# LAB REPORT: LAB 1

MESH DATA STRUCTURES

TNM079 - MODELING AND ANIMATION

Emma Forsling Parborg

emmfo731@student.liu.se

Monday 18th January, 2016

**Abstract**

This report describes the first out of six labs in the course TNM079 - Modeling and Animation given at Linköping University. This lab deals with the creation of a half-edge mesh structure and studying which benefits it can have. For this, different basic mesh calculations and operations were implemented, such as area and volume calculations and vertex, normal and curvature computations. The resulting half-edge data structure features quick neighbor access which illustrates the advantages in using this structure instead of other structures, such as indexed face lists.

## 1 Introduction

The purpose of this lab is to demonstrate the advantages of the half-edge mesh data structure. When comparing this data structure to others, e.g. *indexed face list* the half-edge mesh data structure allows quicker neighborhood access. This is essential when performing decimation or subdivision, which will be implemented in following labs.

## 2 Assignments

This section describes the assignments that were completed in order to create a half-edge mesh.

### 2.1 Implement the half-edge mesh

When loading an object, consisting of triangles, the function `AddFace` is called for each triangle, with the vertices as input argument. In this function, the vertices are added to a vertex list, by the function `AddVertex`. To make sure that no vertices are stored more than once, this function iterates through the vertex list for each new vertex. If it exists, it returns an index which represents the position the vertex has in the vertex list. If it doesn't exist, it adds it to the vertex list and returns that index as well. These indices are then used when the half-edge pairs are to be created. This is done by calling the `AddHalfEdgePair` for each edge. This function creates two half-edges with opposite directions between two vertices, representing an inner edge and an outer edge. It also sets the origin for each edge to a face vertex, which in turn is connected to the edge. When these connections have been made, it then stores the edges in an edge list.

After all edges have been created, the last thing to do, when creating a face, is to connect the inner ring of edges. This is done by connecting the edges by their `previous` and `next` pointers. In this step one can also define the outer loop of edges. However, if one assumes that the mesh is closed, then the inner loop will suffice. An illustration of what a half-edge mesh data structure looks like from a current edge is visualized in Figure 1.

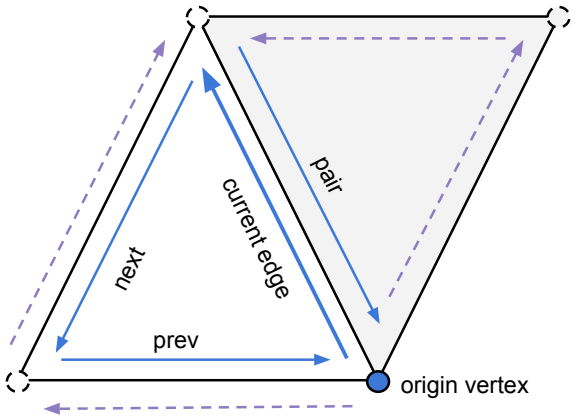


*Figure 1:* The half-edge data structure as seen from the current edge, represented by a bold blue line. The blue dot represents the origin vertex. The pointers `next` and `previous`, for the current edge, are represented with thin blue lines. The white triangle represents the current face, and the gray triangle a neighboring triangle. The pair for the current edge is also represented with a thin blue line.

## 2.2 Implement neighbor access

To implement neighbor access, the functions `FindNeighborFaces` and `FindNeighborVertices` were used which returns all neighboring vertices and faces, respectively. These functions are quite similar, and traverse the half-edges of the data structure and store the desired information in arrays.

To access all neighbors of a face, the data structure is traversed by utilizing the `previous` and `pair` pointers. Starting from the origin vertex, the first neighbor is found by using its previous pointer and then its pair. This procedure is then repeated until the origin vertex is reached again, see Figure 2.

