Additional Techniques that have been implemented:

CSE 5542 Lab 5

Submitted by: Soumya Dutta

1. Model a see through/Transparent object
2. Model a see through glossy object
3. Implement Lens effect

Click here for Images generated: http://www.cse.ohio-state.edu/~duttas/Images_5542/RT_rendering.html

Overview:

Basic principal for the above three techniques is somewhat similar. Implementing these techniques along with shadow map requires three pass rendering. In the first pass render the depth from light's point of view and store in a Frame buffer object. In second pass, render the scene again without transparent objects and store the scene in another FBO. In the final rendering pass, render shadow using depth buffer and shadow map technique discussed in class.

Now, for adding transparency, first calculate normalized screen coordinate. Next, look up the FBO of second pass and get the color. Then mix that with objects diffuse color. In that way we are assigning screen's color to the object if it was not there. This will make the object transparent as if we are seeing through it. This is for normal see through/ a complete or semitransparent object. For Complete transparencies just use FBO's color and do not add any diffuse color.

For modeling a glossy see through object, we have to apply a little trick. While shading the transparent object, if we can somehow jitter (very little) the texture coordinates for FBO lookup, and then the result will be a glossy object.

For modeling lens effect, again while shading transparent object, do not look up the exact same screen texture coordinate. For a texture coordinate (x,y) lookup (x,y+dy) where dy is a very small quantity. What this will do is, it will look up a lower row's value from texture and the effect will be like a lens.

For glossy texture I am using a function which generates random number. I add that with the texture before looking up. Code for shader is given below. I will add comments so that the idea is clear.

1. See through Shader:

```
coord = (gl_FragCoord.xy) / vec2(800,800); //assuming 800X800 is display screen's
resolution.
```

Do texture look up like:

```
vec4 reflectionPlaneColor = texture(reflectionMap,coord); // calculate color for
texture
```

Then mix the color like:

```
gl_FragColor += Iamb + mix(reflectionPlaneColor,Idiff,0.5) + Ispec;
```

2. See through Glossy object:

Function that generates random number:

```
//Generate random number
float rand(vec2 co) // Pass texture coordinate as parametr

{
    return fract(cos(dot(co.xy ,vec2(12.9898,78.233))) * 43758.5453);
}
```

Next do something like:

```
coord = (gl_FragCoord.xy) / vec2(800,800) + rand(texCoordinate)/80; // Here 80 determines amount of jitter. If you increase this value, object will become more glossy/Hazy.

   vec4 reflectionPlaneColor = texture(reflectionMap,coord); // Look up using jittered texture coord
```

Then mix the color like:

```
    gl_FragColor += Iamb + mix(reflectionPlaneColor,Idiff,0.5) + Ispec;
```

3. Lens effect:

```
coord = (gl_FragCoord.xy) / vec2(800,800); //assuming 800X800 is display screen's resolution.

coord.y = coord.y-0.09; // Here adding or subtracting 0.09 will make the object like a lens.
```

Do texture look up like:

```
  vec4 reflectionPlaneColor = texture(reflectionMap,coord); // calculate color for texture
```

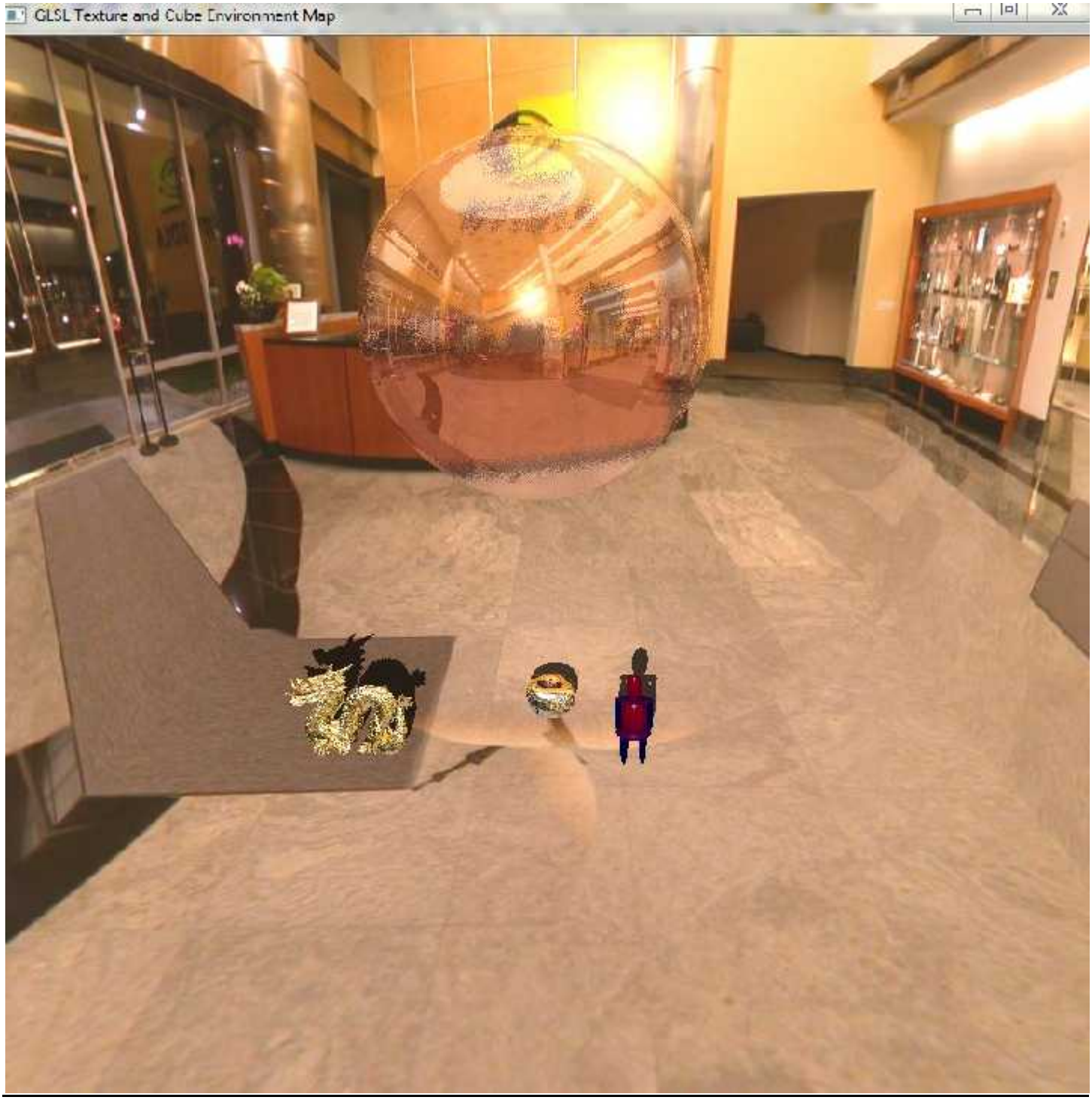
Then mix the color like:

```
    gl_FragColor += Iamb + mix(reflectionPlaneColor,Idiff,0.5) + Ispec;
```

Results of each of the technique:

1. **Normal See through object:**

2. **Glossy See through object:**



3. **Lens effect:**

4. **Bumpy lens:**

Reference: I looked over web and various game development forums for resources. I have learned the idea of rendering transparent object from:

1. http://www.gamedev.net/page/404.html
2. http://www.opengl.org/discussion_boards/forum.php
3. http://gamedev.stackexchange.com/questions/3223/game-development-blogs