Coursework I:
Ray-tracing

COMP3080/GV10 Team
Tobias Ritschel, Anthony Steed, David Swapp, Carlo Innamorati, Peter Hedman, Sebastian Kay, Ben Congdon

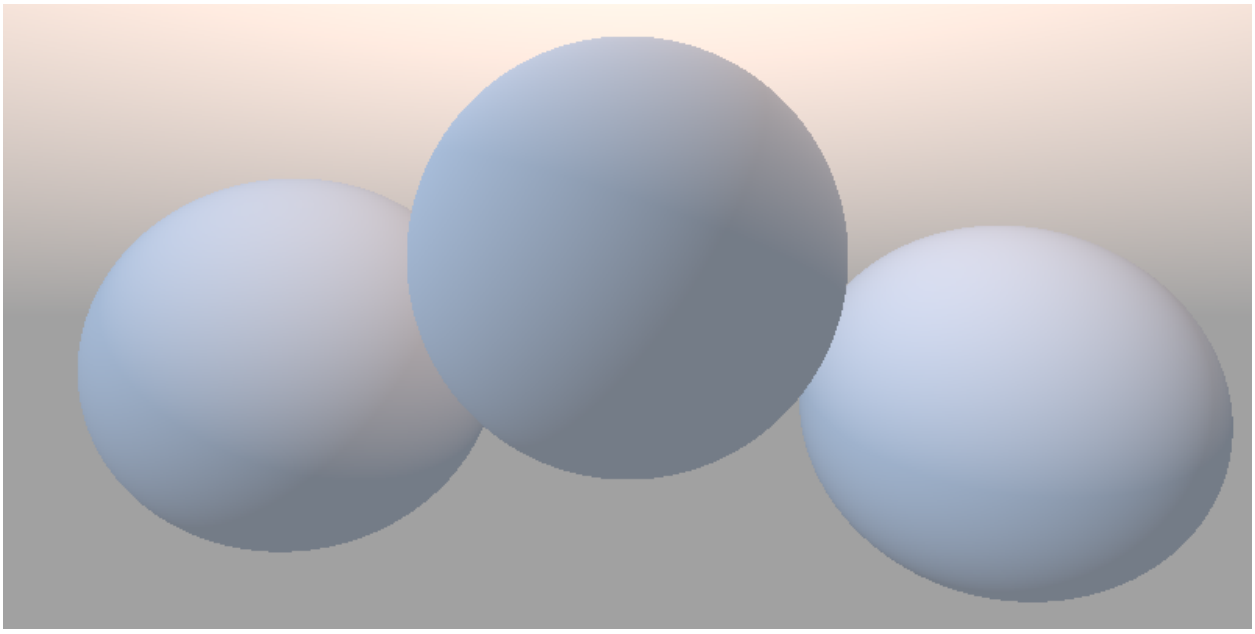
October 16, 2017

We have shown you the framework for solving the coursework at `cg.cs.ucl.ac.uk/`. You should start by extending the example `Coursework3`. 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.

