

GPGP: Assignment 2 (Ray Tracing and Data Structures on the GPU)

Date handed out: Apr 23, 2007

Due date: May 5, 2007

This assignment shall introduce you to writing data structures on the GPU, with ray tracing as the application. It involves writing a ray tracer on the GPU and using a spatial data structure for accelerating the computations. The preferred implementation language is CUDA, you are welcome to try other languages like OpenGL + Cg/GLSL, DX+HLSL. Make a password protected webpage of your submission and email BOTH the instructors with the URL/password. The webpage has helper source code for loading test models (in 3DS format) and writing out images (in PPM format).

Note: Use a perspective camera. For part 1, first test diffuse shading only, and no shadows. Then add reflections, shadows and simple refractions.

Problem 1: Naïve GPU Ray Tracer: Write GPU ray-primitive intersection routines for 2 types of primitives: spheres and triangles. Test with a simple scene with a single sphere and a box. Add support for secondary and tertiary rays (up to MAX_BOUNCE, where MAX_BOUNCE is a predefined constant), model reflection and refraction. Try MAX_BOUNCE = 1,2,...,16. Measure the performance characteristics of your ray tracer (from Homework #1: FLOPS, GPU Memory Bandwidth) and ray intersections/sec.

Problem 2: Uniform Grid: Use a uniform grid as an acceleration structure. Use the test 3DS models given on the webpage. Select one test model and try different grid resolutions (16, 32, 64, 128). Report the resolution which gives you the best performance.

Extra credit.

Adaptive Grids: Compute an adaptive spatial grid for storing the mesh data. The easiest one would be an Octree. You can also use a KD-tree. Perform performance measurements for a particular test model. Note: You can try out 3rd party data structure librares like GLIFT.

Dynamic Scenes: Make the meshes in the scene move. You will need to update your data structures.