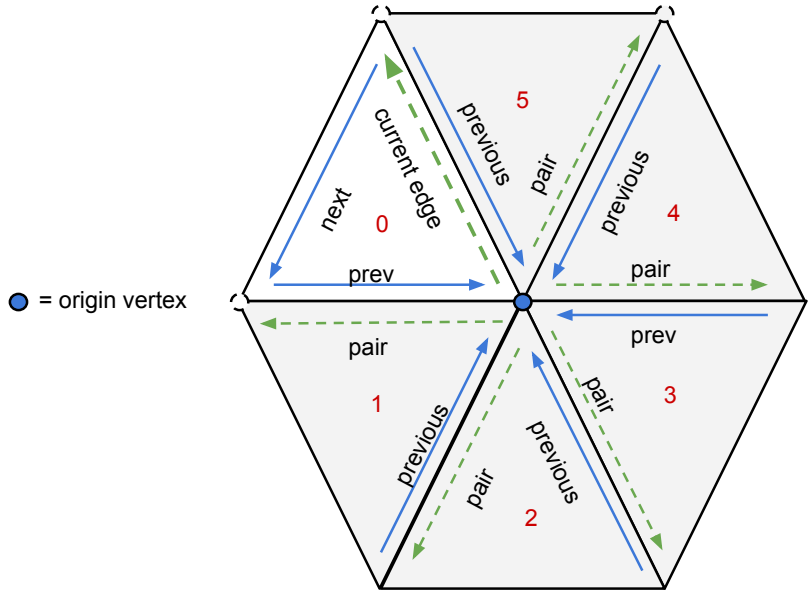


*Figure 2:* This figure illustrates how the half-edge data structure is traversed when determining neighbors. The red numbers represent the order in which neighbors are found with the exception of 0 which is the current face.

## 2.3 Calculate vertex normals

To calculate the vertex normals, Equation 1 was utilized:

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

Here, the normal $\mathbf{n}_{v_i}$ is calculated as a mean value of the surrounding face normals $\mathbf{n}_{f_j}$.

## 2.4 Calculate surface area of a mesh

To calculate the surface of an area, the *Riemann sum* approximation has been utilized:

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

The surface area of a mesh $A_S$ is calculated as a sum of the area of all faces $A(f_i)$. For this lab, the faces are triangles, and therefore the area for a face can be calculated as half the magnitude of the cross product of two edge vectors.

## 2.5 Calculate volume of a mesh

The volume of a mesh is calculated using the *divergence theorem*, which is also known as *Gauss' theorem*. This theorem expresses the volume integral of the divergence of a vector field as the

surface area integral of the vector field times its surface normal.

The surface area integral is then approximated as a Riemann sum over each face, which is displayed in Equation 3.

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

Here, $\mathbf{n}(f_i)$ and $A(f_i)$ denote the normal and area for the face $i$ and $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$ represent the vector field which is evaluated at the centroid position of that face.

## 2.6 Implement and visualize Gaussian curvature

Curvature describes the smoothness a surface. In this assignment Gaussian curvature was implemented to visualize the smoothness of a surface, see Equation 4.

$$K = \frac{1}{A} \left( 2\pi - \sum_{j \in N_1(i)} \theta_j \right). \tag{4}$$

Here, the deviation of $2\pi$ is weighted by the area of the 1-ring neighborhood of a vertex, see Figure 3. To calculate this, the neighbor access is utilized in order to loop through all vertices and sum their angles, $\theta_j$, at the current vertex position. The area, $A$, is calculated as half the magnitude of the cross product of two edge vectors.

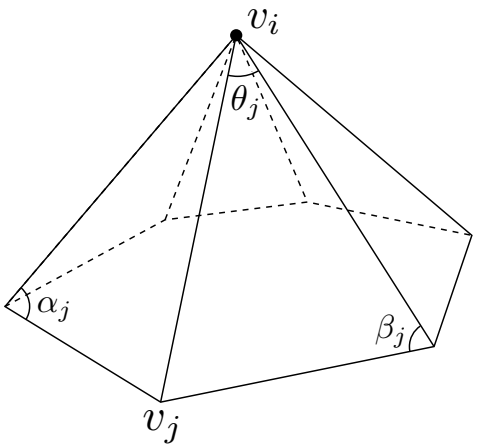


*Figure 3:* This figure illustrates the angles needed in order to calculate the Gaussian curvature of a vertex $v_i$.

## 2.7 Implement and visualize mean curvature

In this assignment, mean curvature was implemented. This method allows both negative and positive values which enables it to distinguish between convex and concave surfaces. In Equation 5, the mean curvature is calculated.

$$H\mathbf{n} = \frac{1}{4A} \sum_{j \in N_1(i)} (\cot \alpha_j + \cot \beta_j)(\mathbf{v}_i - \mathbf{v}_j). \tag{5}$$

The $A$ represents the area of the 1-ring neighboorhood. The angles $\alpha_j$, $\beta_j$ are illustrated in Figure 3, as well as the vertices $\mathbf{v}_i$ and $\mathbf{v}_j$. By replacing $A$ with the Voroni area of the neighborhood $A_v$, the accuracy for the mean curvature can be improved, see Equation 6.

$$A_v = \frac{1}{8} \sum_{j \in N_1(i)} (\cot \alpha_j + \cot \beta_j)|(\mathbf{v}_i - \mathbf{v}_j)|^2 \tag{6}$$

### 2.8 Classify the genus of a mesh

The genus of a mesh is determined by how many holes the mesh has, and can be calculated using the Euler-Pointcaré formula, see Equation 7.

$$V - E + F - (L - F) - 2(S - G) = 0 \tag{7}$$

Here, $L$ represents the number of loops, $S$ the number of shells, $G$ the genus, $V$, $E$ and $F$ the number of vertices, edges and faces, respectively. For this lab, the shell $S$ is set to one. To determine the genus, the equation can be rewritten to:

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

### 2.9 Extend half-edge mesh to handle non-closed surfaces

For non-closed meshes, connected boundary edges exist in order to maintain the half-edge data structure. These boundary edges do not have any faces connected to them and therefore no `next` or `previous` pointers either. In this assignment the outer loop of edges is defined.

