Modeling and Animation

The Half-edge data structure

Gustav Gahm
gusga293@student.liu.se

April 22, 2010

Abstract

In this short paper I present theory and results for the first lab session in the course TNM079 - Modeling and Animation. The lab session was about *Mesh data structures* and in particular the *Half-edge mesh data structure*. The data structure and analysis of meshes is explained. Topics that are covered are e.g. area and volume calculations and different methods of curvature calculations.

1 Introducion

During the course TNM079 - Modeling and Animation six lab sessions are to be performed by the participants. Each of these sessions are to be presented in a short report describing what has been done and how. This report covers my work in the first lab session about mesh data structures.

1.1 The polygon mesh data structure

A mesh data structure is a way of describing the geometry and the topology of objects in computer graphics.

A common mesh data structure is the polygon mesh. A polygon mesh consists of vertices connected to form faces that in turn forms an object (Fig. 1). The popularity of the polygon mesh comes from its simpleness. It is fast to render and it is sparse in memory. The memory load of a polygon mesh in 3D space with an indexed face set[2] is equal to Eq. 1, in which V is the number of vertices and F the number of faces in the mesh.

$$size = 12V + 12F \ bytes \tag{1}$$

A disadvantage of the polygon mesh is that it is hard and time consuming ($O(N^2)$) to analyse[2], e.g. to find faces that share a specific vertex or calculate its area. Because of this a new mesh data structure called the half-edge mesh data structure was invented.

1.2 The half-edge mesh data structure

The half-edge data structure is an extension of the simple polygon mesh. Not only does it contain information of vertices and faces, it also stores information

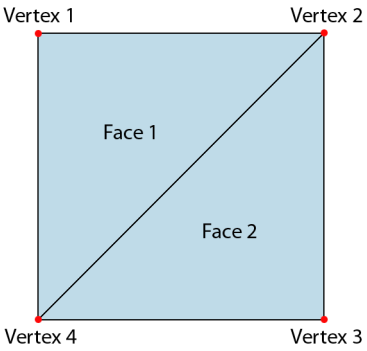


Figure 1: Polygon mesh of a square. Four vertices are connected to form two faces.

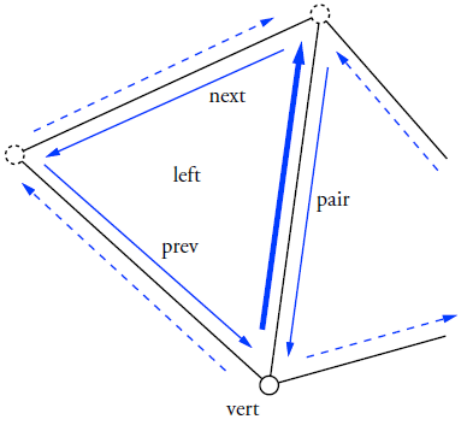


Figure 2: The half-edge mesh data structure.

about edges in the mesh, which simplifies analysis of the mesh. An edge is defined as a vertex, a face and three connected edges known as *next*, *prev* and *pair*, see Fig. 2. Using these connections it is possible to access any other edge on the mesh.

Because the data structure use more information it will use more memory. The size of a half-edge mesh is equal to Eq. 2 for a mesh that is a manifold [2].

$$size = \frac{F}{2} * 16 + 3F * 20 + F * 4 = 72F \ bytes \tag{2}$$

2 Method

2.1 Implementation of the half-edge mesh data structure

Implementing the hald-edge data structure can be a bit complicated. There are a lot connection that has to be done between vertices, edges and faces. Luckily

some of this work is already implemented in the base source code for lab session. The following steps explain the process.

1. In the *AddFace* function (located in HalfEdgeMesh.cpp) start by adding the three positions located in *verts* as vertices using the *AddVertex* function and save the indices returned by the function.

