Wendy Chen
CPSC 490
January 26, 2017

# Extended Path Tracing Exploration: Spring 2017

For the second term of my senior project, I intend to build off the results of my project from Fall 2016. At the end of last term, I had studied the theory behind path tracing. Path tracing is a nondeterministic Monte Carlo rendering technique that builds paths in a scene by drawing samples to determine where to go next. Sampling techniques include random sampling, importance sampling, and multiple importance sampling. Methods like Russian roulette can be used to determine when the recursive path tracing process stops. The light intensity at a point in the scene is the result of an initial intensity from a light source multiplied by the throughput of the path.

After studying path-tracing theory, I implemented a unidirectional path tracer in C. This semester, I would like to further refine this implementation and explore optimizations like bidirectional path tracing or Metropolis Light Transport. The work of this semester can be characterized as rendering system design. I am inexperienced in designing larger systems, and by expanding my simple path tracer implementation into a larger but bare bones rendering system, I will need to consider a variety of design decisions.

At the end of the term, I will deliver a path tracer designed from investigating other rendering systems. A gallery of images and the path tracer design will be documented in the final report.

## Project

My project roughly consists of two parts: research of existing rendering system design, and implementation. These two processes may be interleaved, and I expect to follow a think-design-re-implement pattern of work this semester. Some systems I would like to look at include PBRT, Mitsuba, and the skeleton system from UPenn's CIS460 course.

For designing the rendering system, I need to answer questions such as the following: What libraries should I use? OpenGL? Should there be a GUI (ie. PyQt)? Or should the renderer be based in the terminal? How should scene files be specified? Should I adjust the internal data structures? How can I modularize/extend the unidirectional implementation?

In short, I have questions in the following areas:

- **System input:** How should the user interact with the system? How should scene input be specified?
- **Modules/Object-oriented design:** How can the unidirectional path tracer be re-designed to make use of appropriate abstractions? What are the input-output flows between modules (ie. Interfaces between modules)?

- **Optimization:** What sampling techniques should be implemented that will be better than random sampling? How can other algorithms like bi-directional path tracing be implemented?
- **System output:** How should output be delivered to user? What kind of image file will be used? How can the user know the progress of the image being rendered?

After designing and implementing the system, I would like to use it to generate test images. I may compare these test images with results obtained through freely available rendering systems.

While my senior project is nominally on path tracing, this semester's work primarily consists of issues of software design and implementation. I lack such experience, and I hope that I can improve by designing a basic path tracing software that a user (beside me) may find friendly to work with and produce reasonable results.

## Previous Work

The spring term of my CPSC 490 builds off of my work from the previous fall term. I ended the fall term with writing a report on path tracing theory and implementing a simple unidirectional path tracer in C.

## Schedule (~12 weeks)

The tentative schedule lies below. I expect the development to be a process of trial and error, so I may end up deviating from this schedule.

Proposal:
Week 1 (1/22 – 1/28)

Scene specification, parsing:
Week 2 (1/29 – 2/4)

Acceleration:
Week 3 (2/5 – 2/11)

UV-mapping, textures:
Week 4 (2/12 – 2/18)

Materials:
Week 5 (2/19 – 2/25)

Optimized sampling techniques:
(ie. Direct lighting, MIS, indirect lighting)
Week 6 (2/26 – 3/4)
Week 7 (3/4 – 3/11)

~Spring break~

Optimization: bi-directional path tracing
Week 8 (3/26 – 4/1)
Week 9 (4/2 – 4/8)

Refine and report
Week 10 (4/9 – 4/15)
Week 11 (4/16 – 4/22)
Week 12 (4/23 – 4/29)

~Classes End~

## Deliverables

- **Path tracer**: The path tracer in its final design.
- **Gallery of images**: Renders of scenes of varying complexity, and comparison images generated by freely available renderers.
- **Final report**: Write up of implementation details of my path tracer.

## Sources

Like last semester, I aim to most closely follow the PBRT text for guidance. I will also refer to other texts on rendering, path tracing, material models, etc. I may look to SIGGRAPH course notes for help on path tracing.

I would also look at examples of other rendering systems, like:
radsite.lbl.gov/radiance/
www.envymycarbook.com
http://www.mitsuba-renderer.org/