CSE 5542, Lab Assignment 4
Reflection and Refraction, due on 04/01/2012 11:59PM

1 Goals

We will learn how to produce simple reflection and refraction effects using GLSL.

2 Requirements

You will find a sample example called env_texture on the course website. This example draws two objects: a sphere and a bunny. Each object is associated with a GLSL shader program. An environment texture from Paul Debevec’s website\(^1\) has already been uploaded into the two shader programs. Based on this example, your responsibility is to modify the shaders so that they can achieve reflection and refraction effects.

2.1 Environment Mapping (2 Points)

Assuming that the viewer is sufficiently far away from the environment, the environment texture provides the texture color for each viewing direction \((x, y, z)\) in the world space. There are many mapping strategies to achieve this. In particular, Paul Debevec’s light probe environment texture uses the following equations:

\[
\left\{ \begin{array}{l}
u = \frac{rx+1}{2}; \\v = \frac{ry+1}{2}, \\
r = \frac{\cos^{-1}(z)}{\pi \sqrt{x^2 + y^2}}
\end{array} \right.
\]

in which \((u,v)\) are the texture coordinates converted from \((x,y,z)\).

Please implement this in the Environment function provided in the two fragment shaders. After this, you should be able to see the whole environment rendered on the sphere. Drag the cursor to look around.

2.2 Reflection (2 Points)

Although we modified Environment in the bunny fragment shader at this time, the bunny still looks unchanged since we haven’t implemented reflections or refractions yet. Let \(V\) be the original viewing direction from the fragment point to the camera and \(N\) be the surface normal as shown in Figure 1a, the reflected viewing direction \(R\) can be calculated as:

\[
R = 2(V \cdot N)N - V.
\]

Both \(V\) and \(N\) should be normalized.

Please implement Equation 2 in the Reflection function and combine it with Environment to achieve the reflection effect in the bunny fragment shader, as Figure 1b and 1c show.

Note that the sample example provides \(V\) and \(N\) in the eye space, while Environment takes the direction in the world space. To solve this problem, you can either convert \(V\) and \(N\) before you calculate \(R\), or you can simply convert \(R\) after it gets calculated. You can use the gl_ModelViewMatrixInverse matrix to convert a vector from the eye space to the world space. Keep in mind that the vectors represent directions here and their homogeneous coordinates are different from those representing positions. How?

\(^1\)http://www.paulDebevec.com/Probes.
2.3 Refraction (2 Points)

Let $V$ be the original viewing direction and $N$ be the surface normal as Figure 1a shows, the refracted viewing direction $T$ can be calculated as:

$$T = -\sqrt{1 - S \cdot SN - S},$$

in which,

$$S = \frac{1}{n} (V - (V \cdot N)N).$$

Here $n$ is the refractive index and you can assume $n = 1.5$ as glass. Implement this in the Refraction function and combine it with Environment to achieve the refraction effect.

2.3.1 The Fresnel Model (3 points)

Both Reflection and refraction happen on the surface of a transparent object. Once we calculate their directions and their corresponding colors from the environment texture, we need to combine these two colors together in a physically plausible way, so that the combined effect looks photorealistic. Here we use the Fresnel model. Let $n$ be the refractive index, $\theta_i$ be the incoming angle and $\theta_t$ be the refractive angle as Figure 1a shows. The Fresnel model defines the reflective coefficient $c_r$ as:

$$c_r = \frac{1}{2} \left( \frac{\cos \theta_i - n \cos \theta_t}{\cos \theta_i + n \cos \theta_t} \right)^2 + \frac{n \cos \theta_i - \cos \theta_t}{n \cos \theta_i + \cos \theta_t} \right)^2,$$

and the final color $C$ can then be blended as:

$$C = c_r C_r + (1 - c_r) C_t,$$

in which $C_r$ is the reflection color and $C_t$ is the refraction color. Implement Equation 5 in the Fresnel function and combine Equation 6 with other steps in the main function to achieve both effects, as shown in Figure 1d.

2.3.2 HDR vs. Non-HDR (1 point)

The environment texture used by the sample code is known as the High-Dynamic-Range image, which allows each pixel to have a larger value range by using float points instead of unsigned bytes.
In fact, even if the texture image is HDR, the sample program can still load it as a regular (non-HDR) texture into GPU by using the `Load_Texture` function.

Please define such a regular texture in the shaders and use it for rendering. Use the `h` key to toggle on and off between HDR and non-HDR textures. You will see some difference.

3 **Bonus Credit (2 points)**

The bunny fragment shader considers the refraction event only once. In the real world, however, the refraction event happens at least twice: once entering the object and once leaving the object. To handle the second refraction, we need to calculate the intersection between the refracted ray and the object, which is not easy when the object is an arbitrary mesh.

But when the object is a sphere, we can do this analytically. You can create a sphere object using the `Create_Sphere` function in the MESH file. Can you come up with a method to calculate the second refraction? Please describe it in an attached document file (1 point). Can you implement it as a new shader as well (1 point)? Use the `1` key to toggle between single refraction (bunny) and double refraction (sphere).

4 **Submission Guideline**

Please submit the source code (without the executable or object files) by email to our grader: Xiaoyin Ge (gex AT cse.ohio-state.edu). Use “CSE 5542 Lab 4” as the email subject.