CS5984 Advanced Computer Graphics Fall 2010
Homework 3: GPU Ray Tracing using kD-Tree

1 Due Date

Homework 3 is due on October 19th, 2010, 11:59pm.

2 Introduction

Spatial index is an important performance acceleration technique for real time ray tracing. In this homework, you are requested to write a ray tracing program using CUDA which is able to render complex triangle meshes indexed by kD-tree.

3 Project requirement

The same as homework 2, your GPU ray casting application should be able to read a scene file, including camera, light source, and all the objects in the scene. For this homework, you also need to specify an object which is represented by a triangle mesh. The mesh is stored as a separate mesh file. The mesh file is in *Wavefront OBJ* format. Please find a collection of sample mesh files here. The description of the file format can be found at this Wiki site.

Please build a kD-tree for all the triangles in the scene. Your program should build kD-tree before it is used in your ray tracing (traversal step) program. Please record the time for building kD-tree and ray tracing separately, and report them in a readme file (please indicate which scene file you used for each recorded timing information). Please at least run your ray tracing using these three mesh files: $castle.obj, teapot.objandworld_curved.obj$.

Your ray casting program should have the following features:

- Read from a scene file. You can define your own scene file format.
- Tracing one ray for each pixel of the image plane (through the center of the pixel). The image plane resolution is 640×480.
- Only calculate the shadow ray color, ignore reflection and refraction ray.
- Your scene should be able to handle triangle mesh. One of your test scene should include at least two triangle meshes.
- Your ray tracing should be accelerated by a kD-tree structure.
- The local illumination model is Phong Shading model, as we discussed in the class. Please decide on the parameter for the light source and object materials by yourself (including specular, diffuse, and ambient).

4 What to Submit

Put your solution in one or more source files. The main file (which includes function main()) should be named homework3.cpp or homework3.cu. Include all your source files and visual studio project file in a zip file and upload to class Scholar site in your own dropbox.

Your submitted zip file should include your test scene files, final rendered images, and a readme file which includes performance (timing) information for your test scene files.

Please do NOT include the compiled EXE files.

Note: Please use CUDA 3.1 for all the homeworks for this course.