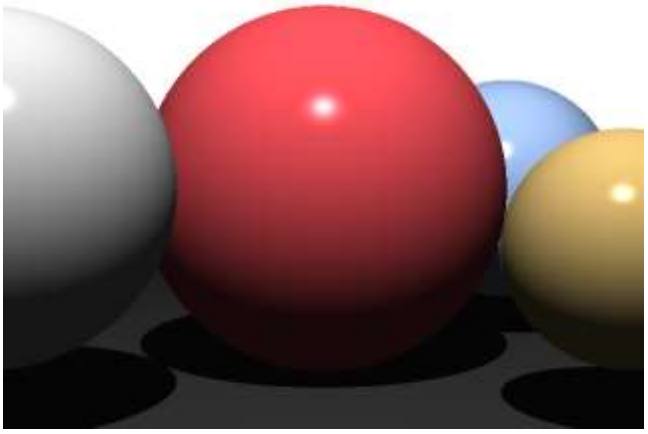
Advanced Graphics Programming

Workshop Three (Part A) – Ray Tracing (Hard Shadows)

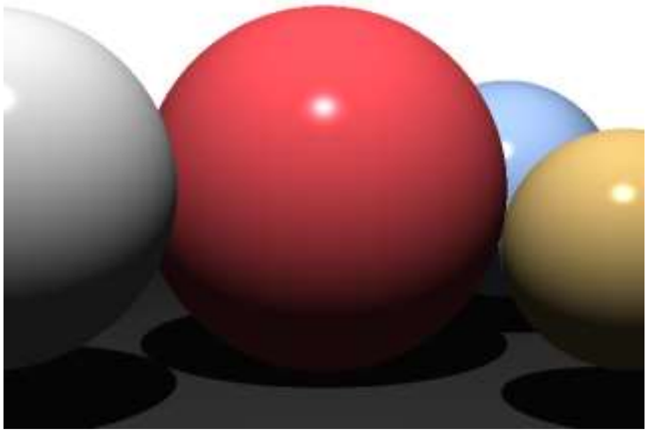
This workshop follows on from the Ray Tracing algorithm you developed in workshop 2. You need to have completed at least workshop 2 part A to add shadows. This workshop involves standard ray tracing to create hard shadows as shown below:



Step 1: Add hard shadows for spheres. To create hard shadows you will use the point light source that you created in workshop 2A. First, reuse the light ray (a.k.a. shadow ray) that you created for the diffuse shading with direction $\mathbf{l}$ and origin $\mathbf{s}$ which is a ray from the point of intersection $\mathbf{p}$ to your light source with position $\mathbf{s}$:

$$\mathbf{l} = \mathbf{s} - \mathbf{p}$$

Then use your existing ray-sphere intersect method (created in workshop 1B) to test if the light ray intersects with any spheres if it does then the pixel should be set to just the ambient light as it is in the shadow of a sphere, otherwise it is in the light so set the pixel to the colour based on the ambient + diffuse + specular light. Now when you run your code you should see shadows for each sphere (as shown below).



Step 2: If you use the point of intersection $\mathbf{p}$ as the origin of the light ray you may see a self-shadowing effect caused by a precision error (see examples below). This can be fixed by adding a small amount to the origin of the light ray in the direction of the normal e.g.

$$\mathbf{p}' = \mathbf{p} + \mathbf{n} * \varepsilon$$

Where $\varepsilon =$ 1e-4 and $\mathbf{n}$ is the normal at the intersection point $\mathbf{p}$ of your sphere with centre $\mathbf{c}$

$$\mathbf{n} = \mathbf{p} - \mathbf{c}$$

Additionally, you can avoid performing the intersect test between the current primitive and itself which is also more efficient.

Step 3: Add hard shadows for triangles and planes. First you need to add an additional check to your triangle-ray and plane-ray intersection functions to make them more robust. Add a test to see if the intersection is behind the ray origin (e.g. t <= 0) in this case return false.

A single triangle is not sufficient to cast a shadow so you need to add a ground plane. The ground plane in the example below has position (0, -1, 0), normal (0, 1, 0) and diffuse colour (0.8, 0.8, 0.8). The triangle has vertices: (0, 1, -2), (-1.9, -1, -2), (1.6, -0.5, -2), normals (0.0, 0.6, 1.0), (-0.4,-0.4,1.0), vec3(0.4, -0.4, 1.0), diffuse colour (0.5,0.5,0.0), specular colour (0.7, 0.7, 0.7) and shininess (100). The light has a position (1,3,1) and intensity (1,1,1).

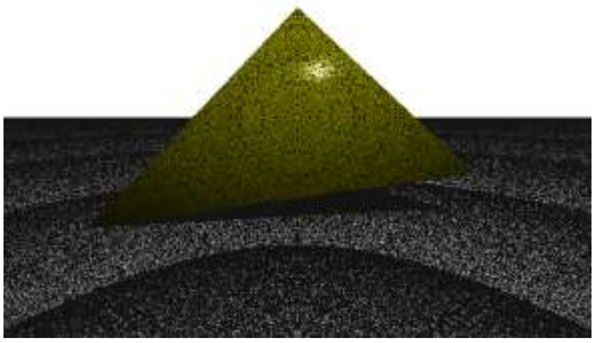


Figure 1 Triangle and plane with hard shadow and shadow acne

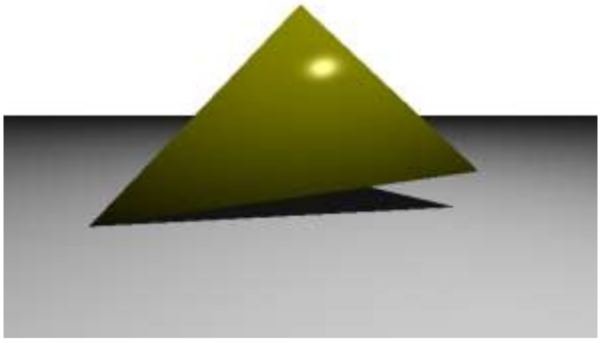


Figure 2 Triangle and plane with hard shadow

Now the shadow code should work with any model that you have previously implemented. See examples below.

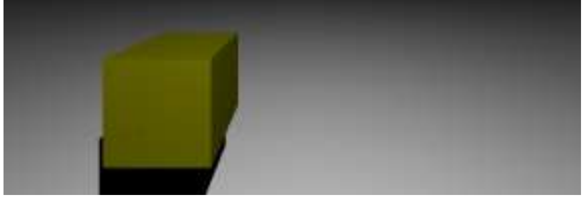


Figure 3 Cube with hard shadow

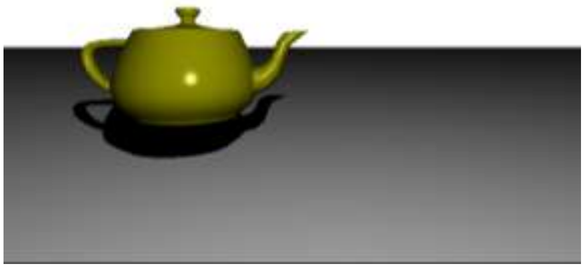


Figure 4 Teapot with hard shadow