In the lab, the boundary edges are connected after all faces of the mesh are initialized. This is performed by using two loops: one for the vertex of the current edge and one for its `pair` edge. When an edge with the face pointer `uninitialized` is encountered, meaning that it is a boundary edge, its `next` and `previous` pointers have to be set. This can be done by connecting them to other boundary edges on the current face or neighboring faces.

## 3 Results

The important results for each of the assignments are presented in this section.

### 3.1 Vertex normals

Visualizing the vertex normals with the program, yielded the following result, see Figure 4.

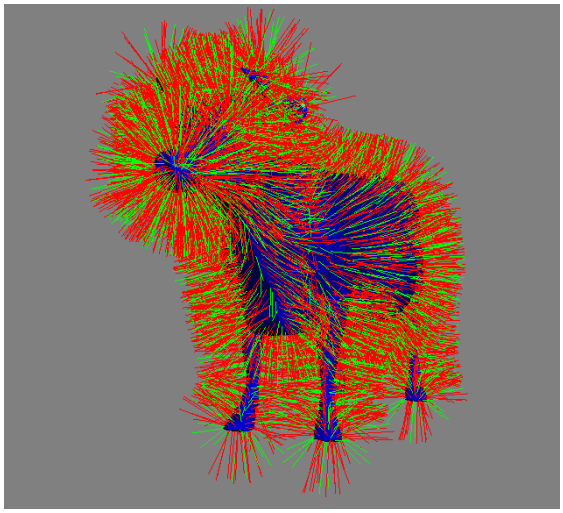

*Figure 4:* This figure illustrates the mesh cow.obj with its vertex normals visible. The color map Jet is used on the mesh and the vertex normals are colored red and green.

### 3.2 Surface area of a mesh

The results from calculating the surface area for a sphere, of varying size, are displayed in Table 1.

| Sphere radius | Calculated area | Analytical area |
|---|---|---|
| 0.1 | 0.12511 | 0.12566 |
| 0.5 | 3.12775 | 3.14159 |
| 1.0 | 12.5110 | 12.5664 |

*Table 1:* This table illustrates the area of a sphere, the analytical area and the calculated area.

### 3.3 Volume of a mesh

The results from calculating the surface volume for a sphere, of varying size, are displayed in Table 2.

| Sphere radius | Calculated volume | Analytical volume |
|---|---|---|
| 0.1 | 0.0041519 | 0.0041888 |
| 0.5 | 0.5189900 | 0.5235988 |
| 1.0 | 4.1519200 | 4.1887902 |

*Table 2:* This table illustrates the volume of a sphere, the analytical area and the calculated area.

### 3.4 Gaussian curvature

In Figure 5 the results from implementing Gaussian curvature are visible.

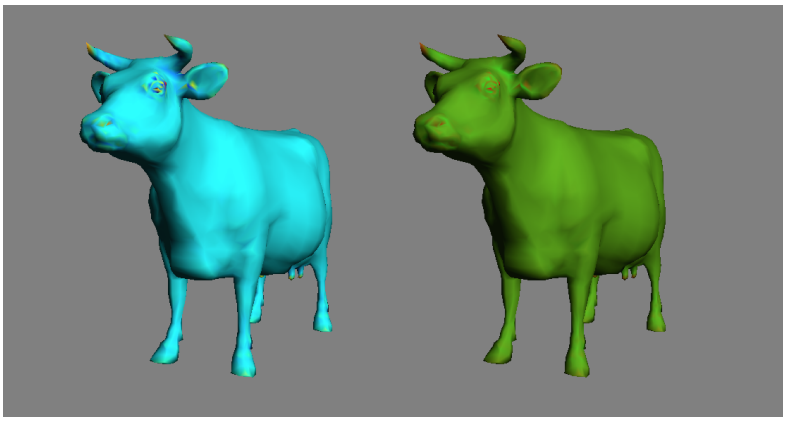

*Figure 5:* Gaussian curvature on the mesh *cow.obj*. From left to right, the color maps Jet and Green-Red have been utilized on the mesh.

### 3.5 Mean curvature

In Figure 6 the results from implementing mean curvature are visible.

