LAB REPORT: LAB 1

TNM079, MODELING AND ANIMATION

Viktor Sjögren
viksj950@student.liu.se

Sunday 4th April, 2021

Abstract

In this short paper a brief discussion of the half edge mesh data structure for representing objects in 3D space is presented. This mesh method allows for more efficient traversal to adjacent faces, by storing the edges of the faces as two half edges, with opposite directions. With this mesh data structure, ways of identifying properties such as area, volume and curvature has also been implemented. The calculations for the area and volume have then been analyzed to grasp how close the formulas used is to the real values. Two different objects have also been looked at to see the clear difference between curvature methods.

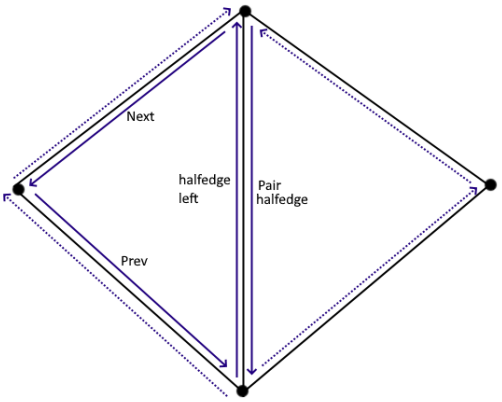

Figure 1: Visual representation of the half edge mesh

1 Introduction

This lab was done in order to implement and improve the way a mesh can be traversed, which in this specific case involves the half edge data structure. This is an important feature to understand because it can improve different applications where efficiency is key.

2 Background

The half edge data structure is one of the many ways of representing polygonal objects and could be described as an modification to the traditional simple triangle mesh method. The main goal of the modification is to make information about adjacent faces an easier process to get access to. This is done through describing each edge of a triangle as two separate halves, and storing information about the left half edge for each triangle. The corresponding half edge can then be accessed through its pair variable which points to the half edge to the neighbouring triangle. The operation to find all adjacent triangles can then be done in less time then the $O(n^2)$ time complexity that comes with a traditional indexed face set or independent face list. In Figure 1 a simple sketch of the half edge principle can be seen.

3 Lab tasks

In this section each task is presented and the theory behind the implementation is discussed. Manifold-meshes was assumed dur-

ing all implementation of the tasks. Every task was implemented in the class *HalfEdgeMesh.cpp*.

3.1 Creating the half edge mesh

The initial goal was to complete the half edge mesh implementation, which involved implementing the *AddFace* function. The function first adds the tree vertices that spans the face using the *AddVertex* function. *AddVertex* also ensures only unique values are placed, and only returns the index in cases of shared vertices. Then by using the function *AddEdgeHalfPair*, the pairs for each edge is generated using the vertices from the current triangle. Each vertex is assigned to one of the edges. With the edges being represented as half edges, the *previous* and *next* are then set up in a counter clockwise rotation, creating an inner loop.

With the configured vertices and half edges, the face is then created with the corresponding inner edges along with its normalized normal pointing out from the face. Only inner loops are relevant here as manifold meshes are the target, making all loops once the faces have been created inner loops.

3.2 Neighboring faces access

If the only information that exists about normals is that each triangle has one normal vector perpendicular to itself, the results would be a flat shaded mesh. In order to improve the shading model, per vertex normals can be calculated and used instead. This makes the more efficient neighborhood access operation of the half edge mesh useful.

The implementation done in this Lab builds on the mean weighted equally algorithm. Simply put, the algorithm sums the normal for all the adjacent faces to the specific vertex, creating a so called 1-ring neighborhood.

$$n_{v_i} = \sum_{j \in N_1(i)}^{n} n_{f_j} \tag{1}$$

n_{f_j} is the normal vector for each triangle, and $N_1(i)^n$ is the 1-ring neighborhood around v_i. This will be the same for all vertices, again with the reasoning being that the targets are closed meshes.

The implementation for obtaining the information was done in the functions *FindNeighborFaces* and *FindNeighborVertices*. Both functions work in the same way, with the difference being that the vertex indices are returned in *FindNeighborVertices* and face indices in *FindNeighborFaces*. The 1-ring is traversed using the pair properties for the edges in the triangles.

