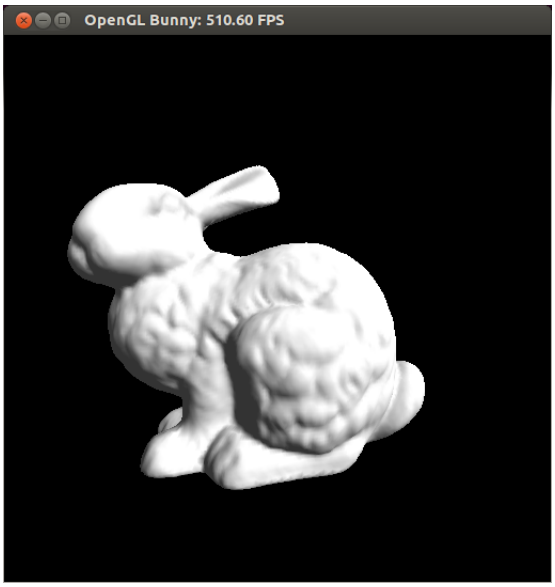
Programming Assignment 3

COMP 575/770 Spring 2013

Due: March 28, 2013

In this programming assignment, you will use OpenGL to render a bunny. A sample screenshot is provided below.



Triangle Mesh The bunny is provided as a triangle mesh in `.obj` format, downloadable from the course webpage. The bunny is represented using 69,451 triangles. Sample code to load the triangle mesh from the `.obj` file is also provided at the course webpage. This code is written in C++, so if you're using a different language, you will have make the necessary translations.

Transform Parameters While rendering the bunny, use the following transformations:

- Scale the bunny uniformly by a factor of 10 in all directions, and then translate its origin to the position $(0.1, -1, -1.5)$.
- Position the camera at $\mathbf{e} = (0, 0, 0)$, with coordinate axes $\mathbf{u} = (1, 0, 0)$, $\mathbf{v} = (0, 1, 0)$, and $\mathbf{w} = (0, 0, 1)$.
- Use perspective projection, with $l = -0.1$, $r = 0.1$, $b = -0.1$, $t = 0.1$, $n = -0.1$, and $f = -1000$. (Note that OpenGL reverses the signs of n and f.)
- Use a viewport transform with $n_x = n_y = 512$.

Shading Parameters For hidden surface elimination, use a depth buffer, but disable back-face culling. Render the bunny using OpenGL's fixed-function shading pipeline, with the following parameters:

- Assume the bunny has $k_a = k_d = (1, 1, 1)$. Assume it is not specular, i.e., $k_s = (0, 0, 0)$ and $p = 0$.
- Assume an ambient light intensity of $I_a = (0.2, 0.2, 0.2)$.
- Add a single *directional* light source with ambient color $(0, 0, 0)$, diffuse color $(1, 1, 1)$, and specular color $(0, 0, 0)$. Assume the light strikes every point along the direction $(-1, -1, -1)$ (normalized, of course). *(Hint: You will have to set the position of the light to a homogeneous vector with $w = 0$.)*

Rendering This assignment consists of 2 parts:

1. **Immediate Mode**
   Render the bunny using OpenGL's *immediate mode*. In other words, specify triangles to OpenGL using `glVertex3f` and `glNormal3f` every frame.
2. **Vertex Arrays**
   Now create a vertex array object (VAO). Bind it to one or more vertex buffer objects (VBOs) containing the position and normal data for each vertex. (You are free to store this information any way you like.) Also create a buffer to store triangle indices. Copy all the data to GPU memory before you start rendering, and render the bunny using `glDrawElements`.

In each case, measure how long it takes to render the bunny. There is code provided on the course webpage that will automatically update the title of your GLUT window with an FPS counter. Report ballpark values of this counter for each case in your README.

GLEW On some platforms, the OpenGL functions for creating VBOs and/or VAOs may not be available by default. If you get compilation errors due to this, you will need to install and use GLEW from `http://glew.sourceforge.net/`. The Usage page on the GLEW website contains the necessary information to add GLEW to your code.