ICS 186A: Computer Graphics
Spring 2002
Gopi Meenakshisundaram

Programming Assignment 2
Assigned: Wednesday, April 19, 2002
Due: Friday, April 26, 2002, **11:59pm**.
Estimated Time: 6 hrs

Source Code Path: /home/graphics/teaching/ics186a/gopi-S02/framework-lab2

1. Implement a stack of matrices and the following functions to operate on the stack of matrices. (This stack is in addition to the stack you implemented in Programming Assignment 1).

```
void my_gluLookAt(Gldouble eyeX, GLdouble eyeY, GLdouble eyeZ,
                  GLdouble centerX, GLdouble centerY,
                  GLdouble centerZ,
                  GLdouble upX, GLdouble upY, GLdouble upZ )
void my_glFrustum(GLdouble left, GLdouble right,
                  GLdouble bottom, GLdouble top,
                  GLdouble zNear,   GLdouble zFar )
void my_gluPerspective( GLdouble fovy,   GLdouble aspect,
                        GLdouble zNear, GLdouble zFar )
```

(NOTE: "glu" instead of "gl" for LookAt and Perspective)
This stack will be comparable to what OpenGL calls the Projection Matrix Stack. To switch between the two stacks, you will also need:

```
void my_glMatrixMode( Glenum mode );
```

This function will allow you to switch between the two matrix modes, `GL_MODELVIEW` and `GL_PROJECTION`.

2. Derive how you arrived at the matrices you have implemented for the above three functions. Use your answer to Question 6 of Written Assignment 2 to derive the matrix for my_gluLookAt. Use the discussion we had in the class to derive the matrix for my_glFrustum. my_glPerspective is a special case of my_glFrustum, where off-axis projection is not allowed. So derive the matrix for my_glPerspective substituting the appropriate values in the matrix for my_glFrustum. This is the written part of the assignment to be submitted during the class hours on April 26,2002.

3. Implement the computation of normal vector of a triangle, when triangle vertices are given in the counter-clockwise direction. If A,B,C are the vertices of the triangle given in the counter clockwise direction, then the normal vector is ABxAC. Finally normalize the normal vector by dividing it by its length to get the unit normal vector. The normals should be computed in a function called `void computeFaceNormals(void)`. The resulting normals must be store in a globally defined array.

4. Implement the computation of the normal vector at a vertex. The normal vector at a vertex is the average of the normal vectors of the triangles incident on that vertex. The vertex normals

should be computed in a function called `void computeVertexNormals(void)`. Each resulting vertex normal must be stored in a globally defined array.

(The computation of the vertex and face normals must happen before a call to `initCallLists()` is made. The normals will be used to create the display list.)

5. Add an additional animated light to the scene. This requires modifying the `initDisplay()` function in the file `cube.c` to initialize the lighting in OpenGL. First, define the ambient light color, the diffuse reflection color, and the specular highlight color locally in `initDisplay()`. The light's position should be defined globally, such that it can be used to initialize the additional light in `initDisplay()` and be modified in another function. The starting position of the light should be {1, 1, 3}. Then define a function called `void animate(void)` which modifies the light's position such that the light travels back and forth between x = -1 and x = 1. Add an additional case to `readKeyboard()`, where if the user hits the 'a' key, the GLUT Idle function will be defined to be the function `animate()`. If the user hits 'a' again, the idle function is set back to `NULL`.