SFU CMPT 361
Computer Graphics
Spring 2017
Assignment 4

Assignment due Thursday, April 6, by 11:59 pm.

For this assignment, you are to expand your renderer from assignment 3 with point light sources and phong-model lighting calculations.

The required display is 1 panel, set up just as in assignment 3: the drawing area should be 650 x 650 pixels, and the white margins are all 50 pixels wide or tall. Thus, the total display area is 750 x 750 pixels. Use black for the background of the viewing area.

Command line arguments
Your program must accept a command line argument, which is the filename of the simp file that you are to read and render.

Graphics file expansion
We will add some commands to the "simp" format files, and modify others. All of the previous commands which are not included here are still accepted as part of a simp file. Again, all numbers are floats or doubles (your choice). The light command is written here on two lines but should be written on one line in a simp file. All colors are in [0, 1] coordinates.

Line format	**Meaning**
light <red> <green> <blue> <A> <B>	At the origin in the coordinate system defined by the current transformation matrix (object space, or CTMCS), place a point light source with R, G, B intensities, and attenuation constants A and B, as given. Advice for simp file writers: Generally, the intensities can be any number, but initially put them between 0 and 1 and then scale them up if necessary. There may be multiple light commands in a simp file, and the contribution of all lights should be taken into account. No commas between the values.
surface (r, g, b) k_s p	Everything from this point forward will have default surface color k_d = (r, g, b). At the start of the run, this default color starts as pure white: (1, 1, 1). This command also sets the specular reflection coefficient of everything from this point forward to the scalar k_s and specular exponent to p. k_s and p start at the beginning of the run at .3 and 8, respectively. Note that this is the only way to set these coefficients, and they apply regardless of whether a polygon has specified color or not.
phong	Set the shading style to phong. This is the default shading style to be used if no other rendering style has been specified.
gouraud	Set the shading style to gouraud.
flat	Set the shading style to flat.

None of the "this point forward" properties are affected by push and pop ({ and }).

Note: in all that follows, I'm using euclidean and not homogeneous vectors. In world space.

Lighting calculation
The lighting calculation is now using the model:

$$I_\lambda = k_{d\lambda} I_{a\lambda} + \sum_i I_{i\lambda} f_{att,i}(k_{d\lambda} (\hat{N} \cdot \hat{L_i}) + k_s(\hat{V} \cdot \hat{R_i})^p)$$

Where

$I_{a\lambda}$	Is the ambient light intensity
$I_{i\lambda}$	Is the intensity of light i.
$f_{att,i} = \frac{1}{A_i + B_i d_i}$	A_i and B_i are the attenuation constants from the **light** command for light i; d_i is the world-space distance from the light to the surface.
λ	in a subscript denotes an RGB vector. The above equation must be evaluated three times: once for the red component, once for the green, and once for the blue. Be sure to compute the things that are not color-varying (for example, $k_s(\hat{V} \cdot \hat{R_i})^p$) only once.
$\widehat{\ }$	Denotes a unit vector.
$\hat{L_i}$ and $\hat{R_i}$	Have the subscript i; the i may be missing its dot on some devices. These denote the vector from the point being illuminated to light i, and that same vector reflected through the normal vector N, respectively.
$\hat{N}$	Normal vector to surface at the point being illuminated.
$\hat{V}$	View vector (from eye to point being illuminated).
$k_{d\lambda}$	Color (diffuse reflection coefficient) of object. From surface command or from vertex colors.
k_s, p	Specular reflection coefficient and exponent. From surface command.
$\sum_i$	The sum is over all light sources encountered so far in the scene.

Do not apply the lighting model to line primitives. Use ambient light only (as in assignment 3) on lines.

Shading style
There are three options (phong, gouraud, and flat) for the rendering style.

In flat shading, one averages the vertices of a polygon to find a center point on the primitive. If there are normals present at the vertices, then the renormalized average of these normals is used the normal for the lighting calculation:

N = normalize($\frac{N_1+N_2+N_3}{3}$).

Otherwise the face normal is used:

N = normalize((V_2 - V_1) × (V_3 – V_1)).

Next, using the center point with the normal, apply the lighting model to get a lighting value (RGB vector). Use this value as the lighting-calculation color for all pixels in the polygon. (The lighting-calculation color is modified by the atmospheric perspective (depth) shading as described in assignment 3.)

To do gouraud shading, at each vertex of a polygon, find a normal. If the vertex is from an obj file and has a specified normal, use it. Otherwise, use the face normal (see previous paragraph) at every vertex. Next, at each vertex, apply the lighting model to get a lighting value. Then blerp these lighting values across the polygon, using the blerped value as the pixel value at each pixel.

To do phong shading, find a normal at each vertex as with gouraud shading. Now, blerp the normals, the world space points, and color (if necessary) across the polygon. At each pixel, used the blerped point, (renormalized) normal, and color (k_d) in the lighting model to obtain a pixel value for the pixel.

README

Do not forget to include a readme file with your submission, including your development environment, any problems still existing in the code, instruction on how to run, and any comments or questions for the TA. (Send questions to me directly to me. Don't send me more than a snippet of code—bring your code in to me during office hours if it's a longer question.)

Other details and implementation hints will be discussed in class. It is your responsibility to know all of these details, so do not miss class.