CGRA354 Computer Graphics Programming T1/2024

Assignment #4: Ray tracing (20 points)
Assigned: Tuesday, 14th May
Due: 6th June 2024 11:59pm

The areas addressed in this assignment include the handling of vector algebra in order to compute intersections, lighting, reflections and shadows. Learning how lighting, sampling and shading models are implemented and work together in Ray tracing.

Codebase

You are provided with a cross-platform programming framework specifically designed for the Computer Graphics assignments. The framework performs the initialization of the pipeline allowing to focus on the assignment tasks. Comprehensive instructions for building the framework on different platforms can be found in the `README.md` file located in the source code root folder. The initial configuration of the project is displayed in `Initial_setup.png` file.

Turn in procedure

You should submit your work as a zip file using the ECS submission system. Please name your file as `<LastName><FirstName>-Assignment<X>.zip` where X is the assignment number. When your file is unzipped you should have:

1. The C++ code and programs you have written. You should use files in the form of the samples given, rather than producing files from scratch. This will help us follow your code.
2. Sample images created by your program. You can save these by taking a screenshot or use a screen capture tool.
3. A readme.md, txt, pdf or MS Word file that answers any questions posed, and that lists the input used to create the images you include. Alternatively you can provide a narrated screen-capture video showing the functionality of your program.
4. Any other information or supporting documentation to help us run and evaluate your submission.

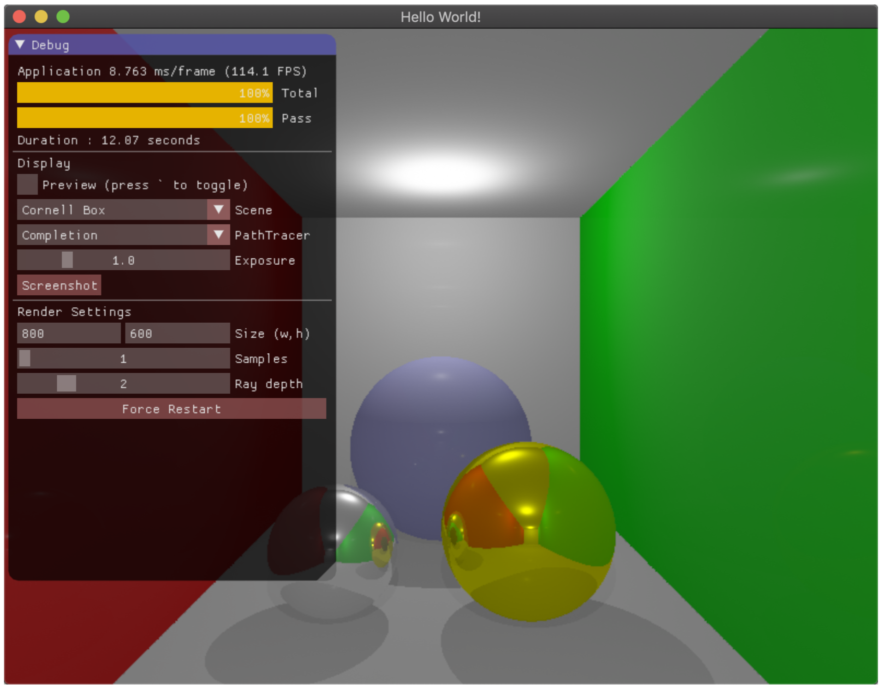
If your programs fail on the machines used for grading, you may be asked to bring in your system to demonstrate that the files you submitted functioned in the environment you worked in.

Grading rubrics

There are multiple questions in this assignment, with some question having multiple subparts corresponding to a particular tier (core, completion, challenge and writing).

Part 1: Ray tracing (20 points)

This assignment is to implement a basic ray tracer, including sphere, plane, triangle and disc primitives; multiple light sources; shadows; Lambertian, mirror and Blinn-Phong specular reflection.



Application 8.763 ms/frame (114.1 FPS)
100% Total
100% Pass
Duration : 12.07 seconds
Preview (press ` to toggle)
Cornell Box Scene
Completion PathTracer
1.0 Exposure
800 600 Size (w,h)
1 Samples
2 Ray depth

Figure 1: Cornell Box rendered within CGRA251 Framework.

Core (12 points)

Write code that will allow you to cast a ray into a scene defined by a collection of spheres lit by multiple light sources.

1. Complete the `generateRay` function of the `Camera` class, to create a ray for a given pixel.
2. Complete the `intersect` function of the `Sphere` class, to return the result of a ray intersection.
3. Complete the `DirectionalLight` and `PointLight` classes, to correctly test for occlusion and return correct irradiance values.
4. Complete the `CorePathTracer` class by implementing indirect diffuse (ambient) reflection, Lambertian diffuse reflection, and Blinn-Phong specular reflection.

Completion (5 points)

Completion has two parts:

- Demonstrate that you can handle more than one type of object by modifying and rendering the `Shape test` scene with a plane, disk, and triangle.
- Complete the `CompletionPathTracer` class by extending the implementation of `CorePathTracer` to handle reflections by recursively bouncing rays off objects in the scene (to a given depth).

Challenge (3 points)

Add to your ray tracer:

- Implement a textured material for texture mapping on objects.
- Complete the `ChallengePathTracer` class by extending the implementation of `CompletionPathTracer` to handle stochastic ray tracing. This means that, rather than calculating a Lambertian diffuse and Blinn-Phong specular reflection, a ray is shot in a randomly chosen direction to calculate part of the diffuse and specular reflection.

Note: You must use the OpenGL Mathematics (GLM) facilities on this assignment.

References

[1] Graphics at Cornell: History of the Cornell Box `http://www.graphics.cornell.edu/online/box/history.html`

[2] YouTube: Ray Tracing and other Rendering Methods, `https://www.youtube.com/watch?v=LAsnQoBUG4Q`

[3] Wiki: Ray tracing, `https://en.wikipedia.org/wiki/Ray_tracing_(graphics)`

[4] Scratchapixel: Introduction to Ray tracing, `https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-ray-tracing/how-does-it-work`