2. For each pair of vertices in step 1 connect them to form edges. This is done with the *AddHalfEdgePair* function and the indices returned by *AddVertex*. *AddHalfEdgePair* will return a new set of (edge)indices that are to be saved.

3. Continue by connecting the edges to each other. That is; for each edge set its indices for *next* and *prev* to the correct indices returned by *AddHalfEdgePair*.

4. Create a new *face* and push it to *mFaces* vector. Set its edge index to the first index returned by *AddHalfEdgePair*.

5. Calculate the normal of the newly created face using the *FaceNormal* function and add it to the face's *normal* variable.

6. Finaly set the *face* variable for each edge to the face created in 4.

2.2 Implementation of neighbor access

Finding neighbors of a vertex involve finding vertices or faces connected to that vertex. This is easily done using half-edge meshes.

Finding neighboring vertices to a vertex ($\mathbf{v}$) is done as follows.

1. Find the edge connected to $\mathbf{v}$.

2. Follow the *next-pointer* of the edge. Its vertex is the first neighboring vertex ($\mathbf{v_{n,i}}, \mathbf{i = 0}$) to $\mathbf{v}$.

3. The next neighbor vertex is found as $\mathbf{v_{n,i+1}} = \mathbf{v_{n,i}} \rightarrow \mathbf{next} \rightarrow \mathbf{pair} \rightarrow \mathbf{next} \rightarrow \mathbf{vertex}$.

4. Repeat step 3 until $\mathbf{v_{n,i+1}} = \mathbf{v_{n,i}}$.

Finding neighboring faces is done in a similar way.

1. Find the edge connected to $\mathbf{v}$ and record its face ($f_{n,i}, i = 0$) as the first face.

2. The next face is found as $f_{n,i+1} = f_{n,i} \rightarrow prev \rightarrow pair \rightarrow face$.

3. Repeat step 2 until $f_{n,i+1} = f_{n,i}$.

To get a better understanding of the above processes one can look at Fig. 3.

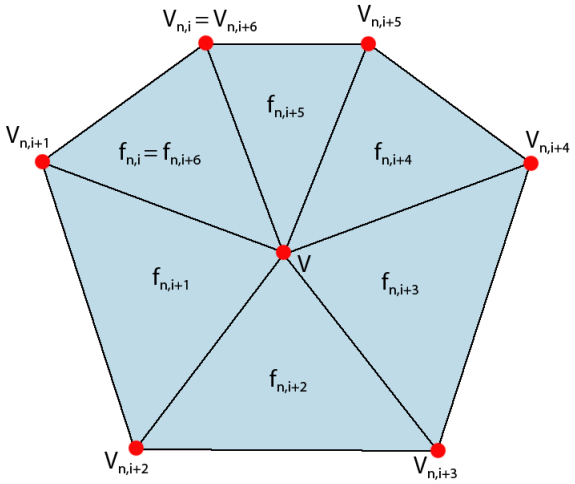


Figure 3: Neightbors to a vertex $\mathbf{v}$.

2.3 Calculation of vertex normals

Calculating per vertex normals becomes real easy using a half-edge mesh. All that has to be done is summing the normals of neighboring faces and normalize the resulting vector (Eq. 3).

$$\mathbf{n_{v_i}} = \widehat{\sum_{\mathbf{j \in N_1(i)}}^{\mathbf{n}} \mathbf{n_{f_i}}} \tag{3}$$

2.4 Calculation of surface area

The area of a surface can be calculated using a surface integral[2], Eq. 4.

$$A = \int_S dS \tag{4}$$

In the discreet case of a mesh it is possible to calculate Eq. 4 a Riemann sum (Eq. 5), in which $A(f_i)$ is the area of the i:th face and $A(f_i)$ is equal to Eq. 6, in which $\mathbf{v_1}$, $\mathbf{v_2}$ and $\mathbf{v_3}$ are the vertices of the face.

