CS5500 Computer Graphics
Assignment 5

Assigned: May 14, 2007
Due: 23:59 May 31, 2007 (10% penalty for each day late)

Extend your graphics pipeline to handle:

(1) Triangles in 3D world space
(2) Lighting using the Phong illumination model
(3) Rasterization and Gouraud shading
(4) Hidden surface removal using the Z buffer

The input file to your program is now extended to:

```
eye: xEye, yEye, zEye
lookat: xLook, yLook, zLook
up: xUp, yUp, zUp
Perspective: vFOV, near, far
display: width, height

light_pos: x, y, z, w
light_color: r, g, b

translate: x, y, z        <--- Optional
rotate: angle, xAxis, yAxis. zAxis <--- Optional
color: r, g, b
material: Ka, Kd, Ks, shininess
triangle:
   normal: nx1, ny1, nz1
   vertex: x1, y1, z1
   normal: nx2, ny2, nz2
   vertex: x2, y2, z2
   normal: nx3, ny3, nz3
   vertex: x3, y3, z3
triangle:
   ...
```

TASKS

(a) When you transform a vertex from 3D world space to 2D screen space, remember to compute its Z as well. The Z values will be used for hidden surface removal.

(b) In the above example, the same color is set to all 3 vertices of the triangle. However, it is possible to set a different color to each vertex by adding a "color: r, g, b" line before each vertex.

(c) Apply Phong illumination model to shade each triangle vertex. Store the shaded color with the vertex for the next step (rasterization).

(d) Once you have computed the screen coordinates for all 3 vertices of the triangle, you may now fill its interior by scan line conversion. Use the edge equations of the triangle to determine whether a pixel should be shaded.

CS5500 Computer Graphics
Graphics Pipeline (Part 3 of 3)

Assigned: May 8, 2006
Due: 23:59 May 28, 2006 (10% penalty for each day late)

Extend your graphics pipeline to handle perspectively correct texture mapping.
The input file to your program is now extended to:

```
eye: xEye, yEye, zEye
lookat: xLook, yLook, zLook
up: xUp, yUp, zUp
Perspective: vFOV, near, far
display: width, height

light_pos: x, y, z, w
light_color: r, g, b

translate: x, y, z        <--- Optional
rotate: angle, xAxis, yAxis. zAxis <--- Optional
color: r, g, b
material: Ka, Kd, Ks, shininess
texture2d: filename.ppm
triangle:
   normal: nx1, ny1, nz1
   texcoord: u, v
   vertex: x1, y1, z1
   normal: nx2, ny2, nz2
   texcoord: u, v
   vertex: x2, y2, z2
   normal: nx3, ny3, nz3
   texcoord: u, v
   vertex: x3, y3, z3
triangle:
   ...
```

TASKS

(e) The texture image is stored in the PPM format. See the example code in the data folder to read and write image files in the PPM format.
(f) Mip-mapping is not required in this project.
(g) You may use either the nearest neighbor or the bilinear interpolation for texture filtering.