LAB REPORT: LAB 1

TNM079, MODELING AND ANIMATION

Johnny Elmér
jonel107@student.liu.se

Sunday 10th April, 2022

Abstract

This lab report describes the process used to implement some key components for a half edge mesh structure. More specifically the process of finding all neighbouring faces in the one-ring of a vertex, finding all the neighbouring vertices in the same one-ring, what equations were used for calculating the area and volume of an object and a short mention of Gaussian curvature. The results point towards the process being able to be used to load simple objects but having some flaws when it comes to the colour mapping. A way to fix these issues would be to implement mean curvature, but this was not attempted.

1 Background

A set of three vertices were connected to each other in pairs of half edges, after creating each half edge the so called "inner ring" was connected with the help of the already implemented *prev* and *next* components. Once the edges were connected a $face$ was created and in turn connect to one of the edges, making sure to calculate it's normal vector. The last step consisted of connecting each of the inner edges to the $face$. The main purpose is to get the structure seen in Figure 1.

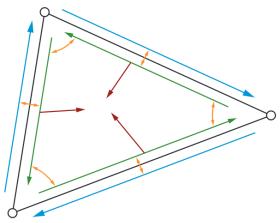

Figure 1: "An incomplete halfedge mesh consisting of one triangle." *Image taken from lab instructions.*

To find all faces in a vertex's one-ring (FindNeighborFaces) in counter clockwise order the *prev* edge and connected face of the wanted vertex was saved. Using a loop, each face connected to edge currently being looked at was stored inside of a vertex before moving onto the previous pair of the current edge. The loop continued until the starting edge was reached.

To find all vertices in the same one-ring (FindNeighborVertices) a similar approach was used. In a loop, the edge connected to the given vertex as well as pointers to the *next* and *prev* edges was used to store the vertex of the *next* edge inside of a vector. Once that vertex was stored the *prev* edges pair was visited and added to the vector, this continued until the starting edge was reached. To calculate the vertex normal the FindNeighborFaces function was called to get a list of all faces in the object before going through the list, summing up each face normal before normalizing the sum using $glm :: normalize$.

The surface area of the mesh was calculated using Equation 1, which was was quite simply implemented using the vertices of each face.

$$A_S = \int SdA \approx \sum_{i \in S} A(f_i) \tag{1}$$

The normal used a slightly more complex calculation, mainly Equation 2. Similarly to the area, this equation was also quite simply implemented exactly as given.

$$3V = \sum_{i \in S} \frac{(v_1 + v_2 + v_3)_{f_i}}{3} \cdot n(f_i)A(f_i) \tag{2}$$

Finally the Gaussian curvature, in the program called VertexCurvature, was implemented. As per instruction, the code for this function was copied from SimpleMesh.cpp and no further changes were made.

2 Results

Figure 2 consists of a unit sphere which was loaded in using the pre-implemented GUI using the half edge mesh option. When loaded the program stated that it consisted of 1984 faces, 2978 edges, and 994 vertices. The area and volume of the sphere, calculated as described in 1, was 12.551 and 4.1519 respectively. As seen in the Figure there are quite a few strange looking issues with how the surface is coloured using the face curvature. This is mostly due to how the face curvature is not good enough at interpreting the surface, which is only amplified by the blockiness of the sphere, leading to strange patterns.

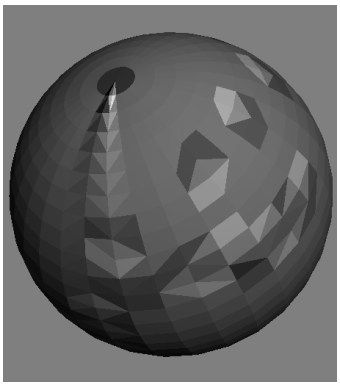

Figure 2: A unit sphere using face curvature for colour mapping.

Figure 3 shows the same unit sphere as the one in Figure 2 but with Gaussian curvature visualized. The curvature of the unit sphere after applying vertex curvature was ranged from 0.252875 to 0.504374. With a correct approximation we would expect to see values closer to one for both the lowest and highest curvature, however, as the sphere is built up of a rather small amount of faces the surface is not as smooth as it could be and the area of each triangle differs greatly depending on where on the sphere you are looking. As the area greatly affects the calculation of the curvature, as the value is divided by the area, the resulting curvature is much lower than what we look for in a more correct representation.

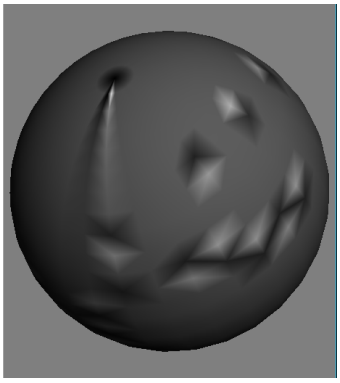

Figure 3: The same unit sphere as in Figure 2 but with Gaussian curvature for colour mapping.

To show that the strange patterns in the colour mapping seen on both previous figures are not the result of wrongly directed normals Figure 4 was created. This figure shows that the direction of each normal is pointed outwards and therefore not the reason for the colouring issue.

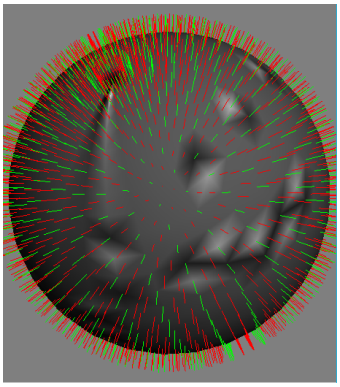

Figure 4: The same unit sphere as in Figure 3 with each triangles normal visualized .

3 Conclusion

Half edge meshes are incredibly useful when used to calculate normals over larger areas, finding the one-ring of a given vertex and when you wish to know which edges are connected to which face, amongst other things. Gaussian curvature is also very useful in creating a smoother looking

surface without having to alter the number of triangles used in the mesh. The main problem however is that the Gaussian curvature (and face curvature) are not complex enough to correctly work with surfaces we want to perceive as round even though it consists of many edges, leading to some strange colour issues. The ”easiest” way to fix this, or at the very least get a better representation, is to implement the mean curvature, but this was not attempted.

4 Lab partner and grade

For the records: My lab partner was Felix Lindgren. All assignments for grade 3 were completed, hence, I should get grade 3.