CS52 Assignment 4: Raytracing II

Out: Nov 12. Due: Dec 1.

Introduction

In your fourth assignment, you will add various features to a basic raytracer implementation to support more efficient and higher quality image synthesis.

Differently from previous assignments, you can choose which features you would like to implement; for this reason we will not provide framework code, but you are welcome to use all the code presented in the solution for the previous three homeworks as posted on the web (or any of your previous solutions if you find it easier). We suggest you take either your previous raytracing implementation or the solution to Assignment 1 and start adding feature to it.

The features you can implement are

- Ray-scene intersection
  - Texture mapping
    - Mip-map texture filtering
  - Acceleration Structures
    - Axis-Aligned Bounding Boxes
    - Regular Grids
    - KD Trees
  - Normal Interpolated Meshes
- Distribution Raytracing
  - Soft Shadows
  - Blurred Reflections
  - Depth of Field
  - Motion Blur
  - Quad/Circular Sampling
  - Combined estimators
- Path Tracing
  - Indirect Diffuse Illumination
  - Next Event Estimation (area lights)
  - Importance Sampling (faster diffuse and glossy materials)

Requirements

For this assignment you can choose which of the following features to implement. To receive full grade, the sum of the points of your chosen features should add up to 25 for undergraduate credit and 30 for graduate credit. Any additional feature you implement is considered extra credit and will be counted as such. There is only one limitation to your choice: at least one of the features should come from the distribution raytracing or path tracing group.

For each feature you are to supply scene files and rendered images. Since most of these features will require a lot of time to render, we might not rerun your raytracer to full convergence. This means that we will detract points if you do not hand in images. Also you have to create scenes that clearly show the effect you are trying to display; if such scenes do not clearly show the effect we will detract points, while if the images are particularly nice, we will grant extra credit. You can find some free models at http://graphics.stanford.edu/data/3Dscanrep/ in ply format; you can easily translate them to the xml format with a simple script.

You are to put together a short document and submit it in PDF format. For each feature you implement, this document should contain a clear indication of the feature and a few images to illustrate the feature. You should also include a "before-and-after" comparison, i.e. a set of images showing the scene with and without the feature. For each image please indicate the running time and how many samples where used to compute it (having the before and after image will give us an idea of how fast you have implemented these features). Finally pick one of your images, the one that is most impressive in your opinion, and render a large version (1024x024 will do) with low noise; submit this one separately. *If the report is missing we will detract a considerable number of points.*

1. **Texture Mapping** [**5 points**]: Sphere and triangle primitives should support texture mapping. Triangles should support mapping by interpolating UV sets stored at vertices using baricentric coordinates. Spheres should support mapping using the two spherical angles as the parameters.

2. **Texture Filtering** [**5 points**]: Implement trilinear mip-map based texture filtering. For the purpose of this assignment you can use a distance-based heuristic to determine which mip-map level to lookup. This should come after you have implemented the previous one.

3. **Acceleration Structures** [**10 points**]: Implement one of the following acceleration structures: axis-aligned bounding box hierarchies, regular grids or BSP trees. Demo you implementation by showing speed up in a complex scene (or mesh) of your choice.

4. **Normal Interpolated Meshes** [**5 points**]: Add a mesh data type that supports normal interpolation for triangles based on baricentric coordinates.

5. **Distribution Raytracing - Soft Shadows** [**5 points**]: Add soft shadows to the raytracer using Monte Carlo uniform sampling. You can do so by implementing a square area light and sampling on a square domain. Or you can approximate a spherical light by uniformly sampling a disk oriented along the source to point vector (this changes for every point).

6. **Distribution Raytracing - Blurred Reflections [5 points]**: Add blurred reflection to the raytracer using Monte Carlo uniform sampling of the perturbed the reflected direction. You can do so either using a square uniform sampling or a disk uniform sampling.

7. **Distribution Raytracing - Depth of Field [5 points]**: Add depth of field to the raytracer using Monte Carlo uniform sampling when generating view rays (this subsume antialiasing). You can do so either using a square uniform sampling or a disk uniform sampling.

8. **Distribution Raytracing - Motion Blur [10 points]**: Add motion blur to support animation rendering by adding time to ray intersection routines and performing Monte Carlo integration using uniform sampling in the time domain. To do this you have to support some form of simple animation (translation or rotation is all is needed) and change the ray intersection code to carry time as well as position and direction of the ray.

9. **Distribution Raytracing - Disk Sampling [2.5 points]**: When possible use a disk (circular) domain instead of the quad one. This will be counted only once.

10. **Distribution Raytracing - Combined Esimators [2.5 points]**: Combine multiple estimators in single ray to speed up rendering time. For example show depth-of-field with soft shadows or motion-blurred soft shadows with a small number of rays.

11. **Path Tracing - Indirect Illumination [10 points]**: Implement indirect illumination for a simple scene made of diffuse materials using the path tracing algorithm. Use uniform sampling of the hemisphere and russian roulette as stopping criterion.

12. **Path Tracing - Next Event Estimation [5 points]**: Add area lights to the path tracer using next event estimation (i.e. directly sampling the light area and not the hemisphere). Use lambertian source for this. Note that this is different than the previous soft shadows since these are physically correct, thus have a cosing falloff.

13. **Path Tracing - Importance Sampling [10 points]**: Add importance sampling to the indirect illumination evaluation of your path tracer. You should use cosine distributed samples for lambertian surfaces and weighed cosine and phong distributed samples for a Phong-like material.

Hints

Most the these examples will take a fair amount at time to render. This is particularly true for the distribution raytracing and path tracing cases. You should do your testing using a small number of samples and that increase that number only for the final submission. One other simply way to make things faster in our framework is to mostly use “setToXX” math methods since they reduce Java allocation quite a bit (at the price of less clear code).

Submission

Please send your code, compiled class files, PDF report and final image to cs52@cs.dartmouth.edu.