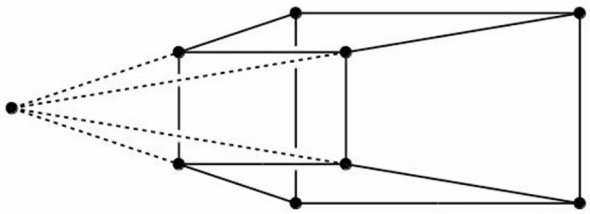
Programming Lab 07

TASK #1 Implement a frustum mesh class.

I will provide you with the header file FrustumMesh.h, which contains the following declarations.

```
class FrustumMesh : public Mesh {
  public:
    FrustumMesh(float fov, float aspect, float near, float far);
      int VertexCount(void);
      Point GetVertex(int i);
      Vector Dimensions(void);
      Point Center(void);
      int FaceCount(void);
      Face GetFace(int i);
      int EdgeCount(void);
      Edge GetEdge(int i);
      private:
      Point vertices[9];
      Point center;
      Vector dimensions;
      static const Edge edges[16];
      static const Face faces[12];
};
```

The purpose of this class is to represent the viewing frustum of a canonically oriented camera: a camera whose center of projection is the origin, and whose right, up, and back vectors are parallel to (respectively) the x axis, y axis, and z axis. As indicated in the figure below, the wire frame of the mesh should also include the center of projection. However, the faces of the mesh do not include the center of projection.



This means that the mesh will have a total of 9 vertices, although the triangular faces will only use 8 of these vertices. Note that since the mesh represents a canonically oriented camera, the view frustum geometry is completely determined by horizontal field of view (fov), the viewport aspect ratio (aspect), and the distances (near and far) to the near and far faces of the frustum.

TASK #2 Re-implement the Drawing Function to support GLSL.

From previous assignment that you have implemented DT285_Drawing.cpp to display wireframe and solid mesh with DisplayEdges() and DisplayFaces() function reimplemented these functions to support opengl shading language pipeline.

Your submission for both part of the assignment should be the zipped visual studion project consist of source file FrustumMesh.cpp and vertexShader file and fragmentShader file