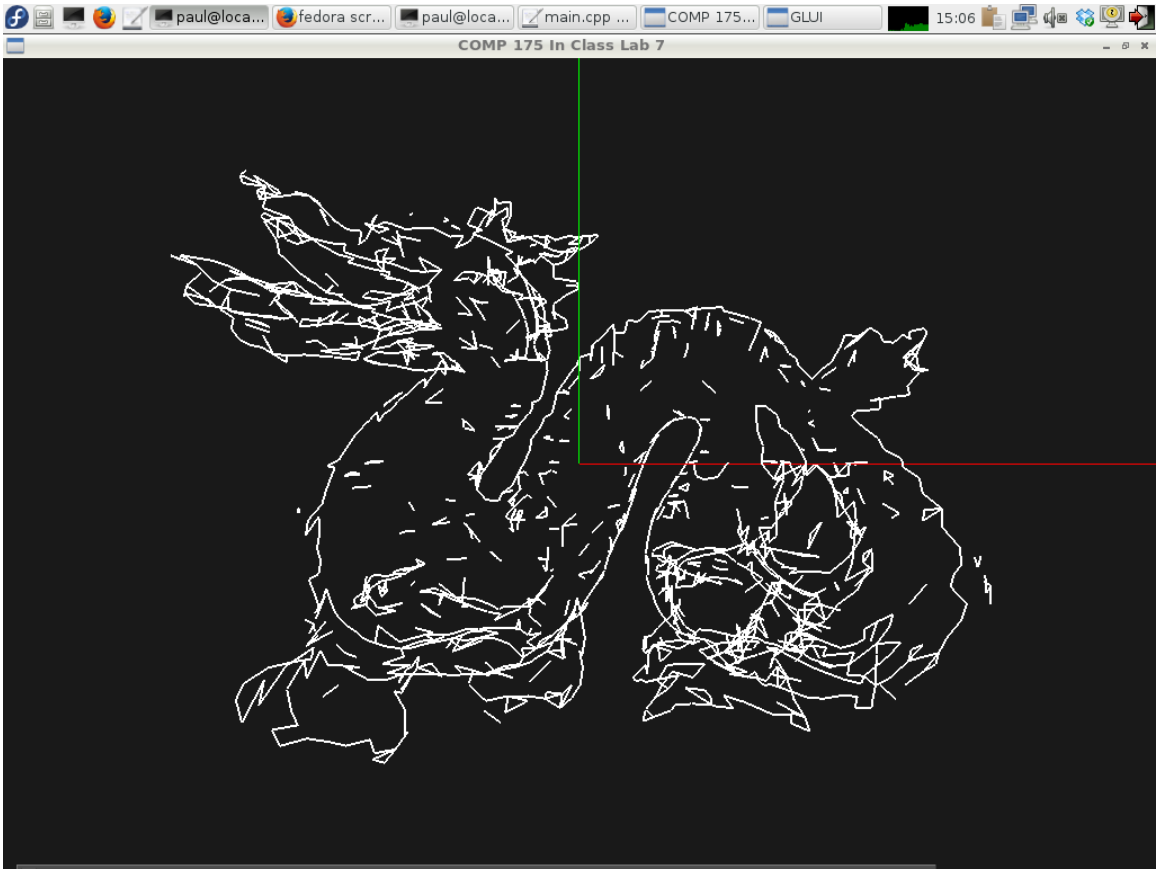
Silhouette



Description:

Today you will be working on code to find and display the silhouette edge of a mesh object. The silhouette edge is the edge between faces that are in view and faces that are not in view.

The way to calculate this is to take the dot product of the look vector and the face normal for each face. If this product is positive, then the face is facing towards the camera, but if the product is negative, then the face is facing away from the camera.

To be able to do this calculation, you will need to build a data structure to keep track of all of the edges in the mesh, such that each edge knows which two faces it exists between.

Currently, the ply.h file assumes that your edgeList is an array of "edge" objects. Each "edge" keeps track of the two vertices that it connects, as well as the two faces that it neighbors (definition is in geometry.h). However, you can rewrite either the edge class (in geometry.h), or change the definition of edgeList in any way you like.

Since the viewer allows rotation around the y axis, the look vector has components in X and Z which can be calculated from the angle of rotation (i.e., when rotation=0, lookX=0.0 and lookY=1.0).

Your Task:

- Fill in the empty functions in ply.cpp.
  - findEdges fills in the edgeList (don't forget to update edgeCount)
  - renderSilhouette uses the edgeList to find and draw all of the edges which are part of the silhouette.

    The idea is to loop through all the edges in renderSilhouette(), check the two faces that it neighbors, and use a dot product (between the normal of the face and the camera's look vector) to determine if the two faces are both front-facing, both back-facing, or one front-facing and one back-facing. Obviously, the edge should only be considered as a silhouette edge (and therefore rendered) if it's the last condition.

Files Given:

ply.cpp – Fill in findEdges and renderSilhouette
ply.h – edgeList is declared here.
geometry.h – the edge struct is declared here.
data/ - some .ply files
Algebra.h
entity.cpp
entity.h
main.cpp

Going Further

Did you enjoy this lab? Consider making the following additions:

- Perform and edge detection such that you only get an outline of the object (one continuous line i.e. a tracing around the model)