Coursework I: Raytracing

COMPGV3080 Team

October 18, 2016

We have shown you the framework for solving the coursework at `cg.cs.ucl.ac.uk/`. You should start by extending the example `Coursework1`. The programming language is OpenGL ES Shading Language (GLSL) `www.khronos.org/files/opengles_shading_language.pdf`. Do not write any answers in any other programming language, in paper, or pseudo code.

Remember to save your solution often enough to a `.js` file. In the end, hand in that file via Moodle.

The total points for this exercise is **100**.

The answers are due on **Friday, November 4th 2016, 23:59 UK time**.

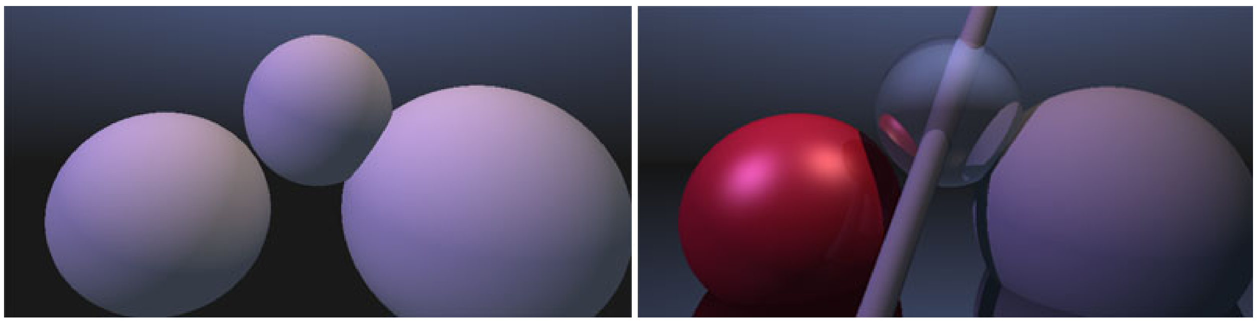

Figure 1: Left: *The scene from this coursework, rendered with the simple initial ray-tracer you are asked to extend.* Right: *A rendering with shadows, reflections, refractions and different materials.*

1 Shadows, reflections and refractions (65 points)

We have discussed how shadows, reflections and refractions work and how recursions can be unrolled into loops under some conditions. The framework contains the skeleton of such a traversal, but without the code for shadows, reflection and refraction.

First, add shadow tests to the shading (**10 points**), with comments (**5 points**) and explain what typical error can be made here (**5 points**).

Second, extend the `Material` class to hold all information you need for computing reflection and refraction (**15 points**).

Next, the code already contains the loop to perform the ray traversal iteratively, what remains to be added is code for computing the reflection (**10 points**) and refraction direction (**10 points**).

Finally, make the scene show different materials and explain why those parameters are adequate in some table: paper (**2 points**) metallic mirror ground plane (**2 points**) a glass sphere (**4 points**) and a red plastic sphere (**2 points**). It is important that settings are chosen such that the distinct visual features become visible in the scene.

2 New primitives: Plane and cylinder (35 points)

We have added definitions of planes (a normal $\mathbf{n}$ and a distance r to the origin) and cylinders (an orientation $\mathbf{o}$ and a radius r) to the scene. You are asked to add code to intersect them (**15 points**) and explain how they work (**10 points**). Pay attention on how `Sphere` is intersected. Stick to the same function signature for `Plane` and `Cylinder` that contain `HitInfo`, including normals and material (**10 points**).