ECM3423: Computer Graphics
Worksheet 9 - Reflections

The aims of this workshop are as follows:

1. Render an environment map around the bunny/sphere from all six angles (top, bottom, left, right, front and back)
2. Write a new shader program based on the Phong shader (you can use the flat shader if your Phong does not work), that samples from the environment map for reflection.

Before starting you should study carefully:

- The lecture slides on textures and environment mapping
- The additional lecture notes on textures provided on the VLE.
- The provided code for WS8 that implements textures.

Part 1: Render the cube map

In this part, you want to render the scene in six different directions, by using framebuffers to generate all six faces of the cube map texture. The framebuffers are described in the lecture notes:
We can create a framebuffer (FBO) object as other OpenGL object.

```
fbo = glGenFramebuffers(1)
```

Then, as with other objects, we need to bind the FBO before operating on it:

```
glBindFramebuffer(GL_FRAMEBUFFER, fbo)
```

Once it is done, any rendering command will operate on the FBO and not on the screen. In order to recover the product of such rendering, we need to attach the output of the rendering pipeline to some texture. The rendered image can be saved by binding a texture to the `GL_COLOR_ATTACHMENT0`:

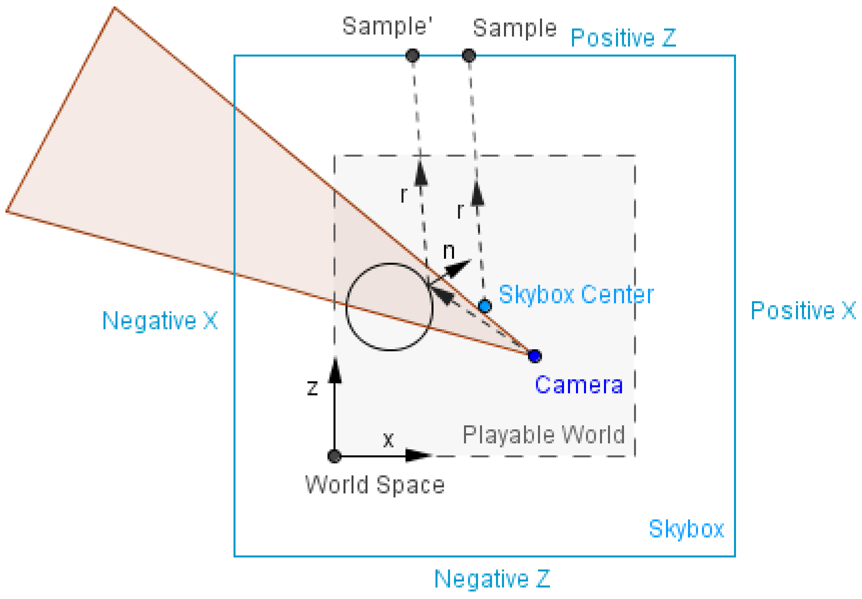
```
glFramebufferTexture2D(GL_FRAMEBUFFER, GL_COLOR_ATTACHMENT0, textarget, texID, 0)
```

For each of the cube's faces here are the required steps:

1. Bind the corresponding frame buffer object (we will use one framebuffer per face of the cube).
2. Setup the view matrix properly to ensure that the camera is located at the cube's centre and facing the desired face (note that the center of the object is at the origin of the world coordinate system).
3. Clear the frame buffer (if required), both the depth and colour buffers can be cleared with:
   `glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);`
4. Render as usual.

Part 2: Sampling the cube map for reflections

In this part, the task is to implement the shader code to render reflective surfaces using the cube map generated in the first part. Start by copying your existing shader code, for example from WS8. The goal is to implement the correct operations to sample from the cube map texture. Please see more details in the relevant lecture slides.

