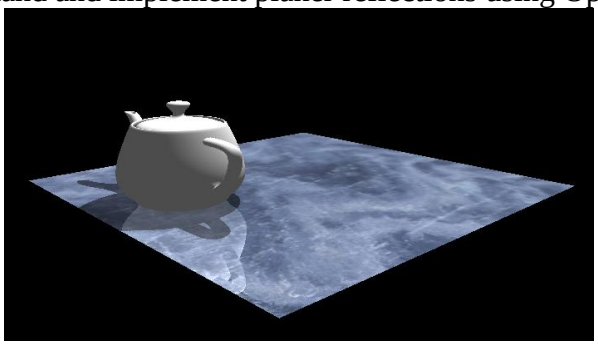
Exercise 9: Reflections

Reading	Angel: Chap. 7.8 – 7.9
Introduction	Like shadows, reflections are almost trivial to compute in ray tracing and more difficult when we use OpenGL. But again, if we simplify the situation and consider only planar reflectors, it becomes rather simple to draw the reflection. Imagine that we render an object in front of a reflecting planar surface. To draw the reflection, we can simply mirror all the vertices of the object in the plane, and render the mirrored object.
Purpose	To understand and implement planer reflections using OpenGL. 
Part 1 Introduction to stencil buffer	Run the program Exercise09-01. • Explain what the `setupStencil`-function does and how it affects the rendering taking place after the function is called.
Part 2 Setup	The rest of this exercise builds on top of the exercise from last time (Exercise08-09). Before we begin we need to disable the rendering of the plane. • Comment out the line that renders the plane in the `drawPlane()` function
Part 3 Reflect geometry	The plane we are reflecting about is incident on (0,0,0) and has normal (0,1,0). Now draw the teapot in such a way that it is also reflected in this plane. This is done by scaling using (1,-1,1). • In the `drawMirror()`-function draw the reflected teapot You should now see both the reflected and the normal teapot moving around.
Part 4 Reflect lighting	Notice that even though the geometry is reflected the light is not yet reflected. • Reflect the light when rendering the reflected teapot

Exercise 9: Reflections

Part 5 Blending	Here we will re-enable the rendering of the plane and blend the reflected teapot with the result. • Change the color of the reflected teapot to be an appropriate value between 0 and 1. This will make the teapot appear darker. • Enable the rendering of the plane again – notice that this will overdraw the reflected teapot. Solve this by enabling blending – choose the blending mode that adds source and destination value together. • After rendering the plane disable blending.
Part 6 Stencil buffer	To fix the problem where the reflected teapot is rendered outside the reflecting plane we need to only render the part of the reflected teapot "inside" the reflecting plane by using the stencil buffer in the `drawMirror` function: • Setup the stencil ◦ Enable stencil test ◦ Disable the depth testing and any writes to color buffer (using `glColorMask`) ◦ Set the stencil reference value to 1 when a pixel is rendered ◦ Render the plane object • Render with stencil ◦ Enable depth test and writing to color buffer ◦ Set stencil buffer to render when the stencil value is 1 ◦ Render the reflected teapot • Disable the stencil test
Part 7 Clipping plane	The reflection has one problem: The reflection plane always reflects the whole teapot – even the parts of the teapot which is below the surface. Fix this by introducing a clipping plane using the parametric form of a plane equation $c = (x,y,z,w)$, where (x,y,z) is normal of the plane and w is the distance to the plane from the origin. Every point p should be clipped when $p.x * c.x + p.y * c.y + p.z * c.z + c.w > 0$ Currently the c++ file contains a clipPlane variable, which is sent to the diffuse fragment shader. • Set the clip plane before rendering the mirror teapot (and set it back to zero after rendering the mirror teapot). • Implement clipping in the fragment shader using the `discard` command, by comparing the world coordinate with the clip plane uniform.

Exercise 9: Reflections

Part 8 Optional Use clipping plane to improve shadow	The shadow from the teapot is always cast to the surface – even when the object in under the surface. Use the clipping plane to clip away part of the teapot under the surface when the shadow-map is updated.