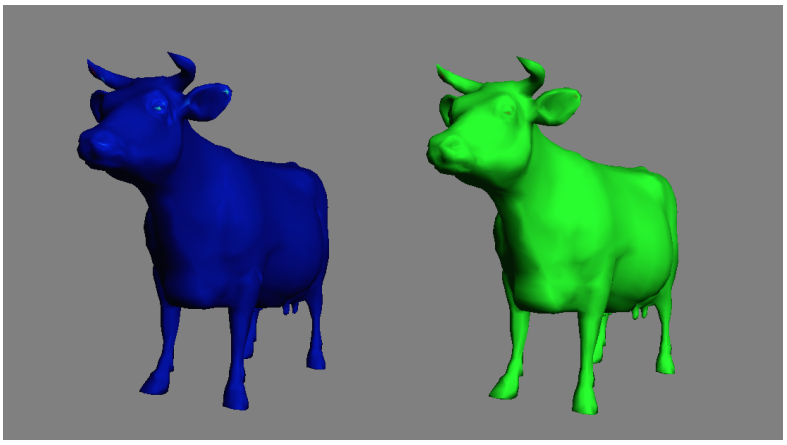

*Figure 6:* Mean curvature on the mesh *cow.obj*. From left to right, the color maps Jet and Green-Red have been utilized on the mesh.

### 3.6 Genus of a mesh

The results from determining the genus of a mesh can be seen in Figure 7. In this image the genus calculations have been performed on the meshes *cube.obj*, *genuscube.obj* and *torus.obj* and yielded the results 0, 5 and 1 respectively.

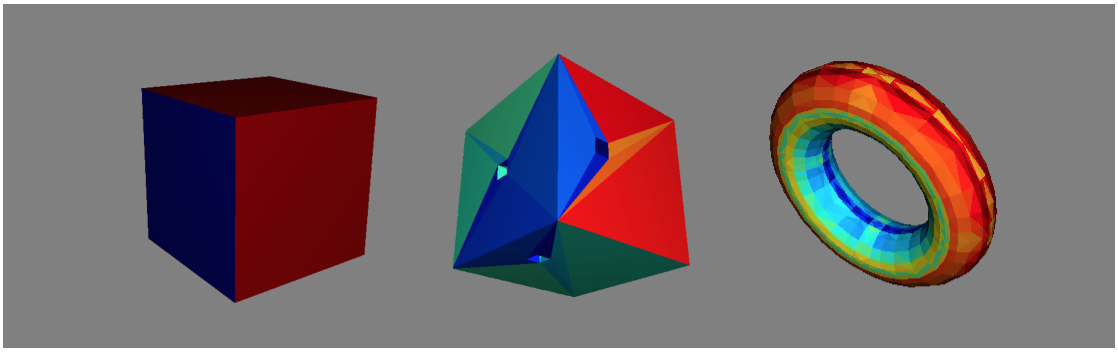

*Figure 7:* This figure illustrate three shapes with different genera. From left to right: A cube, a genus cube and a torus with genus 0, 5 and 1, respectively. The color map Jet was used on the shapes.

### 3.7 Non-closed surfaces

The results from extending the half-edge mesh so it would be able to handle non-closed surfaces is visible in Figure 8.

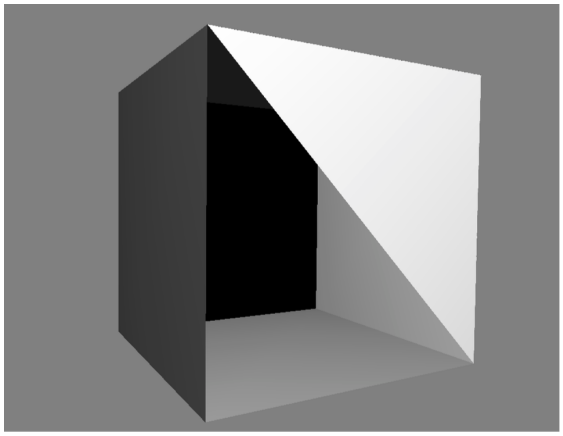

*Figure 8:* Non-closed surface.

## 4 Conclusion

The results for this lab illustrate that the half edge mesh allows for quick neighbor access and the assignments have been successfully carried out.

The vertex normals in Figure 4 point outwards and follow the mesh, and thereby show that the access to neighboring face normals is correct.

In Table 2, the calculated area for spheres, with differing sizes, is proven to be correctly implemented. The values correspond well to the analytical area, and the small deviation comes from the fact that the spheres are built by triangle faces. The results for calculating the volume of

spheres with varying size, in Table 2, illustrate, similarly to that of the surface area, correct results and the small loss of volume is due to the triangle approximation.

The approximations for the Gaussian curvature and the mean curvature seem to be correct. For the mean curvature, the results are more accurate when incorporating Voronoi area.

# 5 Lab partner and grade

This lab was completed in collaboration with Martin Gråd, margr484@student.liu.se. Since I have completed all the assignments marked with (*) and (**), as well as one marked with (***), I should qualify for grade 5.