

Spring 2008
22C:251 Advanced OpenGL Techniques
Assignment 2

Due: Thursday, February 28th at 11:59pm

Goal: Use the stencil and depth buffers to draw models using various line types. Enable OpenGL fog, and allow the user to select and move around various objects simply by clicking on them.

Problem 1 (20 points): Create an OpenGL/GLUT program that loads some simple models (e.g., 1,000–3,000 polygons) such as those provided on the web and draws at least three onscreen at the same time. Again, put a textured ground plane in the scene. For each of the onscreen models, the user should be able to selectively draw them in one of four modes:

1. Standard OpenGL lighting
2. Standard wireframe mode
3. Without hidden lines, drawn using a depth test and polygon offset
4. Wireframe mode where frontmost lines have “halos”

Make sure the user can rotate the scene (so that I can easily move the viewpoint so one object lies behind another).

Problem 2 (10 points): Add a menu/key callback that allows me to enable and disable OpenGL fog. You can pick your favorite fog mode (linear, exp, exp2) as long as the values when enabled clearly add fog towards that back of your scene.

Problem 3 (20 points): Write code to load and execute GLSL shaders. I recommend you write an easy-to-use C++ class to hide the details and simplify your display routine.

1. Write and load a shader program (to apply to the floor) that replaces the texture with the sinusoidal pattern from 22c:151 Homework 11 (please bind the +/- keys to change the parameters). This can be done with a single fragment shader.
2. Write a shader program that changes the alpha component of an object’s color depending on the surface orientation. Areas that directly face towards (or away from) the eye should be completely transparent; locations with normals perpendicular to the viewing direction should be completely opaque. As the object (or eye) rotates, these alpha values should vary based upon the new orientation. This obviously requires the use of blending, the alpha variation can either be implemented as a vertex or geometry shader, and you may wish to write a corresponding fragment shader to have control throughout the pipeline. Apply this shader program to the complex objects from Problem 1.

NOTE: Please include a README file describing the platform your program runs on as well how to compile your program. If you do not include all the models with your submission, tell me where the models must be for your program to find them.

NOTE: Post images from this assignment on your web page. Send me a link to your web page, if you have not already done so!