Programming Lab 09

In this, you implemented two functions to draw a mesh: one to draw the mesh as a wireframe, and one to draw it as a solid figure. However in that assignment, a special perspective projection was used (essentially a fixed camera). Here you will do the same, but from a given cameras point of view.
The header file DT285_DrawingCam.h, which I will supply, contains the following two function prototypes.

void DisplayEdges(Mesh& mesh, const Affine& obj2world, const Camera& cam, const Vector& clr);
void DisplayFaces(Mesh& mesh, const Affine& obj2world, const Camera& cam, const Vector& clr);

The header files Camera.h, Mesh.h, Affine.h, and glut.h have been included. In both functions, obj2world is the affine transformation that maps the object space vertices of mesh to world space. Each function should draw mesh from the point of view of cam.

For DisplayEdges, the wire frame should be drawn using clr as the color. And for DisplayFaces, faces should be drawn using diffuse shading, where clr gives the color of a face at normal incidence (maximum brightness). The components(x,y,z) of clr specify the red, green, and blue colors, and each component values should be assumed to be in the interval [0, 1].

- You are to assume that the background has been cleared immediately prior to calling DisplayEdges and DisplayFaces.

- Backface culling should be used to determine if a mesh face should be rendered or not. That is, face f will be drawn if

$$\vec{n} \cdot \overrightarrow{PE} > 0$$

where n is the outwardly pointing surface normal for face f, P is any point on f and E is the center of projection. Note that this can be done in either world space, or in camera space (remember that in camera space, E is the origin).

- For the diffuse shading of each visible mesh face, assume that the light shines in parallel to direction l that the camera is looking in (equivalently, in the direction parallel to the back vector of the camera). Recall that if clr has values (R,G,B), then the face f should be drawn using the color

$$(\mu R, \mu G, \mu B), \quad \text{where} \quad \mu = \frac{|\vec{l} \cdot \vec{n}|}{\|\vec{l}\| \, \|\vec{n}\|}$$

and n is the outwardly pointing surface normal for face f.

- Simple view frustum culling should be applied: any face or edge of the mesh that has at least one vertex that lies behind the viewer should not be rendered. This is most easily done in camera coordinates, since in camera coordinates a point P = (x,y,z) is behind the viewer if z >= 0.

Your submission for this task should be visual studio code project.