OpenCl Path Tracer

Proposal

The use of general purpose computing on graphics processing units (GPGPU) to accelerate ray tracing has become increasingly ubiquitous. I am proposing an implementation of a path tracer with OpenCL, a C framework which allows for the delegation of operations to the GPU. This provides acceleration to the typical, highly parallelizable path-tracing operations carried out on the CPU. A path tracer recursively fires rays from the camera, and “follows” their recursive, stochastic progression through the rest of the scene. As these rays progress, they accumulate radiance at every intersection with geometry. The image is then capable of simulating global—that is “non-direct”—illumination, provided enough samples are allowed to accumulate. My path tracer will allow for user movement within a scene, with a clearer image being rendered depending on how many “paths” are gathered. The project builds upon my own work with C++, ray tracing, and global illumination, and will require me to become involved with parallel programming on the GPU and path tracing.

Roadmap

This is my tentative roadmap of the project. I will meet with Professor Rushmeier weekly to discuss my progress.

[ Phase 1 ] Research : September 25
I plan to conduct research for 3 weeks and provide an overarching design for the software in that time, including desired classes, their interactions, and functionality.

Books: OpenCL in Action – GPU Programming and Parallel Programming Techniques
Real Time Rendering – Ray/Path Tracing Book

[ Phase 2 ] Math Libraries : October 9
Implement basic Matrix/Vector/Quaternion/Transformation Classes. I may use an existing library, which would shorten this phase and allow for more complex additions to the path tracer at the end of the project.

I will use an outside library for the GUI, and find a scene parser as well.

[ Phase 4 ] Scene Specification cont. / Camera Class – October 23
Continue to hack together user interface and scene format. Code a camera class.

[ Phase 5 ] Rays/Intersections : October 30
Implement primary rays, shadow rays, and loop. At this point, very basic path tracing functionality should be achieved.

Code classes containing basic geometry (most primitives). I, at the minimum, will include code for for box, sphere, and triangle intersections.

[ Phase 7 ] Lights : November 13
Implement “Light” Classes, including Spot, Point, and Diffuse.

[ Phase 8 ] Acceleration : November 20
I will either code a KD-Tree or BVH to hold the elements of the scene, possibly one accelerated by the GPU.
[ Phase 9 ] Materials / Writeup

Add support for more complex materials/images. Troubleshoot any existing issues.

Deliverables

[ A ] GPU Path-Tracer

A GPU Path Tracer that, having read a scene file, allows for, in its GUI, camera rotation and path tracing within the scene.

[ B ] Portfolio of Rendered Images

A collection of interesting renders, including classics such as the Cornell Box, the Stanford Bunny, and the Utah Teapot.

[ C ] Write Up of Results

Detailed report on workflow, equations and algorithms, trials and their corresponding work-through, and, of course, results.

Sources (so far)

1. UPenn Writeup of GPU Path Tracer in CUDA: http://gpupathtracer.blogspot.com/

2. Comprehensive Resource on Programming with OpenCL: *OpenCL in Action*