LAB REPORT: LAB 1

TNM079, MODELING AND ANIMATION

Julian Voith
julvo429@student.liu.se

Wednesday 5$^{\text{th}}$ April, 2023

Abstract

This lab report describes the tasks and results from lab 1 of the course TNM079, Modeling and Animation at Linköping University. The lab explained the theory and implementation of the half-edge mesh data structure which uses adjacency information to efficiently calculate different physical attributes like the area, volume, genus, vertex and face normals, as well as the curvature and the number of shells of 3D objects. The results show that the implementation of the half-edge mesh is more complex to establish compared to a simple mesh structure but gives new ways of accessing and processing meshes, which especially operations accessing neighboring information benefit from.

1 Background

The most straight forward way to create a data structure for triangles is to create a unique list of all vertices and faces. In this type of data structure no neighboring information (e.g. adjacent triangles) of a vertex are included. This is acceptable for operations executing a linear traversal through triangles like rendering. But once operations depending on neighboring information like identifying adjacent faces of a vertex are performed, those quickly turn into expensive operations because the whole face list needs to be iterated each time.

A way to create a more efficient way is to store adjacency information is the *half-edge data structure*. The name is defined by the way the information of a triangle is stored. Each edge of a triangle is ”split” down its length explicitly storing only the left face. Information of the adjacent triangle can be accessed via its pair. It becomes more clear by taking a closer look on the components of the data structure:

- **Vertex:** Contains the x-y-z coordinates of the vertex as a float and a pointer to its half-edge.
- **Half-edge:** Contains several pointers. One for the vertex of the edge, one for the next and previous half-edge of the triangle, one for the pair of the half edge which belongs to the neighboring triangle, and one for the face it is part of.
- **Face:** The Face stores a pointer to one of the half-edges of the face. The other half-edges of the face can be accessed via next and prev attribute of the half-edge.
- **Mesh:** Contains a lists of all structures listed above. Lists objects are uniquely stored.

For some operation all adjacent faces and their vertices of a vertex are needed. This is the *1-ring* of a vertex. To achieve this, the half-edge of the vertex is fetched. Then the *1-ring* is traversed **counter clock-wise** by using the *previous pointer* and *pair pointer* within the structure, until we reach our starting half-edge/vertex again. An illustration of the traversal can be found in figure 1. Depending on the operation, traversed faces or vertices are stored in a list through-out the traversal.

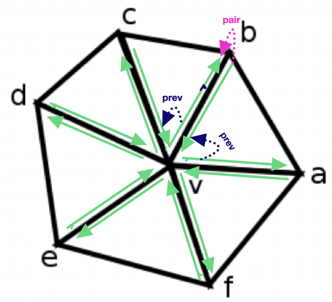


Figure 1: Illustration of the traversal of the neighboring faces and vertices. The green arrows represent the half-edges and the dotted arrows the pointers which are used for traversal.

Accessing neighboring information enhances the calculation of the *vertex normals* and *face normals*. There are different ways to calculate *vertex normals* but for the lab the *mean weighted equally (MWE)* variant was used. This variant simply **sums up** the normalized face normals of the surrounding faces. For this calculation we fetch all adjacent faces by traversing the *1-ring* of the starting vertex. The normal of a face is calculated by the cross product of the terms in equation 1 and is then being normalized by its magnitude.

$$n = (v_2 - v_1) \times (v_3 - v_1); \hat{n} = \frac{n}{||n||} \tag{1}$$

The adjacent information in the data structure is also useful to calculate the area of a 3D object. The overall area of the object is the sum of the areas of all single faces given by formula 2.

$$A_s = \sum_i A(f_i) \tag{2}$$

Hence, the area for each face is calculated by the given formula 3.

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

Once the normals and areas of a face are calculated. The volume of an object can be calculated by the given formula 4.

$$V = \frac{1}{3} \times \sum_{i \in S} \frac{(v_1 + v_2 + v_3)_{fi}}{3} \times n(f_i) \times A(f_i) \tag{4}$$

where v1, v2 and v3 are the vertices of the i:th triangle and calculates the inside mid point of the face f_i, n is the normal of the face, and A the area of the given face.

Another physical attribute is the *curvature* of a mesh. It is a quality measure for meshes and is frequently used in algorithms. The attribute describes the smoothness of the surface and how

the normal at a specific point behaves once its moved along the surface. A 3D unit sphere should have a curvature **around 1**. There are two most used types of curvatures: *Gaussian curvature* and the *mean curvature*. These calculations use angles of the triangles in the mesh which can be found in figure 2. The equations 5 and 6 are used.
Gaussian curvature

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

where A_n is the area of the adjacent faces and the angle of the current face.
Mean curvature

$$Hn = \frac{1}{4A} \sum_{j \in N_1(i)} (\cot \alpha_j + \cot \beta_j)(v_i - v_j). \tag{6}$$

Also here the A is the total area of the 1-ring neighborhood. Both equations can be improved by choosing the *Voronoi* area formula, which can be found in following equation:

$$A_v = \frac{1}{8} \sum_{j \in N_1(i)} (\cot \alpha_j + \cot \beta_j)|(v_i - v_j)|^2 \tag{7}$$

The *genus* is a quantitative attribute which describes the topological holes of a surface. To calculate the genus, the Euler-Poincaré-Formular can be reformulated to G:

$$G = S - \frac{V - E - L + 2F}{2} \tag{8}$$

where S is the number of shells, V the number of vertices, E the number of edges, L the number of loops, and F the number of Faces. The number of edges in a half-edge mesh are half as many as the number of half-edges. As long as the mesh has no holes, a triangle mesh has exactly one loop per triangle face which results into L=F.