$$A_S = \int_S dS \approx \sum_{i \in S} A(f_i) \tag{5}$$

$$A(f_i) = \frac{1}{2}|(\mathbf{v_2} - \mathbf{v_1}) \times (\mathbf{v_3} - \mathbf{v_1})| \tag{6}$$

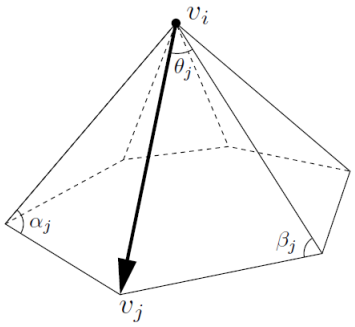


Figure 4: Calculation of θ_i. Image courtesy of [2]

2.5 Calculation of mesh volume

Calculating the volume of a closed surface is normally done using a volume integral. It can be shown[2] that a volume integral can be calculated as a surface integral as Eq. 7.

$$\int_S \mathbf{F} \cdot \mathbf{n} \, \mathbf{dA} = \int_\mathbf{V} \nabla \cdot \mathbf{F} \, \mathbf{d}\tau \tag{7}$$

It can also be shown[2] that by setting $F = (x, y, z)$ and calculate the integral as a Riemann sum the volume of a mesh is equal to Eq. 8, in which $\mathbf{v_1}$, $\mathbf{v_2}$ and $\mathbf{v_3}$ are the vertices of face f_i.

$$V = \frac{1}{3} \sum_{i \in S} \frac{\mathbf{v_1} + \mathbf{v_2} + \mathbf{v_3}}{3} \cdot \mathbf{n}(f_i) A(f_i) \tag{8}$$

2.6 Implementation and visualization of curvature

Curvature can be can be seen as a measure of how smooth a surface is[2]. There are a number of ways to calculate the curvature of a surface. The two most common techniques are *Gaussian curvature*[2] and *Mean curvature*[2]. Mean curvature will be explained in section 2.9 and Gaussian curvature will be covered here.

The Gaussian curvature K for a vertex ($\mathbf{v_i}$) is calculated as Eq. 9, in which A is the summed area of connected faces and θ_i is the angle between edges connected to $\mathbf{v_i}$, see Fig. 4.

$$K = \frac{1}{A} \left(2\pi - \sum_{j \in N_1(i))} \theta_i \right) \tag{9}$$

2.7 Improving the curvature estimate

Using Eq. 9 to calculate the curvature will only result in an estimate of the real curvature. It is possible to improve the accuracy of this calculation by changing the way A is calculated.

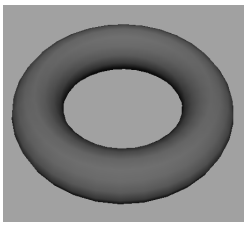


Figure 5: The torus is an example of an object with a genus of one.

Eq. 10 is known as the *Voronoi area* and by using $A = A_v$ one will get a more accurate curvature estimate.

$$A_v = \frac{1}{8} \sum_{j \in N_1(i)} (cot\alpha_j + cot\beta_j)|(\mathbf{v_i} - \mathbf{v_j})|^{\mathbf{2}} \tag{10}$$

2.8 Classification of the genus of the mesh

The genus of a mesh describes how many holes there are in a mesh, e.g. a genus of $G = 1$ means that the mesh has one hole in it, for example a torus (Fig. 5).

