LAB REPORT: LAB 1

TNM079, MODELING AND ANIMATION

Linnea Bergman
linbe810@student.liu.se

Tuesday 5th June, 2018

Abstract

In this lab a variety of mesh data structures was used and manifolds where classified according to the Euler-Poincar equations. Physical attributes such as normals, curvature, area and volume where inspected and implemented.

1 Introduction

This lab focused on implementing algorithms to represent meshes with triangles. Triangles are the most common format to represent polygon modules. In the lab a half-edge mesh was implemented which had neighbour access and normals, volume and surface area of a mesh was calculated. A half-edge mesh is a common way of representing a mesh and is a quicker method than just drawing each triangle by itself. By implementing a half-edge mesh knowledge of the structure was gained.

2 Assignments

This section contains a detailed explanation to the finished assignments. All function in the following section can be found in HalfEdgeMesh.cpp.

2.1 Assignment Implement the half-edge mesh

The first task was to implement the half-edge mesh. The half-edge mesh is useful for accessing neighbours data. The objective was to implement the functionality in the function AddFace which exists in the file HalfEdgeMesh.cpp. Before implementing AddFace, the structure of the functions AddVertex and AddHalfEdgePair was studied. A face is the enclosed area inside the triangle which consists of three vertices and edges that connects the vertices and triangles to each other. To start with, in AddFace, for each triangle its three vertices was added to the data structure using AddVertex. A half-edge pair was added for each edge in the triangle using the function AddHalfEdgePair.

The next part was to connect the inner ring of edges. This was done by connecting the pointers to the next and previous edge with the respective edge. In order to create the actual face, it had to be connected to one of the edges. The normal vector was calculated and pushed to the face on the mFaces-vector. It was assumed that the mesh is a closed mesh. By connecting the inner ring, every face could access all connecting faces and vertices.

Lastly all edges share the same left face and the edges where then connected to that face. When all of this was done a half-edge mesh could be rendered.

2.2 Assignment Implement neighbour access

Here, the task was to implement the functionality in the two function FindNeighborFaces and FindNeighborVertices. Both functions uses the connection between the edges by the pointers that points to the next and to the previous edge.

In FindNeighborFaces a starting vertex is given and then, by walking along the edges that are connected by the pointers, all the neighbouring faces could be found and added to an array that represented the one-ring of the given vertex. The function then returns an array containing all of the connected faces to the given vertex.

The same principle was used in FindNeighborVertices but instead of saving the faces in an array, all of the vertices connected to the one-ring where saved in the array and returned.

2.3 Assignment Calculate vertex normals

To allow fast calculation of per-vertex normals the function VertexNormal was upgraded. The normal that was returned was a variant called the *mean weighted equally*. This is defined as the normalized sum of the adjacent face normals,

$$\bar{n}_{v_i} = \widehat{\sum_{j \in N_1(i)} \bar{n}_{f_j}} \tag{1}$$

where $N_1(i)$ is the one-ring neighbourhood which is all the faces sharing vertex v_i. $\bar{n_{fj}}$ is the normal of face f_j and $\bar{n_{vi}}$ is the sum of all the faces sharing vertex v_i. Equation 1 was implemented in VertexNormal. To find all the faces sharing the same vertex the function FindNeighborFaces was used. All of the faces normals where added, normalized and then returned.

2.4 Assignment Calculate surface area of a mesh

The surface area of a mesh was calculated in the function Area. The face area for every face in the mesh was calculated and summarized.

$$A_s = \int_s dA \approx \sum_{i \in S} A(f_i) \tag{2}$$

For the normalization Equation 2 was used where A_s is the surface area and $A(f_i)$ is the area of the $i : th$ face. $A(f_i)$ is half of the magnitude of the crossproduct of any two edges in the triangle, seen in Equation 3.

$$A(f_i) = \frac{1}{2}|(v_2 - v_1) \times (v_3 - v_1)| \tag{3}$$

v_1, v_2 and v_3 are the vertices of the face. The function Area returns the sum of all the faces area.

