Advanced Graphics 2016/2017 – Assignment 3

Introduction

For this assignment, you have considerable freedom. Assignment 1 was about light transport fundamentals and setting up the framework for basic ray tracing; assignment 2 dealt with efficient ray/scene intersection (the low-level core of any ray tracing based renderer). For assignment 3 you either build on this, or alternatively, you significantly improve a low-level aspect.

Assignment

Broadly speaking, there are three areas to work on, linked to the theory presented in the lectures:

1. Acceleration structures, options:
   a. expanding on assignment 2, build a high quality acceleration structure for interactive rendering of dynamic scenes. The code should handle rigid animation as well as deformations and quick full rebuilds, with dedicated builders for each type of animation. A real-time demo is expected to demonstrate the capabilities of your system.
   b. significantly improve on the quality of an SBVH by implementing a recent paper. Note: SBVH itself was part of P2; implementing this now is not the purpose of P3.

2. Physically based rendering, options:
   a. implement a path tracer. In terms of variance reduction, this should at least support importance sampling, Russian roulette and next event estimation. Light transport must be correct. Extra points for a solid stratification scheme and/or multiple importance sampling and materials beyond basic Lambertian and pure speculars.
   b. Implement a basic photon mapper or bidirectional path tracer.

3. GPGPU rendering, options:
   a. Implement a basic Whitted-style ray tracer or path tracer on the GPU. This must at least support a BVH to render triangle meshes. Extra points for streaming ray tracing (to be discussed in a lecture after the Christmas holidays).
   b. Experiment with efficient GPU ray traversal, e.g. a rope tree or a stack-less traversal scheme.

If you have a different project in mind that matches the intended scope and general topic of the course, feel free to discuss this with me. Note that if you opt for an experimental project, this does not have to yield superior results, in that case the quality of the research will be assessed.

Language Notes

This assignment may be executed in a programming language of choice. Support on the implementation side will be mostly limited to C++ and C# however, and performance is expected to be optimal for C++ code. Choice of programming language will not play a role in grading.

In general, performance will not be considered, although a fast system is more satisfying. Do not spend a lot of time on multithreading; this will in general not improve your grade.

Practical Details

The deadline for this assignment is Tuesday January 31st, 23.59. You may hand in your assignment up to 24 hours late in exchange for a 1 point penalty. You can hand in your assignment using the Submit system. The materials to submit are:

- your project, including sources and build instructions (if these are not obvious);
- a brief report, detailing implemented functionality, division of work, references and other information relevant to grading your submission.
As with the previous assignments, you may work on this assignment alone, or with one other student.

Feel free to discuss practical details on Slack: [http://uugmt.slack.com](http://uugmt.slack.com). You are not supposed to share complete ray tracers there, but if everyone uses the same optimal ingredients, that would be considered ‘research’.

**Tasks & Grading**

Given the increased freedom for this assignment, grading is going to be somewhat subjective. Generally speaking, a passing grade (6) for this assignment requires:

- A BVH for dynamic scenes, complete with top-level BVH, multiple construction schemes (at least dedicated builders for static, rigid, deforming and freeform animation), along with a demo. Emphasis is on construction efficiency, i.e. a single update of the tree must complete in less than 100ms.
- A CPU path tracer with at least basic variance reduction (IS, NEE, RR), basic materials (Lambert, pure specular, dielectric) and triangle mesh support.
- A functional GPU ray tracer or path tracer with triangle mesh / BVH support.

To obtain additional points:

1. Produce an interesting demo (nice scene or animation);
2. Add image postprocessing;
3. Achieve high performance;
4. Implement advanced variance reduction;
5. Implement advanced variance reduction on the GPU.

Obviously, many other options exist. Contact me if you want to discuss an idea of which you are not sure whether it is worthwhile.

**Purpose**

After successfully completing this assignment, you have obtained theoretical and practical knowledge on algorithms for efficient physically based rendering. This is a solid foundation for further research in the field of graphics.

May the Light be with you,

- Jacco.