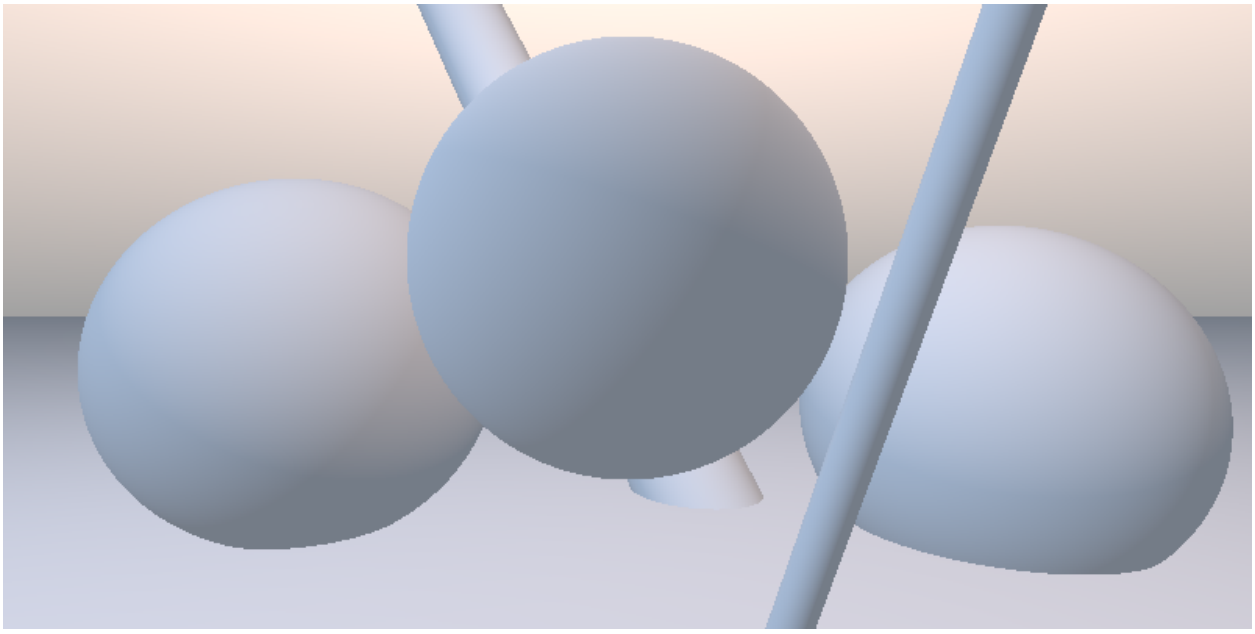
The total points for this exercise is **100**.

The answers are due on **Monday, November 6th 2017, 23:55 UK time**.



Above, the scene from this coursework, rendered with the simple initial ray-tracer you are asked to extend.

1 New primitives: Plane and cylinder (30 points)



We have added definitions of planes (a normal **n** and a distance r to the origin) and cylinders (an orientation **o** and a radius r) to the scene. You are asked to add code to intersect them (**10 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**).

2 Materials (6 points)

Make the scene show different materials and explain why those parameters are adequate in some table: paper

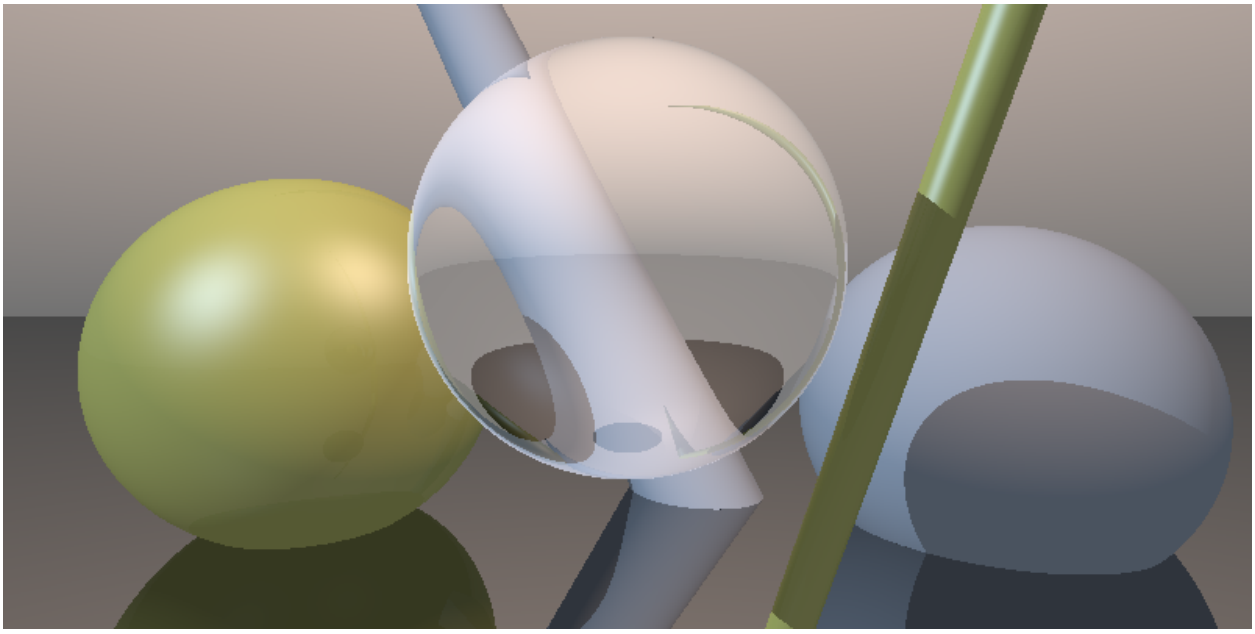
(**1 points**) metallic mirror ground plane (**2 points**) a glass sphere (**2 points**) and a yellow plastic sphere (**1 points**). It is important that settings are chosen such that the distinct visual features become visible in the scene.

