CSE 333/533: Computer Graphics

Lab 06: Environment Mapping

November 23, 2022

1 Introduction

In this lab, we will look at how to use cubemap and use it to map the environment. A cubemap is a texture that contains 6 individual 2D textures that each form one side of a cube: a textured cube. A cube map essentially maps a hemisphere above the surface, but in many cases, maps the entire 360 degree scene around a point. Two popular techniques that make use of cube maps are **sky boxes**, which provide the illusion of a 3d background behind the scenery, and **environment mapping**, which can make a very-shiny surface (think lakes, chrome, or car paint) appear to reflect the environment scenery.

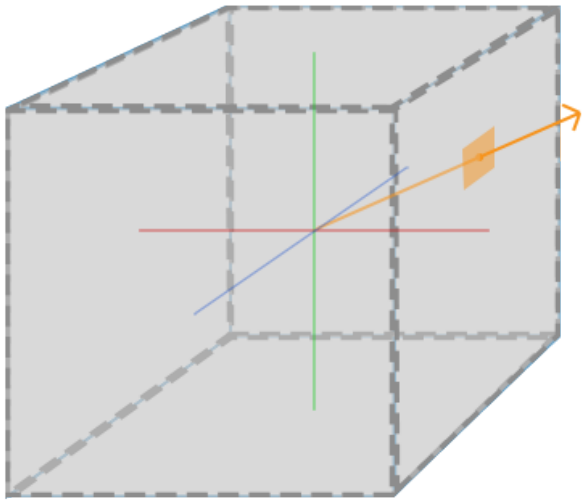

Figure 1: Cubemaps. [1].

1.1 Skybox

- A cubemap is a texture like any other texture, so to create one, we generate a texture and bind it to the proper texture target before we do any further texture operations. This time binding it to `GL_TEXTURE_CUBE_MAP`. For reference, check line no. 142 in skybox.cpp.
- Because a cubemap contains 6 textures, one for each face, we have to either call `glTexImage2D` six times with their parameters set(for ref, check line no. 145). This time however, we have to set the texture target parameter to match a specific face of the cubemap, telling OpenGL which side of the cubemap we're creating a texture for. This means we have to call glTexImage2D once for each face of the cubemap.
- Then you have to set the filers for the texture format.

1.2 Environment Mapping

An environment map behaves like a mirror to translucent surfaces to either reflect or refract the skybox. For this, we have to work out a direction vector from the viewpoint (camera position) to the surface. We then reflect the direction of the surface based on the surface normal. We use the reflected direction vector to sample the cube map. GLSL has a built-in `reflect()` function which takes an input vector and a normal, and produces the output vector.

Similarly, translucent objects should **refract** light. This means that it doesn't go straight through; the ray bends when it hits the surface and changes material. Looking into a swimming pool, for example. This works much the same way as reflection, except that we perturb the eye-to-surface vector about the surface normal, instead of reflecting it. The actual change in direction is governed by Snell's Law. GLSL also has a built-in `refract()` function with a vector-based version of Snell's Law.

Figure 2: Skybox Texture. [1].

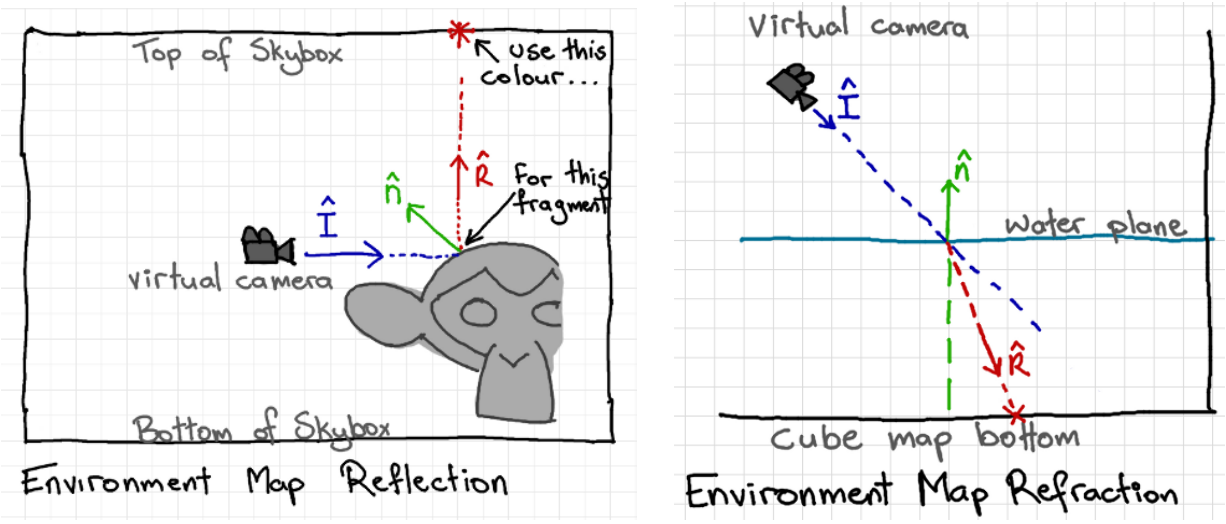

Figure 3: Environment Mapping [2].

1.3 Lab Code instructions

The current code just renders the skybox. If you set `ENABLE_REFLECTIVE_SPHERE` to 1 then you will see the reflective sphere.

2 Deliverables

The task for the lab is to convert the reflective sphere to a refractive sphere. For this you just have to add the refraction part in the fragment shader. To have a realistic glass sphere which both reflects and refracts you have to use Schlicks approximation to get the final color. Submit the screenshot of outputs (Only refraction and reflection refraction together) along with code for evaluation in a zip file. Name the zip file as Lab07_<name_roll no>.zip.

(a) Reflective Sphere (b) Refractive Sphere

Figure 4: Outputs

References

[1] LearnOpenGL - Cubemaps. site: https://learnopengl.com/Advanced-OpenGL/Cubemaps

[2] Cube Maps: Sky Boxes and Environment Mapping - https://antongerdelan.net/opengl/cubemaps.html