

**Spring 2006**  
**22C:196 Advanced OpenGL Rendering**  
**Assignment 5**

**Due: Thursday, May 11th at 11:59pm**

<b>Goal:</b> Spend more time improving and debugging your reflective room demo.
---

**Problem 1 (25 points):** Fix up your Homework 4 so it is easy to use and looks good. You must invest some time in software engineering. Some of these points will be discretionary on my part. Remember, this assignment is replacing your final, so I expect this to be a nice program that you can show off to friends (and I can show off saying, “look at the cool things my students made”). This means you must have in your program:

- Mirrors that are not 100% reflective, so the viewer can see the mirror. The reflectivity should be controllable via a menu or keyboard interface.
- All static textures should use mipmaps to reduce aliasing artifacts.
- All geometry with more than 50 polygons should use display lists or vertex arrays to keep rendering times as short as possible.
- Make sure to use multi-texturing (i.e., `glActiveTexture()` and `glMultiTexCoord*()`) so that you can combine multiple textures. This is particularly important if you are using the framebuffer object based reflections and also using shadow maps (or any other textures in your scene). Also make sure you only use automatic texture generation for those texture units that need it. I don't want to see wall textures that change as I move around the room.
- Allow the user to select the complex object and move it around the room with the mouse (i.e., using OpenGL selection). When you do this, please use only the current mouse coordinate (not the difference between this mouse coordinate and the last mouse coordinate) to position the object on screen. Also remember mouse coordinates start from (0,0) in the upper left (not the lower left as with OpenGL). If you're not sure how to do this efficiently, please come talk to me. Bound the movement so it cannot completely leave the room.
- Add some dynamic objects to your scene... but make sure it makes sense for the idea of a “room.” For instance, a moving ceiling fan would be appropriate, but a teapot orbiting the light source would not be.
- Add controls for moving the eye point around the scene. I would prefer mouse movement.
- Add controls for rotating the viewing direction (in place – not about the look-at point).
- Textured walls *must* be lit by an OpenGL light. This can either be the fixed function lighting, or some lighting model you implement in your shader. Feel free to borrow Cg code from the web to do per-vertex or per-pixel Phong shading.
- Please use `glEnable(GL_NORMALIZE)` if you render lit objects using OpenGL's Phong lighting. (Otherwise scaling changes the intensity of the light, which is why your object might appear very dimly lit.)
- Please fix all features you attempted in Homework 4 if they did not work correctly. For instance, shadows should not be 3D objects, and I would appreciate if they did not get cast on the mirror. You can implement this using a stencil (or by drawing the mirror on top of any existing shadow).
- Use textures that are appropriate for their function. For instance, a checkerboard or wood for the floor. Some relatively unobtrusive texture for the wall and ceiling. You may include your all-time favorite image in your program in alternate ways, as described below.
- For the one person who correctly implemented the approximate Fresnel reflection, I got the formula wrong. Please use  $\text{reflectivity} = 1 - (1 - \vec{N} \cdot \vec{V})^5$ , so that grazing angles are more reflective than when looking straight down.

**Problem 2 (25 points):** Extend your program by implementing additional features. You may implement any feature from Homework 4 (Problem 2) that you *did not* turn in previously (and that is not a requirement above in Problem 1) You may also do the following:

- (10 points):** Add additional complex objects to the scene to make the room more realistic. This will involve some searching for objects online. For instance, you could add some couches, chairs, a table, etc. I have readers for .obj, .ifs, and .m models. You can borrow code from elsewhere to read in other types of models. Please use built in material types, appropriate textures, or at least a material that roughly represents the real object's appearance. Add at least one framed "photo" or piece of art hanging on the wall, to add ambiance.
- (10 points):** Render (at least one of) the complex object(s) in your scene using the Gooch NPR shading model. Add a toggle to swap between this and standard lighting.
- (20 points):** Implement procedural texturing and add a toggle to switch between standard lighting and a marble texture on your complex model. You might consider replacing your floor with a (differently colored) procedurally generated marble. (This would be really cool with a partially reflective floor.)
- (20 points):** Implement real-time hatching, and apply to one of your complex objects. If you decide to implement this, you should come talk to me.
- (10 points):** If you implement Cg shaders for your mirrors, modify that shader to do a spatially varying filter based on distance to the reflected object. For instance, blur the reflection differing amounts based on the distance to the reflected object. If you do this, you should come talk to me.
- (15 points):** Add a framed piece of "art" to the wall, as in the first option mentioned above. Instead of a static photo, this will be dynamic. Implement the game-of-life and display the results inside this frame on one of your walls. This, obviously, will involve a render-to-texture and will require some starting image seed you will need to design. Please make sure your updates do not occur so rapidly I cannot follow the progress. This means you should only update the image two to five times per second. This dynamic image should be reflected in mirrors. The rules for which cells live and die are given here (or do your own Google search): <http://www.bitstorm.org/gameoflife/>
- (25 points):** Implement a GPU-based volume rendering, and use a volume rendered model in place of your complex object. (If this is slow, add a toggle back to a simple OpenGL-lit 3D model). This is worth all 25 points, because I'm unsure if I'll cover this sufficiently to implement it without additional background reading on your part.