LAB REPORT: LAB 1

TNM079, MODELING AND ANIMATION

Lucas Hägg
lucha567@student.liu.se

Tuesday 7th May, 2024

Abstract

In this lab, an exploration, implementation, and analysis of mesh data structures were conducted with a focus on the half-edge mesh due to its efficiency in accessing neighbor data, essential for calculating vertex normals and other geometric properties. By analyzing physical attributes such as normals, curvature, area, and volume, and implementing discrete formulas for these properties, insights were gained into the trade-offs between memory usage and computational efficiency in mesh data structures. The findings from this lab underline the importance of choosing an appropriate data structure for modeling and animation tasks in computer graphics, balancing between storage space and performance for neighborhood searches.

1 Introduction

The lab focuses on understanding and implementing functions for the half-edge mesh data structure. The primary motivation behind this exploration is to address the computational and memory inefficiencies encountered in simple mesh data structures when dealing with neighborhood data. Triangles, being the smallest entities to span a plane, serve as the basic unit for the mesh representation, offering ease of manipulation.

2 Background

A mesh data structure is a collection of vertices, edges, and faces that defines the shape of a 3D object. The efficiency of operations on these meshes is of paramount importance for many applications, and the half-edge data structure is central to this efficiency.

2.1 Half-edge Data Structure

The implementation of the half-edge data structure is central to this lab. This enhanced representation facilitates efficient navigation and manipulation of mesh topology. The half-edge structure introduces the concept of half-edges—edges split into two halves, each directed and associated with a face. This enables quick access to neighboring vertices and faces, making it an optimal choice for operations that require neighborhood data. The term ”inner ring” refers to a face connected by half-edges, while ”1-ring” denotes every neighboring vertex connected by the half-edge structure.

2.2 Area and Volume Calculation

In addition to curvature, methods to compute the area and volume of mesh objects were explored. The area is straightforward, as it is the sum of the areas of individual faces. The volume computation, employs the divergence theorem to relate a volume integral to a surface integral, allowing the calculation of the volume enclosed by a mesh based on its surface properties.

3 Tasks

This chapter goes through the tasks implemented during the lab.

3.1 Implementing the Half-edge Mesh

The implementation of **AddFace** involved adding vertices to the mesh, creating half-edge pairs for each edge of the triangle, connecting these half-edges to form the inner loop, and finally calculating and assigning the face normal. This approach ensures the efficient representation of the mesh topology and allows for quick access to adjacent elements, which is crucial for operations that require neighborhood information.

3.2 Find neighbor faces

The function **FindNeighborFaces** retrieves the indices of all the faces neighboring a specified vertex, sorted counterclockwise. The function iterates through the half-edges connected to the vertex, following pairs of half-edges to find adjacent faces.

3.3 Find neighbor vertices

The function FindNeighborVertices is implemented in a similar way as for finding neighbor faces. It loops over the neighborhood of a given vertex and returns the indices of all the neighboring vertices sorted counterclockwise. It utilizes the half-edge mesh data structure by following pairs of half-edges.

3.4 Calculating Vertex Normals

The **VertexNormal** function calculates the normals for each vertex by averaging the normals of adjacent faces. This method leverages the neighborhood information efficiently provided by the half-edge structure to compute smoothed normals, which are essential for rendering and lighting calculations.

$$\mathbf{n_v} = \frac{\sum_{f_i \in N(v)} \mathbf{n}_{f_i}}{|\sum_{f_i \in N(v)} \mathbf{n}_{f_i}|} \tag{1}$$

N denotes the set of adjacent faces and n denotes the normal of the face f.

3.5 Area and Volume Calculation

The **Area** and **Volume** methods implement discrete formulas to compute the mesh's surface area and volume.

$$A = \frac{1}{2} \sum_{f_i} |\mathbf{v}_2 - \mathbf{v}_1 \times \mathbf{v}_3 - \mathbf{v}_1| \tag{2}$$

A is the area of the face, n is the normal vector of the face, and v1, v2, v3 represent the vertices of the face.

$$V = \frac{1}{9} \sum_{f_i} (\mathbf{v}_1 + \mathbf{v}_2 + \mathbf{v}_3) \cdot \mathbf{n}_{f_i} \cdot A_{f_i} \tag{3}$$

V is the volume. Vertices v1, v2, v3 is the vertices for the face at index i. A is the area of the face and n is the normal of the face at index i. The sum goes through all faces of the half-edge object.

3.6 Vertex Curvature

The Vertex curvature function computes the Gaussian curvature of a vertex by summing the angles between neighboring edges and dividing by the area of the surrounding region. This measure indicates how sharply the surface bends at the vertex.

$$K_v = \frac{2\pi - \sum_j \theta_j}{A_v} \tag{4}$$

K is the curvature for a vertex v. Theta is the angle between consecutive edges j connected to the vertex, and A is the area

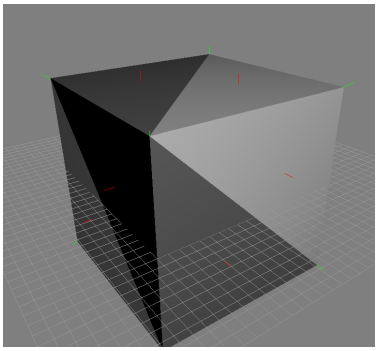

Figure 1: Figure 1 displays the calculated vertex normals for a triangular mesh, as implemented in the VertexNormal function. The mesh is represented as a collection of connected triangles, with the normals indicated by line segments. The uniformity of the normals across the mesh is a testament to the accurate implementation of the algorithm, suggesting a smooth surface orientation integral for lighting calculations in computer graphics.

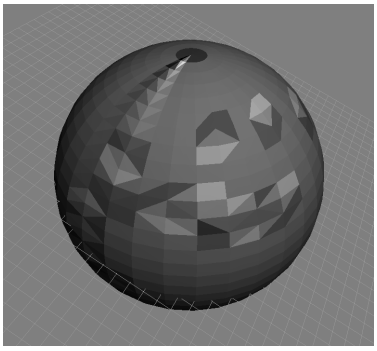

Figure 2: Figure 2 presents the computed vertex curvature of our mesh, as derived from the VertexCurvature function. This figure employs a grayscale color map where lighter shades indicate higher curvature values, and darker shades represent flatter areas. The curvature at each vertex is calculated by using the VertexCurvature function, which applies a discrete form of the Gaussian curvature formula. The code first identifies the 1-ring neighbors of a vertex and then sums the angles formed between adjacent vertices, scaled by the area of the surrounding faces.

4 Results

For the tasks of implementing area and volume functions, the result is demonstrated using a half-mesh sphere built on 994 vertices.

Function	Result	Real value
Area	12.511	12.566
Volume	4.151	4.188

Table 1: Area and volume of a sphere (994 vertices)

5 Conclusion

The lab has illustrated the advantages of the half-edge mesh data structure in modeling and animation tasks. Through the implementation of this structure and the associated geometric calculations, it was demonstrated how it enables efficient access to mesh topology and geometry, which helps with performing complex operations such as calculating normals, curvature, area, and volume.

6 Lab partner and grade

Lab partner was Kenton Larsson and the grade I am for is 3.

References

[1] Mark Eric Dieckmann. *MESH DATA STRUCTURES*. Linkoping university, 2023.