DT285 Programming Lab 3

The following gives the contents of the le Mesh.h, which declares an interface class for the 3D triangular meshes that we will use in this course. Actual meshes will be derived from this class.

```
struct Mesh {
        struct Face {
        int index1, index2, index3;
        Face(int i, int j, int k) : index1(i), index2(j), index3(k) {}
};
struct Edge {
        int index1, index2;
        Edge(int i, int j) : index1(i), index2(j) {}
};
virtual ~Mesh(void) {}
virtual int VertexCount(void) = 0;
virtual Point GetVertex(int i) = 0;
virtual Vector Dimensions(void) = 0;
virtual Point Center(void) = 0;
virtual int FaceCount(void) = 0;
virtual Face GetFace(int i) = 0;
virtual int EdgeCount(void) = 0;
virtual Edge GetEdge(int i) = 0;
};
```

The Face class is used to represent the triangular faces of the mesh. The values of index1, index2, and index3 give the indices of the triangles vertices in the mesh's vertex table. Similarly, the Edge class is used to represent the edges of the mesh. The values of index1 and index2 specify the indices of the edges endpoints in the vertex table. A (nonabstract) subclass of the Mesh class should implement the functions in the interface, which are detailed below.

~Mesh() – (destructor). If a derived class does not make use of dynamically allocated memory, an explicit destructor need not be provided.

VertexCount() – returns the number of vertices in the vertex table of the mesh.

GetVertex(i) – returns the i th vertex in the vertex table. The vertex values are given in object space. You may assume that i is value between 0 and one less than the return value of VertexCount. No error checking need be made.

Dimensions() – returns the vector $\Delta x\ \Delta y\ \Delta z$ that gives the dimensions of the axis aligned bounding box in object space that contains the mesh.

Center() – returns the center (Cx,Cy,Cz) of the axis aligned bounding box in object space that contains the mesh. That is, vertices of the mesh have coordinates (x,y,z) where

$$C_x - \tfrac{1}{2}\Delta x \le x \le C_x + \tfrac{1}{2}\Delta x, C_y - \tfrac{1}{2}\Delta y \le y \le C_y + \tfrac{1}{2}\Delta y, C_z - \tfrac{1}{2}\Delta z \le z \le C_z + \tfrac{1}{2}\Delta z$$

FaceCount – returns the number of triangular faces in the mesh.

GetFace(i) – returns the i th face in the mesh. You may assume that i is value between 0 and one less than the return value of FaceCount. No error checking need be made.
See the Orientation convention description below.

EdgeCount – returns the number of edges in the mesh.

GetEdge(i) – returns the i th edge in the mesh. You may assume that i is value between 0 and one less than the return value of EdgeCount. No error checking need be made.

Orientation convention. The triangular faces of the mesh m should be oriented in a coun terclockwise fashion. That is if i is a valid index, then the vertices V1 V2 V3 de ned by

```
Mesh::Face F = m.GetFace(i);
Point   V1 = m.GetVertex(F.index1),
        V2 = m.GetVertex(F.index2),
        V3 = m.GetVertex(F.index3);
```

are such that the vector (V2 - V1) X (V3 - V1) is an outwardly facing surface normal for the face F of the mesh.

Your task

The header filecs250_h2.h contains the following function prototypes.
MatrixPerspectiveProjection(float D);

void DisplayEdges(Mesh& m,const Affine&A, constMatrix& P,const Vector&c);

void DisplayFaces(Mesh& m,const Affine&A, constMatrix& P,const Vector&c);

PerspectiveProjection(D) – returns a matrix representation for the elementary perspective projection whose center of projection is the point E=(0,0,D) and the plane of projection is z=0. Recall that one possible homogeneous coordinate representation for this matrix is

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1/D & 1 \end{bmatrix}$$

DisplayEdges(m, A, P, c) – draws mesh m as a wireframe; i.e.,only the edges of themesh are drawn. The affine transformation A specifiles the transformation from the mesh's object space to OpenGL worldspace.After the mesh vertices have been transformed to worldspace, the perspective projection P should be applied to the vertices before rendering the mesh edges.Edges should be rendered in the color c.