The number of shells, S, of a mesh can be found by starting at a vertex and "tagging" all vertices that can be reached. If unvisited vertices remain, a new starting point of of the unvisited vertices is chosen and the iteration restarts. This is done until all vertices has been "tagged" or visited. The number of shells is then given by the number of times the iteration had to be restarted which also includes the first one.

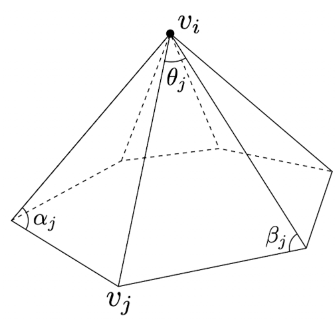


Figure 2: Illustration of the used angles in curvature calculations

2 Results

The following results will be shown based on different sample objects which were provided for the lab.

2.1 Vertex and face normals

In the program given for the lab, the normals are visualized as green and red lines. Green lines represent the vertex normals and red lines represent the face normals. In figure 3 can be observed that the *cube* consists of 8 vertices and 12 faces.

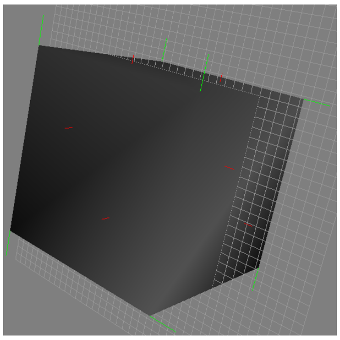


Figure 3: Demonstration of the red face and green vertex normal calculations of a cube.

2.2 Area and volume calculations

The area and volume of a sphere are well-known and were provided in the lab document. In the table 1 a comparison between the approximations done by the calculation and the real volume and area values of a sphere with different radii can be found.

Area and volume calculations					
3D object	Radius	Calc. area	Calc. volume	Exact Area	Exact volume
Sphere 0.1	0.1	0.12511	0.0041519	0.12566	0.00418879
Sphere 1.0	1.0	12.511	4.15191	12.56637	4.18879

Table 1: Results of the area and volume calculation for spheres with radius 0.1 and 1.0.

2.3 Curvature calculation

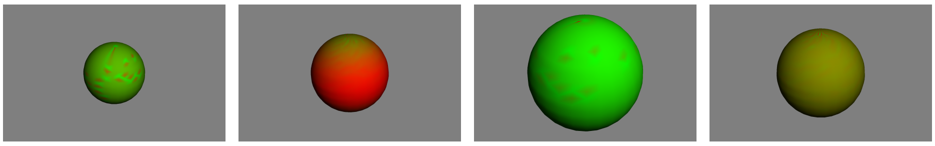
A visual comparison between the two different curvature approximations can be found in figure 4. The minimum and maximum curvature approximation/calculations can be found in table 2.

2.4 Shell and genus calculation

The genus and the shells can be calculated for every existing provided reference objects. In the table 3 the results for multiple examples can be found.

Curvature approximation by variant		
Curv. calculation	Minimum	Maximum
Gaussian	0.252891	0.504374
Gaussian with Voronoi	0.996891	1.00619
Mean	0.128677	1.86172
Mean with Voronoi	0.997823	1.00227

Table 2: Results of the curvature calculations for a sphere with radius 1.0.



(a) Gaussian curvature (b) Gaussian w. Voronoi (c) Mean curvature (d) Mean w. Voronoi

Figure 4: Visualization of the result from different variants of the curvature calculations. It shows each method with and without Voronoi for the area calculation.

Shell and genus calculations				
3D object	Vertices	Faces/Loops	Shells	Genus
Cube	8	12	1	0
Genus cube	46	108	1	5
Genus test	2013	4022	4	3

Table 3: Results of the genus calculation for the cube, genus cube, and genus test object implemented in the lab.

3 Conclusion

Overall the half-edge mesh data structure is more complex to construct and memory intense, but it enables efficiency for operations using neighboring information.

The efficiency of the surface normal calculations can be observed by comparing the initial code from the *SimpleMesh* and from the newly created *half-edge mesh* for calculating the vertex normals. In the simple variant a full search through all faces for all vertices were needed. In the implemented half-edge mesh variant this is done by simply traversing through each 1-ring of the vertices by using a few pointers. This already suggest a performance increase.

The calculation of the surface area and volume of the mesh are approximations and slightly differ from the exact values of a perfect sphere. This is due to their composition of triangles and the way it is calculated. However, the values are quite close to the exact values.

The surface curvature calculations combined with the initial area calculations does a poor job of estimating the curvature. By using the Voronoi area to partition the mesh into closed patches, the estimation comes close to the curvature of a perfect sphere which is 1.

The Results of the Genus and shell calculation were as expected.

4 Lab partner and grade

The lab and all tasks were solved together with William Toft(wilto938). This report aims for grade 5.