

Spring 2008
22C:196 Advanced OpenGL Rendering
Assignment 4

Due: Tuesday, April 22nd at 11:59pm

Goal: Implement basic reflection using OpenGL. Design a room with other interesting rendering effects. You *may* work in groups of two. You *may* use the OpenGL framework I will post online.

Problem 1 (15 points): Write an OpenGL program that reflects a scene around a simple, planar mirror. The scene should be an enclosed room, with one mirrored wall and the remaining surfaces textured. Include at least one complex object (e.g., AI, a teapot, dragon, etc) that can be moved around the scene via mouse or keyboard. Note:

- You should either restrict objects so they must stay in the room or make sure you add a clipping plane at the mirror's surface so objects outside the room are not reflected inside!
- The room should have (at least) one light that affects reflected geometry correctly (i.e., lights the reflected geometry correctly). The user should be able to move the light around the room.
- You may use either a stencil-buffer or render-to-texture approach to this problem.

Problem 2 (35 points): The idea of this problem is to extend your mirrored-room scene into an interesting demo you can show off to people and put on your webpage. Your result should be something like the mirrored-room demo I showed in class, but you may choose which of the following features to implement as long as they total at least 35 points (for individuals) or 55 points (for groups). Features must interact seamlessly (i.e., bumpmapping should be reflected in the mirror) and any that do not interact seamlessly should be clearly documented.

(15 points): Add an additional mirror, and allow recursive reflections between the mirrors. Add a menu or key command that allows the user to increase or decrease the maximum recursion level. Make sure the program starts with a view that can see recursive reflections (perhaps by having a non-rectangular room).

(5 points): Make (at least) one of your mirrors have an interesting shape (e.g., a teapot, bunny, or dragon shaped mirror).

(10 points): Make your floor reflective, but only partially reflective (so you can see the floor texture in addition to the reflection). The reflectivity should vary based upon the angle between the unit-length surface normal and the normalized viewing vector: $\text{reflectivity} = 1 - (1 - \vec{N} \cdot \vec{V})^5$. Use a GLSL program to control this blending. *This is a Fresnel effect, and the equation is Schlick's approximation. Note the value should be in the range [0..1], which means $\vec{N} \cdot \vec{V} \in [0..1]$.*

(20 points): Implement refraction for the complex 3D model (e.g., the bunny, dragon, or triceratops models). In order to see the room refracted through the object, you should intersect the refracted ray with each wall in the room (in your fragment program). If you have other non-planar objects in the room, you may remove them for this part of the assignment. Any mirrors need not reflect the refractive object. Add a menu or key command to toggle the object between refractive and opaque. Use an index of refraction between 1.2 and 1.4.

(10 points): If you add a refractive object, allow the mirror(s) to reflect it.

(15 points): Add shadows to your scene. You may use any shadowing technique you want, but shadows should appear on (at least) all walls, the floor, and the ceiling, and must be reflected (and refracted) by any mirrors (or refractive objects).

(5 points): Add a toggle that applies an image processing effect (from the previous assignment) to the mirrored reflection. (This works best if you use the render-to-texture method for reflections.) For instance, by applying a Gaussian blur, the reflection could show a blurry version of the scene. Or the mirror could only reflect a grayscale image of the scene.

(5 points): Outline your complex object using thick white silhouette lines (like those shown as examples for the last assignment). This means the object should be drawn with OpenGL shading *and* silhouette edges. When seen reflected by the mirror, the silhouettes must be in an appropriate location (i.e., they must be silhouettes as seen from the reflected/virtual eyepoint).

(10 points): Implement bump mapping. You may apply the bumpmapping to any surface in your scene, but the bumps must be clearly visible (i.e., different than the non-bumpmapped version). Add a toggle to enable/disable bumpmapping. Good and easy objects to bumpmap are spheres (e.g., an orange or a globe).

(20 points): Implement displacement mapping. You may use any of the displacement mapping techniques described in class. Make sure the displacement map is applied to an interesting surface. One nice place for displacement mapping is on a brick wall. Make sure you add a toggle to enable/disable displacement mapping, and make sure the controls easily allow the user to move close to the displacement mapped surface.

(Optional) Problem 3 (50 points): Instead of doing Problems 1 and 2, you may implement explosion mapping entirely on the graphics card. If you choose to do this, please come talk to me first. *Groups must satisfy a higher bar if you choose this option, so it is doubly important to talk to me first.*