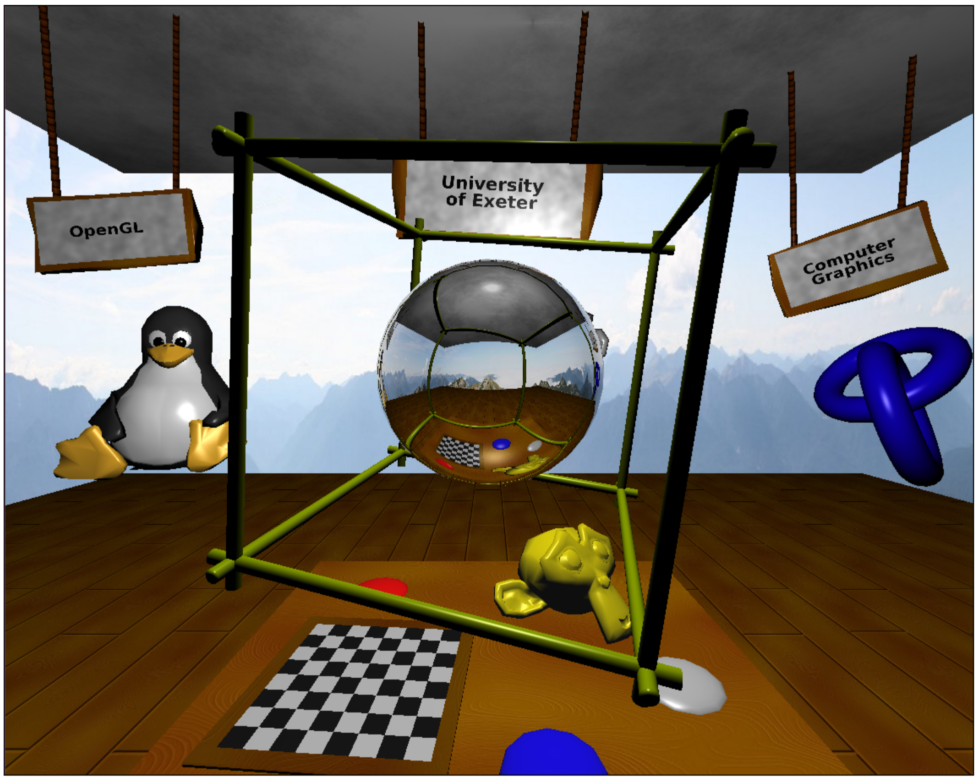


Notes:

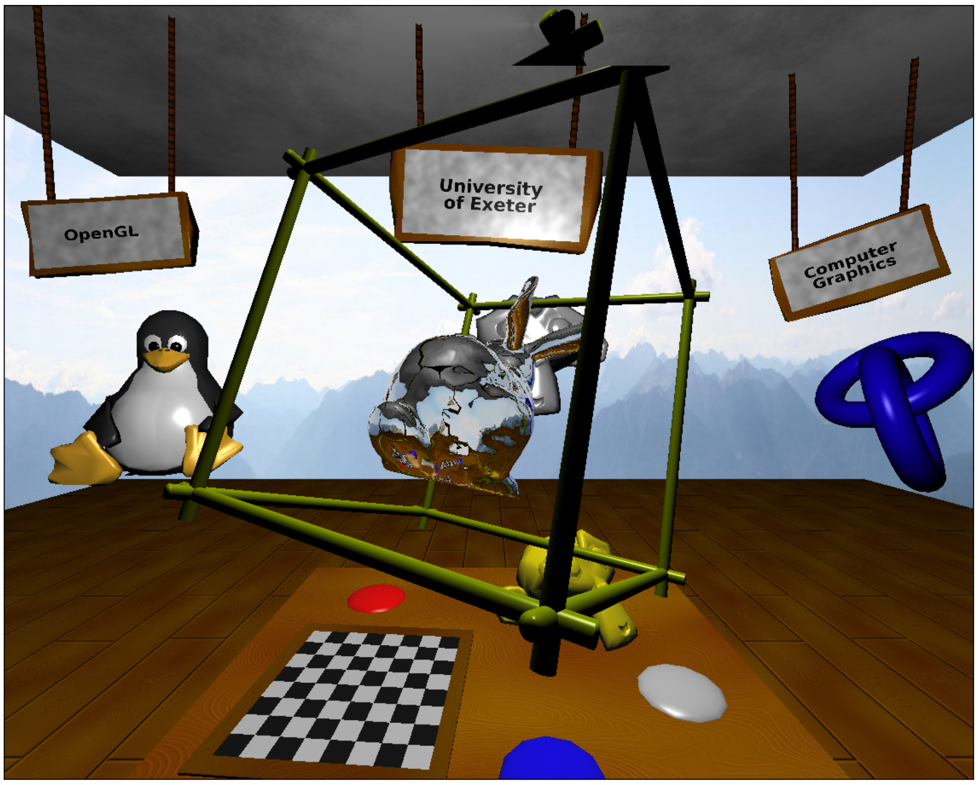
- You can use the GLSL `reflect()` function to calculate the reflection of a vector with respect to a surface normal.
- Using a `samplerCube` object you can sample directly from the cube map texture using the reflected vector. Once implemented, you can check that the reflections look correct both on the sphere and on the bunny mesh.

The rendering results are similar to the following screenshots :

Reflections on the sphere:



Reflections on the bunny:


University
of Exeter
OpenGL
Computer
Graphics