Homework 2

Improve the OpenGL GUI developed in HW 1. Using MeshLib (half-edge data structure) to do some basic geometric computation on triangle meshes.

1. Include meshlib files into hw1 project and make the GUI work like hw1. Now, instead of using your original OBJ reading function, you can directly use ReadOBJFile() function to read an OBJ file to the mesh object. Then, revise the Bounding box computation, and mesh rendering functions accordingly to get the mesh rendered.

2. Implement the `ComputeNormals()` function to compute the normal vector on each vertex. The vertex normal is a weighted average of normal vectors of this vertex's surrounding face,

$$N(v) = \frac{\sum_{f_i \in N_F(v)} \alpha_i N(f_i)}{|\sum_{f_i \in N_F(v)} \alpha_i N(f_i)|}$$

   where $N_F(v)$ is the set of neighboring faces surrounding the vertex v, and the weight α_i is the interior corner angle of f_i at v.

3. Implement a "RenderEdge()" function, so that when you press "e", you can see the edge connectivity shown in blue together with the mesh.

4. Trace all the boundary loops and compute the boundary loop number b. Print b on the screen. Also implement a "RenderBoundary()" function, so that when you press "b", you can see the boundary loops shown in yellow on the mesh.

5. Compute the Gaussian curvature κ_G on each vertex. Colorize the surface:
   a. color a vertex to green if its curvature $\kappa_G < -0.05$ and its curvature is a local minimum (i.e., value ≤ curvatures of its surrounding vertices)
   b. color a vertex to red if its curvature $\kappa_G > 0.05$ and its curvature is a local maximum (i.e., value ≥ curvatures of its surrounding vertices)
   c. color all the other vertices to be gray $(r = 0.7, g = 0.7, b = 0.7)$.
   d. Display/undisplay these color when the key "k" is pressed.

Hint: to make your program efficient, you can use an array to store your computed properties. For example, in Step 2, you can first compute all the face normal vectors, and store them in a point (namely, 3D vector) array $N_f[0, \dots, F-1]$. In the mesh library, each face f has an index $f -> index()$. You can just store each face f's normal in $N_f[f \rightarrow index()]$. In Step 4, similarly, you may want to use a flag array to store if a boundary edge is visited or not. In Step 5, also store computed curvatures of all the vertices.

Please write all you codes in your “hw2.cpp” file **(do not modify any file from the meshlib)** and send the hw2.cpp to me. I will compile it with the meshlib files on my computer.

Homework Due: 11:59pm, March 10th, 2020.

If you are late for x days, each day you get 10% off. Namely, your homework score will be calculated as:

(the score based on your codes) * (0.9^x)