3.3 Per vertex normal calculation

With the functions in the last step retrieving the information of the adjacent faces, the equation 1 can be used to compute the per vertex normal. The code was implemented in the function *VertexNormal*, where the principle of the equation was mimicked.

3.4 Surface area

Computation of the total surface area was done in the function *Area*. With the mesh being represented as a shell of faces, each face can be considered a partial area of the whole mesh. By summing all of the faces together, the total surface area can be calculated. Based on the Riemann sum theorem, the equation used is as follows

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

where each face area is represented by $A(f_i)$. As the faces are triangles, the area is obtained as half of the total area of the cross product of two edges in the triangle. The formula used in the code looked like the following equation.

$$A(f_i) = |(v2 - v1) \times (v3 - v1)| \tag{3}$$

3.5 Volume calculation

The basis of the volume calculation done in this lab comes from the divergence theorem.

Briefly put, the theorem can compute the connection between the surface integrals and the volume. This makes the previous area calculations even more useful as it provides an important variable to compute volume using the divergence theorem.

The formula used to compute the volume in the function *Volume* is the following,

$$3V = \sum_{i \in S} \frac{(v1 + v2 + v3) f_i}{3} \cdot n(f_i) A(f_i) \quad (4)$$

where f_i indicates which face, $A(f_i)$ is the surface area as described previously. Dividing by three is done to get the centroid of the triangle, but any point on the face would suffice.

3.6 Gaussian curvature

A measure of smoothness of the surface was implemented in the lab, namely Gaussian curvature and mean curvature. This section will however only cover Gaussian curvature.

The principal of curvature builds on the fact that a point p with its corresponding normal changes as the point moves over the surface. The specific curvature value can then be obtained through the inversely proportional property to an oscillating circle's radius.

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

By using the Gaussian curvature formula, as seen in equation 5, we get K which represents the Gaussian curvature. The value of this can be used to determine surface smoothness by the indications of that a positive value shows that the surface is pointy, equal to zero means that the surface is flat. As can be seen by the sum in equation 5, the formula looks at the 1-ring neighbourhood.

4 Results

The different curvature visualisation methods were analysed and the final results can be seen in Figure 2.

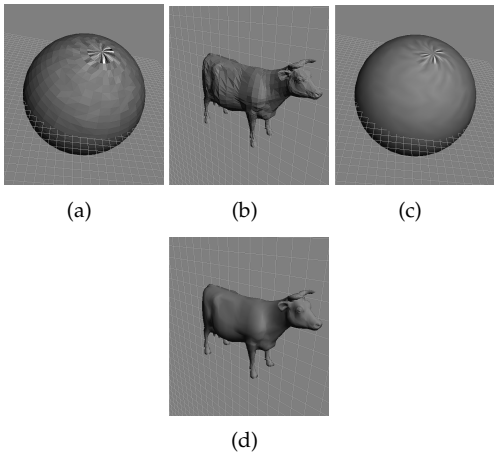

Figure 2: Result from the curvature task, by adding Gaussian curvature which computes the per vertex normals we obtain better visual shading for the half edge mesh models. The face curvature method can be seen in (a) and (b), while the improved per vertex method is seen in (c) and (d).

When implementing the area and volume computations, spheres were used as reference geometries to validate the results. In Figure 3 the two spheres tested with can be seen.

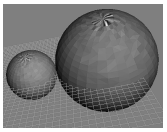

Figure 3: Result from computing the area and volume for two spheres gave close values to the analytical values. The left sphere is of radius 0.5, while the right sphere is of radius 1. The analytical values for the volume respectively would be 0.5236 and 4.1888. The value of the area would be 3.1416 and 12.5664 respectively.

The computed area values were obtained in equation 2 and the volume obtained from equation 4. The obtained values for the sphere with radius 1 were 12.5110 and 4.1519 for the area and volume respectively. Sphere with radius 0.5 resulted in the values 3.1278 and 0.5189, for area and volume respectively. This means the error range with this limited quality test would be ~ 0.5. However the values show that the error is higher for the larger

sphere, making the error margin lower if only the sphere radius 0.5 is regarded.

5 Lab partner and grade

The lab was made in collaboration with Algot Sandahl. The report scope and content is aimed at grade **3**.

References

[1] M.E. Dieckmann. *Lecture 1 & 2* , tnm079, 2021.