3 Casting shadows (20 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**).

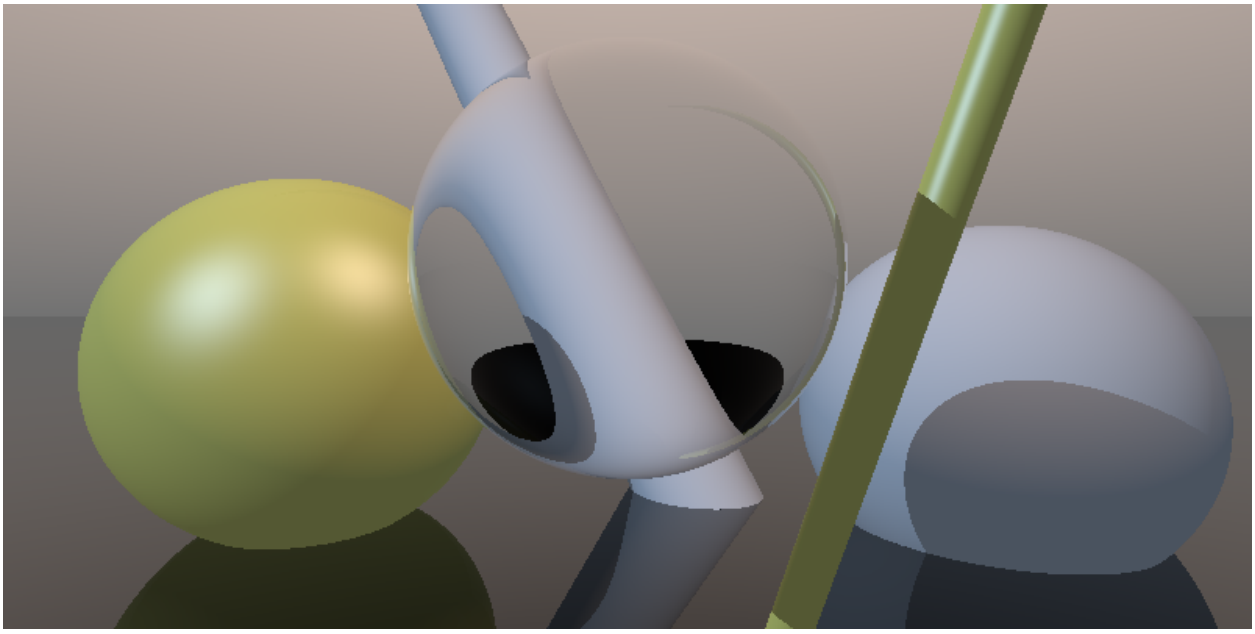
4 Adding reflections and refractions (34 points)



To add these, first extend the `Material` class to hold all information you need for computing reflection and refraction (**10 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 (**12 points**) and refraction direction (**12 points**).

5 Fresnel (10 points)



The image with reflection and refraction looks okay, but the glass likely does not look like glass as it shows reflection and refraction at equal strength all over the sphere. Such an image is shown above. Here, and in reality, reflection and refraction strength vary so that w_{reflect} and w_{refract} sums to one or less, so here we simply assume $w_{\text{reflect}} = 1 - w_{\text{refract}}$. This weighting depends on the view direction and the normal at the hit point. The reflection is typically strong on grazing angles close to the edge of the sphere, while the refraction is strongest in the centre. Do some research to find out how those weights could be computed and tell us what you used (**5 points**) and implement it (**5 points**) in the function `fresnel`. A simple implementation such as using a single dot product or the approximation by Schlick could be a good starting point.