According to Euler-Poincar's formula[2] Eq. 11 is true for all half-edge meshes. Using this equation with a closed one-shell ($S = 1$) mesh of only triangles ($L = F$) it is possible to calculate the genus G as Eq. 12, in which V is the number of unique vertices, E the number of unique edges and F the number of faces.

$$V - E + F - (L - F) - 2(S - G) = 0 \tag{11}$$

$$G = 1 - \frac{V - E + F}{2} \tag{12}$$

2.9 Calculation of mean curvature and implementation of smoothing

The *mean curvature* [2][1] is another technique of calculating curvature and is a little bit more complicated to implement. Instead of using Eq. 4 one use Eq. 13.

$$H\mathbf{n} = \frac{1}{4A} \sum_{j \in N_1(i)} (cot\alpha_j + cot\beta_i)(\mathbf{v}_i - \mathbf{v}_j) \tag{13}$$

But only using Eq. 13 is not enough. H is the quantity that are sought after and $H\mathbf{n}$ is a vector quantity. Taking the magnitude of $H\mathbf{n}$ will give the magnitude of H, but not its sign. To find the sign one has to compare $H\mathbf{n}$ to the vertex normal, resulting in Eq. 14.

$$H = \begin{cases} -|H\mathbf{n}| & ,if\ H\mathbf{n} \cdot \mathbf{n}_{v_i} < 0 \\ |H\mathbf{n}| & ,else \end{cases} \tag{14}$$

An advantage of the mean curvature technique is the sign of H. It is possible to use that to move vertices along its normal in the direction of the sign. Doing

this can results in a smoother mesh. This is done according to Eq. 15, in which $\mathbf{v}_{new}$ is the new vertex position, $\mathbf{v}$ the old position, $\mathbf{n}_v$ the normal at vertex $\mathbf{v}$ and Δt is a small distance to move.

$$\mathbf{v}_{new} = \mathbf{v} + \mathbf{n}_v * \Delta t \tag{15}$$

3 Results

3.1 Calculation of vertex normals

As said before the strength of the half-edge data structure is its speed in analysis of meshes. This is easy to see in the case of per vertex normal calculation when a large number of neighboring vertices has to be found (Table 3.1). Even though it not that big of a difference for small meshes the half-edge mesh is more than 800 times faster for the large bunny mesh.

Table 1: Time required to calculate per vertex normals.

Mesh	Half-edge mesh(s)	Polygon mesh(s)
Sphere 1.0	0.03	0.05
Cow	0.02	0.27
Bunny (Small)	0.09	9.34
Bunny (Large)	0.76	625.58

3.2 Calculation of mesh area and volume

Some results of calculating the area and volume for a set of meshes can be seen in Table. 3.2. It is quite hard to verify that the result is correct for the Cow and Bunny mesh, but it is possible to calculate the area and volume of the sphere. The area of a unit sphere is $A = 12.5564$ and its volume is $V = 4.18879$. When comparing the results in Table. 3.2 with the correct values one can see that the error is very small. Because of this we can assume that the area and volume calculations are good approximations.

Table 2: Area and volume for a set of meshes.

Mesh	Area(s)	Volume(s)
Cow	1.09009	0.0534869
Bunny (Small)	225.482	181.223
Sphere (radius = 1.0)	12.511	4.15192

3.3 Implementation and visualization of curvature

When implemented and visualized the curvature of a surface can affect appearance of an object a lot, as can be seen Fig. 8(b). The curvature visualized in Fig. 7 and Fig. 8 is using the improved approximation described in 2.7.

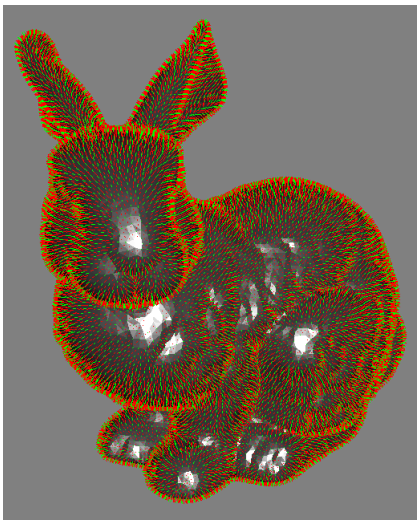


Figure 6: Normals calculated for the *Bunny(Small)*. Face normals are shown in red and vertex normals in green

3.4 Calculation of mean curvature and implementation of smoothing

Smoothing objects with help of the mean curvature works well. By using a very small Δt (< 0.1) and iterating the process for the mesh a couple of hundred times, the mesh becomes smoother. This can be seen in Fig. 8(b).