2.5 Assignment Calculate volume of a mesh

To calculate the volume of a mesh the function Volume was used. This function was improved with the implementation of the following equation,

$$3V = \sum_{i \in S} \frac{(\mathbf{v}_1 + \mathbf{v}_2 + \mathbf{v}_3)_{f_i}}{3} \bar{n}(f_i) A(f_i) \tag{4}$$

where $\bar{n}(f_i)$ represent the normal of the i:th face, $A(f_i)$ the area and $(\mathbf{v}_1 + \mathbf{v}_2 + \mathbf{v}_3)_{f_i}$ the vertices. For each face in the mesh the normal and the face area was calculated and then all elements where summarized according to Equation 4. The value was then divided by three and returned. This gives the total volume of the mesh.

2.6 Assignment Implement and visualize Gaussian curvature

The curvature describes the smoothness of the surface and how the normals at a given point changes as the point is moved along the surface. The Gaussian curvature can be composed from two principal curvatures κ_1, κ_2. The following formula was used in the lab for the Gaussian curvature.

$$K = \frac{1}{A}(2\pi - \sum_{j \in N_1(i)} \Theta_j) \tag{5}$$

In Equation 5 A is the area of the one-ring. For the face and vertex curvature the function VertexCurvature and FaceCurvature respectively was improved. The face curvature was calculated by getting the connecting vertices and summarizing their curvatures and dividing the sum with three. In the VertexCurvature the area had to be calculated first. This was done by choosing the Voronoi area, Equation 7, of the neighbourhood.

$$A_v = \frac{1}{8} \sum_{j \in N_1} (\cot \alpha_j + \beta_j)|(v_i - v_j)|^2 \tag{6}$$

α_j and β_j are the angles on the opposite sides to to given vertex, v_i the chosen vertex and v_j the vertices connected in the one-ring. After the area was calculated the curvature was computed with Equation 7 and returned.

2.7 Assignment Implement and visualize mean curvature

The implementation of the mean curvature in VertexCurvature was similar to the Gaussian curvature except that another formula was used.

$$H_n = \frac{1}{4A} \sum_{j \in N_1} (\cot \alpha_j - \cot \beta_j)(v_i - v_j) \tag{7}$$

This gives the mean curvature for the given vertex v_i to all vertices in the one-ring. The area was calculated using Equation 7 and H_n was the return value of the function.

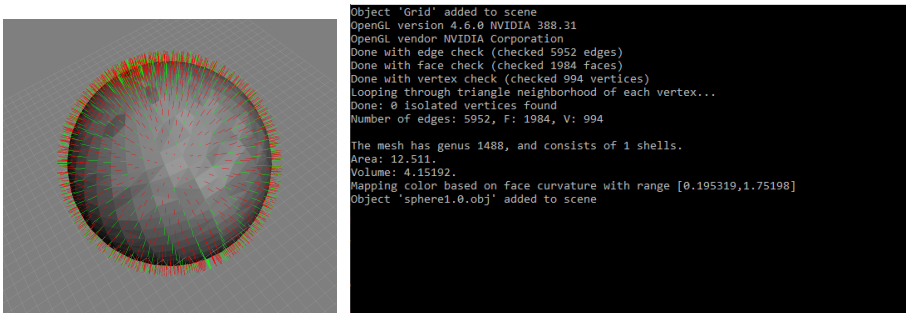
2.8 Assignment Classify the genus of a mesh

The function Genus in HalfEdgeMesh.cpp was modified for the classification of genus of a mesh. Equation 8 is used when computing the genus of a mesh when working with only one shell.

