Due March. 17, 2008

In this project you will write GLSL shaders to solve 3 problems. Problem 1 is a variation of the brick shader, while problems 2 and 3 extend shaders from the GLSL examples covered in class (chapter 10 of OpenGL Shading Language book, links on course website to book examples)

For each problem you will build a full application that will permit the following:

□ Interactive view control
□ interactive lighting model parameter control
□ Support for both color interpolated shading and normal vector interpolated shading.
□ interactive light position control

Additional Requirements:

□ Project Submission: Turn in all source code including Makefile. Please document your sources adequately.
□ Project must be demonstrated to one of the instructors.
□ Implementation in C++; you can use GLUT or any equivalent UI toolkit for application control.
□ Your implementation must be portable, must run on Unix!

1. [Elliptic Polka Dot Shader]: You will build a shader programs (similar to brick shader) to generate elliptical polka dots on surfaces. An ellipse equation is given by

\[
\left(\frac{x}{A}\right)^2 + \left(\frac{y}{B}\right)^2 = 1
\]

where A and B are the radii of the ellipse. Example images are shown below, also refer to

http://web.engr.oregonstate.edu/~mjb/cs519/Projects/proj02.html

In addition to the common requirements, your shaders should demonstrate the following:
You will demonstrate your shader on two surfaces (1) sphere, and (2) a cube.

Transition regions (polka dot boundaries) must be smooth (use the smoothstep() function)

Application (through a GUI) must be able to control polka dot dimensions - width, height (for instance, should be able to draw horizontal or vertical polka dots), as shown in figures below

2. [Multitexturing]: Implement a full application of the multitexturing example covered in class (Chapter 10, GL Shading Language book, links on the course website). In addition, extend the application to

- Generate animations of smooth movement of the sun to simulate transition from daytime to nighttime.
- Also draw the actual rotation axis of the earth, in order to demonstrate its tilt, as well as the orbital path (assume circular orbit). Locating the sun and its true orbit should make the simulation more realistic.

3. [Cube Mapping]: Implement the full application of the cubemapping example covered in class (Chapter 10, GL Shading Language book, links on course website). A set of example textures will be provided (you can also generate your own). Extend this application as follows:

- Compare the shader application by implementing a traditional GL application that uses the fixed function pipeline to perform cubemapping.