (`CornellBlocks.obj`) into your ray tracer.
- Instead of the simplified area light sampler used in Week 1, implement true sampling of a random position on the surface of the area light (first sample a random triangle, then sample a random position on that triangle). Make sure that your shader for Lambertian materials takes a number of samples and estimates the value of the direct lighting integral properly using Monte Carlo integration theory. (In the `pathtrace` project of the course framework, update the `sample` function in `AreaLight.cpp` and, if necessary, the `shade` function in `Lambertian.cpp`.)
- As a result you should obtain an image of the Cornell box where the blocks cast soft shadows. Pick a number of samples that results in smooth soft shadows and save the image. (The framework is set up to sample area lights 4 times per pixel sample by default. See line 274 of `RenderEngine.cpp`.)
- Load the Stanford bunny (`bunny.obj`) and use the sun and sky models implemented in Week 2.
- Ambient occlusion is essentially the idea that the environment (the background) is an infinitely distant ambient area light. Compute ambient occlusion by tracing rays in directions sampled on the hemisphere over each surface point. Consider the colour returned by the sky model to be the incident illumination if the ray is not occluded. (In the framework, implement the `shade` function in `Ambient.cpp` and the `sample_cosine_weighted` function in `sampler.h`. Note that the ambient occlusion shader is used when you press '2' on the keyboard.)
- Repeat the sequence of bunny images from Week 2, but this time with both direct and ambient illumination. In addition, render the Cornell box with the Stanford bunny (`bunny.obj`) and an elephant (`justElephant.obj`) instead of the standard blocks, and with a flat light blue background colour instead of the sun and sky models. Use diffuse materials for the bunny and the elephant. Pick a number of samples that results in smooth illumination and save the image. (The framework is set up to trace 5 occlusion rays per pixel sample by default. See line 99 of `RenderEngine.cpp`.)

Week 4 Deliverables

Cornell box images (with blocks that cast soft shadows, with bunny and elephant shaded by ambient occlusion). Sequence of images captured at several points in time during a day with the bunny on a green plane illuminated by the sun and the sky. Include relevant code and render log (please give details about the number of samples per pixel: number of jitter samples, shadow/occlusion samples, etc.).

Reading Material

The curriculum for Week 4 is

P Chapter 13 except Section 13.4. *Monte Carlo Integration I: Basic Concepts*.
P Section 14.6.3. *Area Lights*. (Soft shadows.)
P Section 15.1. *Direct Lighting*.

Alternative literature available online or uploaded to CampusNet:

- Dutré, P. *Global Illumination Compendium*. Lecture Notes, Katholieke Universiteit Leuven, September 2003. http://www.cs.kuleuven.ac.be/~phil/GI/
- Shirley, P., Wang, C., and Zimmerman, K. Monte Carlo techniques for direct lighting calculations. *ACM Transactions on Graphics 15*(1), pp. 1–36, 1996.

Additional resources:

- Landis, H. Production-ready global illumination. In RenderMan in Production, ACM SIGGRAPH 2002 Course Notes, Chapter 5, pp. 87-101, 2002.
- Pharr, M., and Green, S. Ambient Occlusion. In *GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics*, Chapter 17, Addison-Wesley, 2004.
  http://http.developer.nvidia.com/GPUGems/gpugems_ch17.html