CS535: Assignment #2 – Scene Modeling

Out: September 13, 2007
Back/Due: October 2, 2007

Objective:
This focus of this assignment is to obtain a good understanding of transformation, rasterization, and shading using the OpenGL Graphics Pipeline. You can recycle as much as you like from previous assignments; however, OpenGL has all the necessary matrix operations you will need. You will write a program that loads in a scene model and for each frame issues OpenGL commands to draw the model primitives, control the lights and the light model, and move the viewpoint.

Summary:
The assignment is to implement a program which renders a 3D scene. The user specifies a file containing a description of the scene. The file is in a simple scene text file format. Once the scene is loaded it can be rendered (interactively). The user can control the viewpoint and navigate through and around the scene and can control the lighting/shading parameters. All control is via a GLUI-based user interface. Changes to any of the parameters should be effective in the next drawn frame.

Specifics:

(0) Scene file format. The scene geometry is stored in a simple file format, called ".sim". Countless file formats are possible, this is just one of them. The format is:

```
# this is a comment (textures and polygons can be intermixed)
texture <num textures>
filename0.ppm
filename1.ppm
filename2.ppm
polygon <num polygons>
<num verts> <tex index, -1 == no texture> [NORMALS] [RGB] [UV]
<x> <y> <z> <r> <g> <b> <u> <v>
...
<x> <y> <z> <r> <g> <b> <u> <v>
<num verts> <tex index, -1 == no texture>
<x> <y> <z> <r> <g> <b> <u> <v>
...
<x> <y> <z> <r> <g> <b> <u> <v>
...
```

and empirical examples of this file format are included in the example model files included with this assignment. The format allows you to specify textures and polygons and the position, normal, color, and texture coordinates per vertex. You may store the geometry and textures however you like, but the model should render interactively on just about any computer.

(1) Shading and Lighting. The above file format includes a color for each vertex (and thus indirectly for each pixel). The program should permit changing the global shading and light parameters. In particular, (a) the number of OpenGL lights (up

to eight), (b) selection of point or directional light, (c) the location of the lights, (d) the direction of the lights, (e) the ambient color of the lights, (f) the diffuse color of the lights, and (g) the specular color of the lights. The location/direction of the lights should be controlled in a manner similar to the viewpoint (e.g., using translation and roll, pitch, and yaw). Further, the program should allow specifying for the entire scene (h) a global ambient and diffuse reflection coefficient, (i) a specular reflection coefficient, and (j) the specular exponent (e.g., shininess). By altering these parameters, a model can be given a general satin look, glossy look, metal look, plastic look, etc. Some of the models contain already a per-vertex global-illuminated point color. Thus, you must support two overall shading modes: (i) no shading and lighting – just use the vertex color as is, (ii) consider the lights you are creating to be "spot lights" that modulate the global illumination, and (iii) replace all vertex colors with a single user-selected color (e.g., make the entire model white, yellow, light blue, or some other color).

(2) Rendering: the model should be rendered, shaded, and texture-mapped each frame.

(3) User Interface: a GLUI-based user interface must be used to allow the user to specify all the parameters. Reasonable initial values should be given to all parameters at program initialization. The default viewing position should be (0,0,0) and default viewing direction should be (0,0,-1). A reasonable field of view is 60 degrees. A good near and far distance should be computed based on the world-space size of the model. The camera control parameters should be the same as with Assignment #1 (if you implemented the trackball that would be great to port over to this assignment). The GUI should also include parameters to control all the previously described shading and lighting options.

Extra Credit (0 to 20%): Implement additional shading and lighting effects, e.g. attenuation by distance, atmospheric effects, shadows, etc; additional (and intuitive to use) GUI parameters should be added for these effects; please also include a short description of what you added, how it is implemented, and how to use it.

Grading:

Your program will be tested against all the provided input files and some other ones too. We will use the interface to alter the aforementioned parameters and expect to see the correct behavior. If the interface does not allow a certain parameter to be changed, it will be considered not implemented at all. If the program does not compile, zero points will be given.

To give in the assignment, email the TA, as with the previous assignment, by the due date and time. It is your responsibility to make sure the email is delivered/dated before it is due. **Hint: don't wait until the last moment to hand in the assignment**.