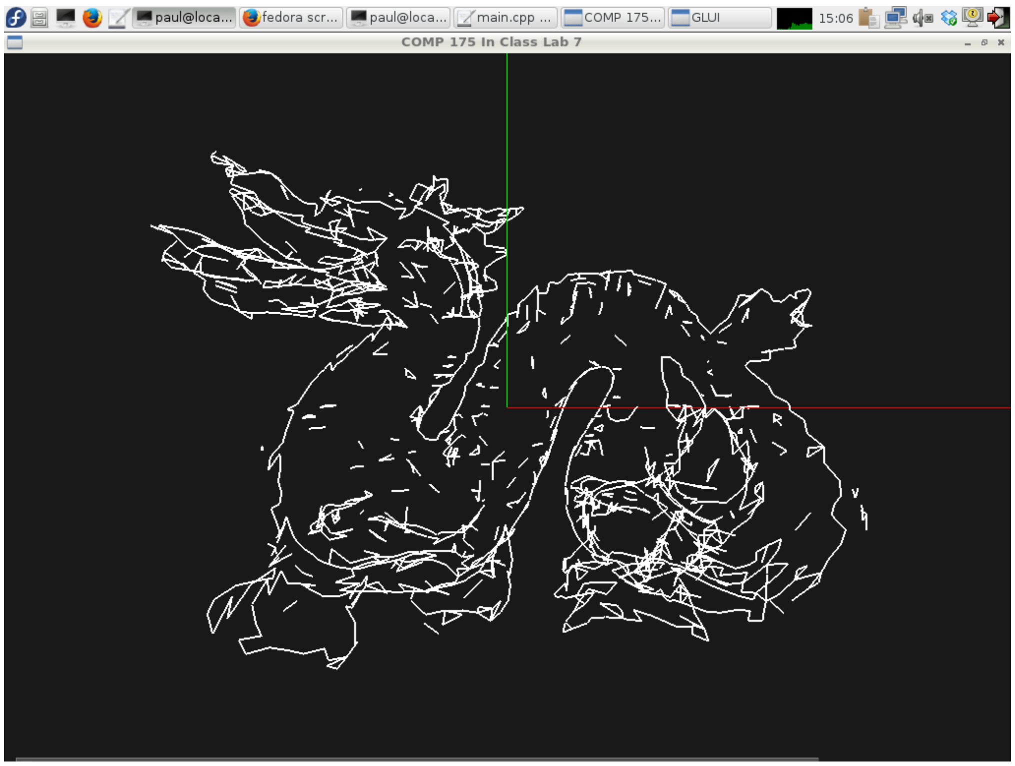
Lab 2: Silhouette



Description:

Today you will be working on finding and displaying the silhouette edges of a 3D tessellated object. The silhouette edge is the edge between faces that are in view (front-facing) and faces that are not in view (back-facing).

The way to calculate this is to take the dot product of the look vector and the face normal of a face (in our case, a face is always a triangle). If this product is negative, then the face is facing towards the camera, but if the product is positive, then the face is facing away from the camera.

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 sits between.

To make things easier for you, we assume that the camera is orthographic (i.e. it is a parallel projection). As such, the look vector for all triangles is always the same. (Imagine, if this is a perspective projection, the look vector will be different for each triangle).

Your Task:

- First, you will need to fill in the ply::findEdges() function. Specifically, you will build a data structure that stores all the edges in the 3D object (without having duplicated edges in the array). You will store these edges in the array edgeList (see ply.h).

  - Each edge should know about the (one or two) faces it shares (see geometry.h). That is, as you identify these edges, make sure you have pointers to its two neighboring faces.
    - Note: an edge that only has one neighboring face is always a silhouette edge
  - The general approach would be to loop through all the faces. Add edges into a dynamic array (e.g. std::vector) if the edge doesn't already exist in the array. Finally, copy the content of the vector into the edgeList array. (And don't forget to update the variable edgeCount).
- Second, you need to fill in the function ply::computeFrontFace().
  - Specifically, in this function, given the lookVector, you will assign whether a face is front facing or not.
    - We have computed the lookVector for you (called curLookVector) in the render loop. This look vector changes as the object is rotated. You may assume the use of this variable when computing whether a triangle is "front" or "back" facing (see the next bullet point).
  - You can check your result by turning on the toggle "Front v. Back Face" in the interface.
  - If you are computing the lookVector and the front-v-back facing properly, you should only see green triangles (that is, front-facing triangles are drawn in green, back-facing triangles as drawn with red).
- Lastly, complete the ply::renderSilhouette() function. This is the function that draws (only) the silhouette edges.
  - The strategy would be to loop through all the edges. For each edge, find its two neighboring faces. If one of those faces is front facing and the other is back facing, then the edge is a silhouette edge. In such a case, draw the edge.
  - Keep in mind that for some of the mesh files, an edge would only have one neighboring face. These should also be drawn as a silhouette edge.

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
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)