CMSC 23700 **Introduction to Computer Graphics** **Project 3**
Winter 2008 **February 7**

A simple OpenGL viewer

Due: Thursday, February 21 at 1pm

1 Summary

This project is the third part of a three-part project. In this part, you will add code to support real-time shadows using the *Shadow Map* technique discussed in class. In addition to programmable shaders, this project will also involve using OpenGL's frame-buffer-object (FBO) extension and depth textures.

2 Description

Shadows are one of the most important visual cues for understanding the relationship of objects in a 3D scene. As discussed in class, there are a number of techniques that can be used to render shadows using OpenGL. For this project, we will use shadow maps.

The basic idea is that for each frame, you will the scene render two times. The first pass will render the scene from the light's point of view, generating a depth texture that will be used as a shadow map in the second pass.[1] The first pass can be done with the fixed pipeline (*i.e.*, without shaders). The second pass will require both a vertex shader and fragment shader. The vertex shader will extend the shader from Project 2 by also computing texture coordinates for the shadow map(s). The fragment shader will combine per-pixel lighting with the shadow-map information to compute the fragment color.

2.1 Lights

You will need to support shadows from one or two lights depending on the current view state (see below). One light is a directional light and the other is a spotlight hanging above the box. To allow light into the box, we have removed the top wall.

For each light l, you should compute the ambient (A_l) and diffuse (D_l) intensity as in Project 2 (these will be color vectors). You should also compute a value S_l, which is 0.5 when the pixel is in shadow and 1.0 otherwise. Then, assuming that the input color is C_{in}, A is the global ambient light level, and $\mathcal{L}$ is the set of enabled lights, the resulting color should be

$$C_{\text{out}} = C_{\text{in}} \, \mathbf{max}\left(A, \, \mathbf{clamp}\left(\sum_{l \in \mathcal{L}} S_l(A_l + D_l)\right)\right)$$

[1] If there are two lights enabled, then you will have to generate two shadow maps.

Note that for the directional light, $A_l + D_l$ will always be greater than zero, but that for the spotlight $A_l + D_l = \mathbf{0}$, when the fragment is outside the cone of light.

2.2 User interface

The user interface will be somewhat different from Project 2. The sample code supports the following keyboard commands:

`l`	toggle the lighting mode between directional, spotlight, and both lights.
`+`	add a ball to the simulation
`-`	remove a ball from the simulation
`s`	enable/disable shadows.
`q`	quit the viewer

2.3 Support code

The sample code will include support for using FBOs (see `fbo.h`) and support for matrix operations (see `matrix.h`).

2.4 Hints

Note that the whole scene can be bounded by a sphere at the origin that has radius $\sqrt{3}$. You can use this sphere to help determine the far plane when setting up the light's projection matrix.

For the directional light, you will need to use an orthographic projection.

Since the lights do not move, you can compute their model-view and projection matrices as startup time. You can also precompute the texture matrix needed to map eye-space vertices to the light's clipping space.

You may find it useful to render the shadow map to the screen as a debugging aid. One way to do this is to create a second window that is the shadowmap size. You can map the depth values to a grey scale (i.e., -1 maps to black and 1 maps to white) using a simple shader.

3 Submission

We will set up a **gforge** repository for each student on the Computer Science server (see the *Lab notes* (Handout 2) for more information on **gforge**). We will collect the projects at 3:30pm on Wednesday February 20th from the repositories, so make sure that you have committed your final version before then. You will also be expected to demonstrate your code during Lab in Week 7 (February 20).

History

2008-02-18 Fixed lighting equation ($\mathbf{min}$ should be $\mathbf{max}$).

2008-02-07 Original version.

2008-02-12 Updated due date.