4 Conclusions

Based on the results above it easy to see that the half-edge mesh is a useful tool in modelling. The time spent analysing a mesh is greatly reduced. The downside of the data structure compared to a simple polygon mesh is its complexity. Half-edge meshes can be quite a daunting task to implement and they use more memory than a simple polygon mesh.

References

[1] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 317–324, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[2] Andreas Söderström Gunnar Läthén, Ola Nilsson. Mesh data structures, 2010.

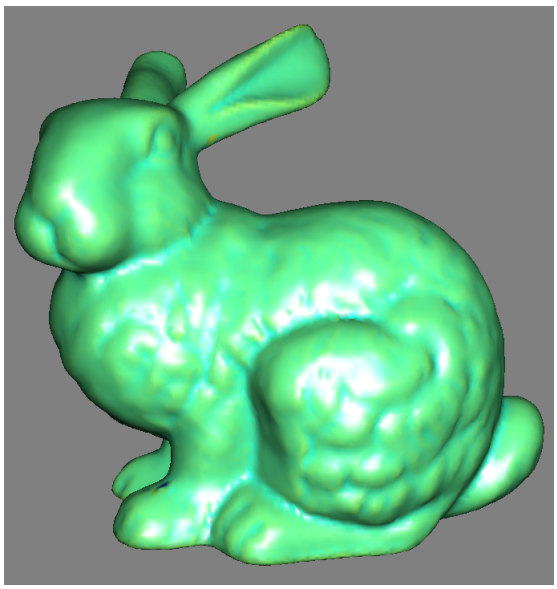
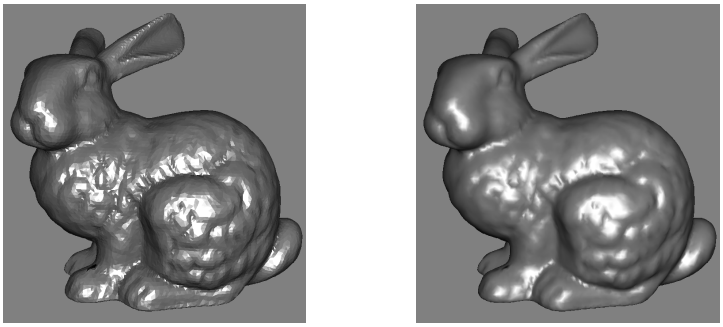
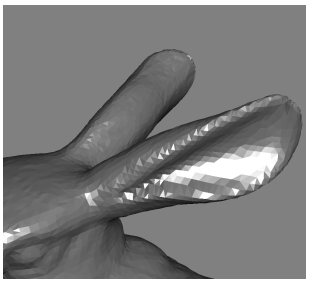


Figure 7: Visualization of curvature.

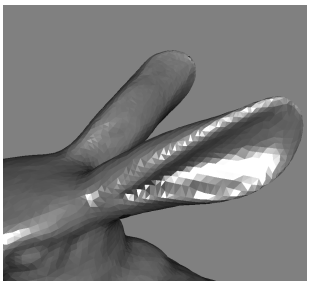


(a) No smoothing (b) Smoothing

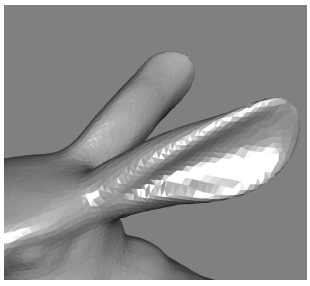
Figure 8: Example of how Gaussian-curvature-smoothing can affect a mesh.



(a) No smoothing



(b) Smoothing



(c) More smoothing

Figure 9: Example of how mean-curvature-smoothing can affect a mesh.