$$V - E + F - 2(1 - G) = 0 \tag{8}$$

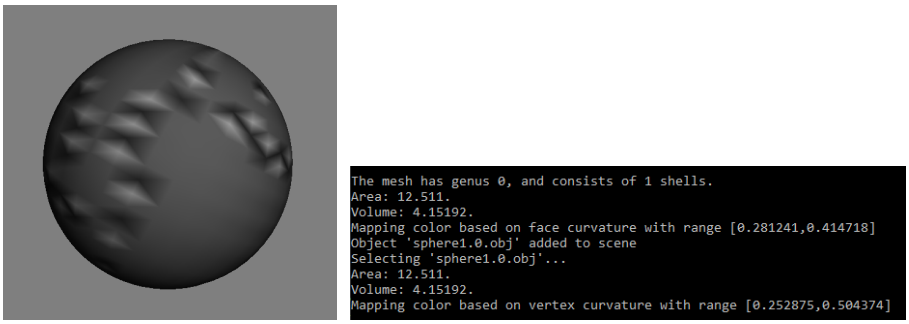
It was assumed that the mesh was a closed manifold mesh, for simplification. In Equation 8 V represents the number of vertices, E the number of edges, F the number of faces and G the genus. The equations was rewritten as to have the genus on one side and everything else on the other. In the function Genus all values exists as the length of the respective array. The genus was calculated and returned.

3 Results



(a) Half Edge Mesh (b) Half Edge Mesh Output

Figure 1
Result from assignment 2.1 showing the half-edge mesh with the computed normals 1(a) and the output 1(b). In the output the genus, volume and the area of the mesh are displayed.

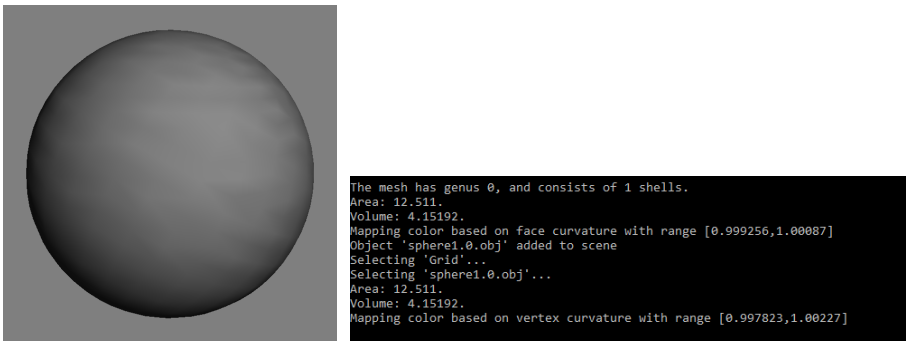


(a) Gaussian Curvature (b) Output Gaussian Curvature

Figure 2
Result from assignment 2.6 showing the Gaussian version of the face curvature (left) and the output (right).

In Figure 1 the half-edge mesh can be seen along with the normals 1(a) to the left. To the right of the Figure 1(b), the output of the mesh are displayed along with its genus, volume and area.

Figure 2 shows the result of the Gaussian curvature 2(a) and the output 2(b). The pattern in 2(a) comes from the Gaussian curvature. The interesting part of the output 2(b) is the interval of the colour mapping of the face curvature.



(a) Standard rendering (b) Fancy rendering

Figure 3
Result from assignment 2.7 showing the result of the mesh curvature (left) and the output with the range curvature intervals stood (right).

Figure 3 displays the mean curvature, with the result of the curvature to the left and the face curvature range in the output to the right.

4 Conclusion

The diameter of the mesh, which was a sphere, in Figure 1 was 1 and the value of the area and volume matched with the expected value for a sphere with diameter 1.

When comparing the time it took to render a half-edge mesh to other methods (such as simple mesh which renders the triangles separately) it was a lot faster. The other assignments helped understanding how the half-edge mesh works and how it is composed.

It can clearly be seen that the mean curvature gives a better smoothing effect than the Gaussian when comparing the mesh in Figure 2 and Figure 3. The output also shows that the mean curvature have an interval closer to 1 than the Gaussian curvature have, which indicates that the mean curvature is a better approximation. Though it is a bit more complex to implement.

5 Lab partner and grade

This lab was done together with Rebecca Cedermalm, rebca973. The assignments that was done was all tasks for getting the grade 4, which includes implementing and visualizing the mean curvature and classifying the genus of a mesh.