Advanced Graphics 2018/2019 – Assignment 1

Introduction

This assignment is a preparation for assignment 2, which involves the construction of a real-time ray tracer. For assignment 1, you will be laying the ground work for this: a test bed for ray tracing. This test bed consists of the ingredients commonly required for ray tracing, plus a basic Whitted-style ray tracer.

Architecture

Ray tracing renders an image based on a scene description, given a camera and a screen plane, using rays originating from the camera and extending through pixels on the screen plane. The desired architecture implements ingredients for this: various classes (camera, screen, ray, light, primitive, …) and methods (creating a normalized ray, intersecting the ray with the scene, plotting pixels, …). For subsequent assignments, we will extend and optimize this functionality. A solid test bed allows you to work on isolated parts of the framework without changing other parts.

Whitted-style Ray Tracing

To demonstrate the suitability of your framework, implement a Whitted-style ray tracer. This involves the construction of a 'renderer' class, which uses the other components to create an image. The Whitted-style ray tracer must at least support shadows and reflections.

Practical Details

The deadline for this assignment is Wednesday December 5th, 23.59. An extended deadline is also available: for a 1 point penalty you can hand in materials until December 6th, 23:59. Please 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.

Please clean your project before handing it in. This is easily achieved by executing `clean.bat` in the template directory. Note that your project must not be open in Visual Studio when you run this file. Please exclude .svn and .git folders from your submission.

You may work on this assignment alone, or with one other student.

You may implement the platform in C++ or C# or any other programming language that you may prefer: do note however that support in working colleges may be limited if you chose an exotic programming language. You may also target other operating systems than Windows, but again, support may be limited.

Feel free to discuss practical details on Slack. You are not supposed to share complete ray tracers there, but if everyone uses the same optimal ray/triangle test, that would be considered 'research'.

Tasks & Grading

A passing grade (6) for this assignment requires:

- implementing a generic and extendible architecture for a ray tracer;
- a ‘free camera’ with configurable position, orientation, FOV and aspect ratio;
- a basic UI or controls to control the camera at run-time;
- support for at least planes and spheres;
- a basic but extendible material class;
- a basic scene consisting of a small set of these primitives;
- a Whitted-style ray tracing renderer to demonstrate and test the architecture.

To obtain additional points, you may work on the following:

1. Support for triangle meshes, using ‘obj’ or ‘fbx’ files to import scenes (max 1pt).
2. Support for complex primitives, complex being a torus or better (max 0.5pt).
3. Texturing on all supported primitives, where the texture is a generic bitmap (max 0.5pt).
4. Flexible lights: besides point lights also spots (0.3), directional (0.2) and IES profiles (0.5 pts).
5. Correct dielectrics: Snell (0.3), Fresnel (0.3) and Beer (0.4 pts).
6. Efficient and generic multi-threading, yielding at least 400% on a quadcore (max 0.5pt).
7. Optimized renderer: render Turner Whitted’s scene (see below) at 20 fps or better (on my laptop) at 512x512 pixels for an additional bonus of 0.5pts. Note that this bonus requires texturing, reflection and refraction to be working correctly.

Additional bonuses may apply; please discus with me to be sure.

Your grade will be capped at 10.

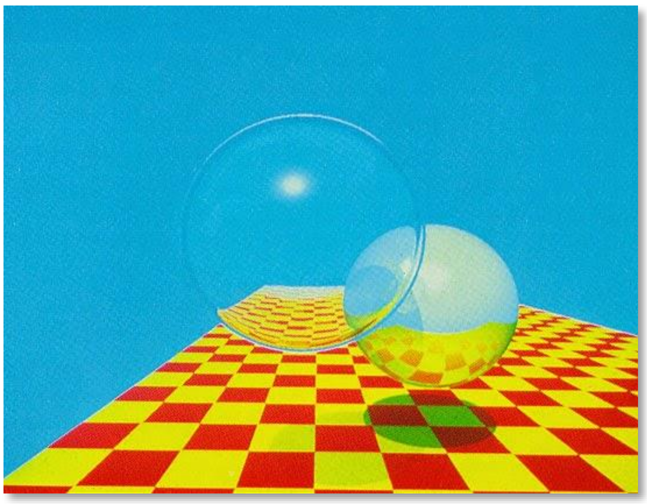
Please focus on features that are generic and applicable to any raytracing-based rendering algorithm. Note that at this stage, building a GPU renderer is not a good idea.

Academic Integrity

The work you hand in must be your own work. If you use materials from others (source code, libraries) please state this clearly in your report.

Purpose

We will use the result of this assignment in the second assignment. The code you produce should therefore be reusable.



May the Light be with you,

- Jacco.