CMSC 635 programming assignment #3:

Out: Mar. 18

Due: Apr. 1 2013 11:59 pm

Photon

Figure 1. Ratatouille kitchen rendered using photon maps (Copyright 2007 Disney Pixar). Interested in reading about how Pixar uses photon mappings ? See Chapter 5 of "Photon Mapping for Complex Scenes:" http://graphics .pixar.com/library/HQRenderingCourse/paper.pdf.

1. Introduction

IN this assignment, you will add photon mapping support a Monte Carlo raytracer. Photon mapping is a global illumination technique that separates scene geometry from radiance calculation. The algorithm is completed in two passes. First radiance from the light, split into discrete chunks called "photons," is transported through the scene, and is saved in a 3D spatial acceleration data structure called a photon map. Second, the renderer queries the photon map for nearby photons, and use them to computer an estimate for indirect illumination.

Photon mapping can be integrated with just about any rendering algorithm. When properly tuned, photon maps produce results that look visually similar to techniques like path tracing, without requiring as much of a wait. However, they do not converge to an exact solution and will look wrong if not tuned properly.

In this assignment, you will implement a simplified version of the algorithms described by Henrik Wann Jensen in Realistic Image Synthesis Using Photon Mapping. AK Peters, ISBN: 1569911470. July 2001.

We'd prefer G3D implementations. And the supporting code is written in G3D. But you are free to use any libraries you wish in this assignment, as long as the work is yours. Please do not just copy something online. If you look for other resources other than G3D, http://www.pbrt.org/ is a good library (produced by Stanford students) to begin with.

Please start earlier, because the reading and thinking will take a lot of time.

2. Requirement

Your final image must support full global illumination, with caustics from reflection/refraction and color bleeding from diffuse interreflection. Specifically, you must have:

- Recursive Monte Carlo raytracing, with support for reflection, refraction, and soft shadows (already provided in support code)
- Photon map generation, by tracing photons through the scene using Monte Carlo techniques
- Final radiance calculation combining Monte Carlo ray tracing and photon map estimate.

There is demo scene on the G3D website and your results should match that when the same rendering parameters and scene file are used.

If you wish to design a GUI, although we strongly suggest Qt Creator for its auto-complete ability, you are free to use anything else. Some of the simpler ones include GLUI (http://glui.sourceforge.net/). Qt Creator can be found at http://qt-project.org/.

3. Getting Started

Supporting code is available online.

As previously mentioned, the support code already implements a Monte Carlo raytracer. You will implement the scatter() method, which shoots a photon into the scene,, and diffuse(), which uses the photon map to estimate the radiance due to diffuse inter-object reflection (hence the name). Short hints have been left as comments in the support code. We also suggest reading and understanding trace(), the recursive raytracing function, as well as the subroutines it calls directly.

Rendering parameters used by the support code are provided as constant s in app.h. The default parameters were chosen to balance between rendering speed and picture quality. You can get better looking pictures by increasing the photon count and / or increasing the number of direct lighting samples taken per pixel. However, if you do this you'll be waiting a long time to get your picture!

By default, the support code shows you photon map as it is built, and then shows the final rendering as the ray tracer runs. To view just the photon map, press the 1 key on your keyboard; to view just the rendering, press 2.

As in the pathtracing assignment, we recommend referring to the "rendering in Practice" chapter of the textbook (Ch. 32). The example ode given in the book uses G3D 8.x, where the support code uses G3D 9.x, but the concepts carry over. We also expect reading Henrik Wann Jensin's paper on photon mapping [1] to be informative. You may find the G3D 9.x. method Surfel::scatter() useful in this assignment.

4. Extra Credit

You may implement any of the following effects for extra credits:

Tone Mapping
Participating Media
Shadow Photons
Image-Space Photon Mapping
Anisotropic Reflection
Subsurface Scattering
Irradiance Caching
Something else in your mind (But check with Dr. Keqin Wu first).

5. Handing In

You are required to hand in a README file with your project, as a plain text file. It should contain anything that could make the faculty assistant's life easier (e.g., major design choices, known bugs, kludges).

Do not hard code any paths in your program!

To hand in, email your code **in one tarball** to keqin@umbc.edu.

Reference:

[1] Henrik W. Jensen. Global illumination using photon maps, *Proceedings of the Seventh Eurographics Workshop on Rendering*, pp. 21-30, 1996. Accessible online at http://www.csee.umbc.edu/courses/graduate/635/spring13/asgn/03Photon/photon_mapping.pdf