University of Innsbruck
February 8, 2016

703044 PS/1 Advanced Computer Graphics

Final Project Report

Topic: Photon mapper

No.	Matrikel Nr.	Name
1	1316136	Bernhard FRITZ
2	1315183	Mathias HÖLZL
3		Florian TISCHLER

Project goal

This project was about understanding the basic idea about photon mapping and implementing a renderer based on that acquired knowledge.

Detailed description

Path tracing does not produce significant caustics in a reasonable amount of time. The reason for this is that there is only a small probability that a ray starting from camera, hits a surface that actually reflects incoming rays directly through a transparent object straight to the light source. To increase this probability, photons are sent from the light into all directions. When photons hit a surface they will be saved in some sort of data structure (e.g. Kd-tree).

Photon mapping consists of two steps. In the first step, photons are sent out of each light source. These photons are followed throughout the scene until they collide with a diffuse object. Once that happens the intersection point and the incoming direction of the photon is saved (e.g. in a Kd-tree). This step is generally known as photon gathering. After that a ray tracer (e.g. path tracer) is used to determine direct illumination. Whenever a ray intersects with the scene the nearest N photons are determined using the nearest neighbour search algorithm on a Kd-tree containing the photons. This step is called photon gathering. The gathered photons then can be used to estimate incident flux and in the end determine indirect illumination.

Implementation details

Our implementation process:

1. Create a Ray-tracer
2. Setup sphere only scene
3. Simple photon mapping for diffuse materials
4. Per-Object kd-tree to store and access photons
    a. First try using “C-Lib kd-tree”
    b. Better fit **NanoFlann** Kd-Tree lib (C++/template support)
5. Parallelization (Photon generation/Path-tracing)
6. Recursive Ray-tracing
7. Live preview using opengl/freeglut
8. BRDF for Ray-tracing and Photon shooting
    a. Diffuse Materials
    b. Specular/glossy
    c. Translucent/glossy
9. Add Triangle mesh support
10. obj. Model Loading via **TinyObjLoader** lib
11. Testing and demo scene creation

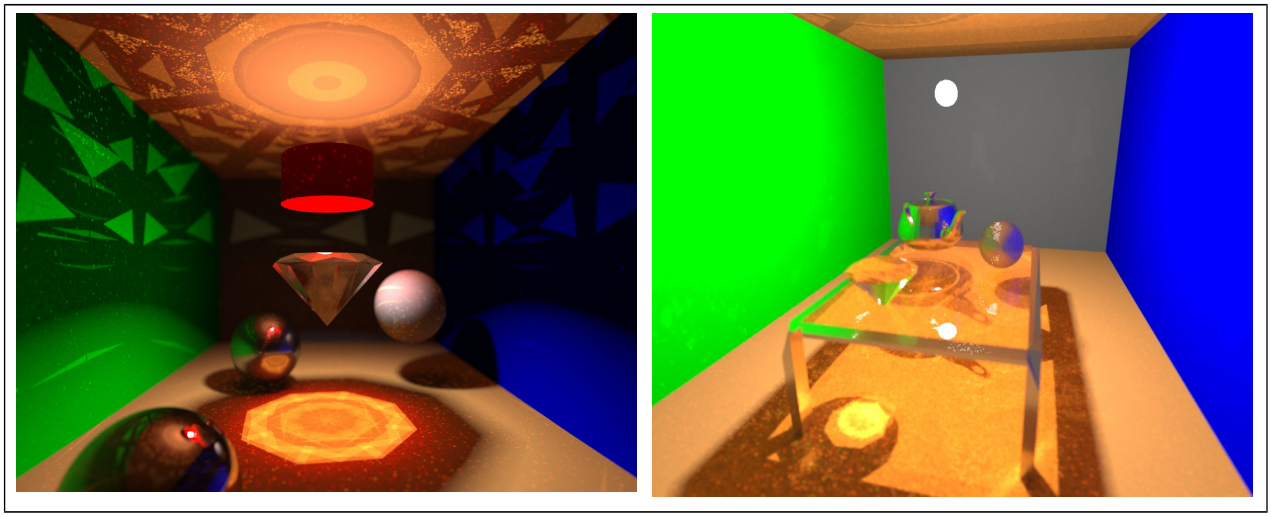
Solution properties:

- Enhanced path tracer
- Photon mapping for handling indirect lighting
- Shadow rays for direct lighting
- Recursive ray tracing for specularity/transmission
- Multithreaded photon map creation and ray casting
- Per object Kd-tree for storing photons

External resources

No.	Name	Description
1	GLM	A C++ mathematics library for graphics programming
2	OpenGL	API for rendering 2D and 3D graphics
3	Freeglut	Window manager for OpenGL
4	NanoFlann	Kd-tree library

Result



Potential improvements

- Getting rid of weird visual artifacts (e.g. red and blue dots on sphere)
- Implementing final gathering
- Adding caustics map