LAB REPORT: LAB 1

TNM079, MODELING AND ANIMATION

Algot Sandahl
algsa119@student.liu.se

Tuesday 13th April, 2021

Abstract

This lab report describes the tasks and results from lab 1 in the course TNM079, Modeling and Animation at Linköping University. The topic of the lab was the half-edge data structure. The adjecency information was used to calculate vertex normals, curvature, surface area, volume, genus and the number of shells. The performance improvements when making neighborhood-dependent calculations were found to be large when compared with a "polygon-soup" model. Additionally, different curvature measures were compared.

1 Background

The most obvious way to store a triangle model is to simply store a list of faces where each face is made up of three vertices. In some applications, for example rendering, this works fine. The problem presents itself when trying to perform operations that depend on the surroundings of a vertex, or face, for example when calculating vertex normals based on the surrounding faces. This information does not exist in such a structure, and to find neighbors, all possible candidates might have to be tested. This way of storing models is often called "polygon soup", since there is no structure to the polygons.

One way to store adjacency information in a model is the well-known *half-edge data structure*. This lab report covers a few of the operations enabled by the half-edge data structure. The half-edge data structure gets its name from the fact that each edge in the model is stored as two half-edges. In its simplest form, the a half-edge model can be described by:

- `Vertex`—x, y, and z coordinates, and a pointer to a neighboring half-edge
- `Face`—a pointer to one of the half-edges of the face.
- `HalfEdge`—a pointer to the vertex at the start of the edge, next, prev, and pair pointers, and a pointer to the face it is part of.
- `Mesh`—Lists of the above structures.

It is often interesting to find the neighboring vertices and faces of a vertex. The neighboring vertices—the 1-ring—of a vertex can be found by following the edge pointer of the vertex. The direction of the half-edge given by this operation is pointing out from the vertex, which means that the vertex pointer of the edge points back to the original vertex. To access a neighbor vertex, the prev pointer of the edge is used. Then a half-edge pointing inwards is obtained, containing a pointer to a neighboring vertex. Then the rest of the vertices are found by iterating around the vertex; each step consists of following the pair-followed by the prev-pointer. The process is illustrated in Figure 1. The processing of finding the neighboring faces is the same with the exception that the face pointer of the edge is used instead of the vertex pointer.

Vertex normals can be calculated in many ways. This lab uses one called *mean weighted*

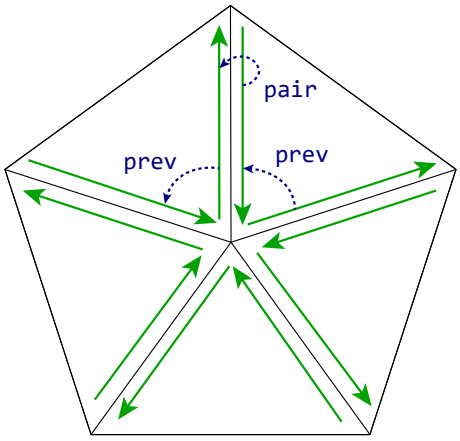


Figure 1: An illustration of the traversal of neighboring vertices. The green arrows represent half-edges, and the dashed lines represent pointers used to traverse the neighborhood.

equally and is, as the name implies, simply the normalized sum of all adjacent faces' normals. The face normals are given by

$$\hat{\mathbf{n}} = \frac{\mathbf{n}}{\|\mathbf{n}\|}, \quad \mathbf{n} = (\mathbf{v}_2 - \mathbf{v}_1) \times (\mathbf{v}_3 - \mathbf{v}_1) \quad (1)$$

where $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$ are the face vertices.

The adjecency information can also be used to calculate the surface area of a model. The surface area is simply given by

$$A_S = \sum_i A(f_i) \quad (2)$$

where $A(f_i)$ is the area of the face f_i, and the summation is done over all faces. Furthermore the area of each face is given by

$$A = \frac{\|(\mathbf{v}_2 - \mathbf{v}_1) \times (\mathbf{v}_3 - \mathbf{v}_1)\|}{2}. \quad (3)$$

Finding the volume of the mesh is not as straight-forward as finding its area, but it can be shown that it can be calculated according to

$$V = \frac{1}{3} \sum_{i \in S} \frac{(\mathbf{v}_1 + \mathbf{v}_2 + \mathbf{v}_3)_{f_i}}{3} \cdot \hat{\mathbf{n}}(f_i) A(f_i) \quad (4)$$

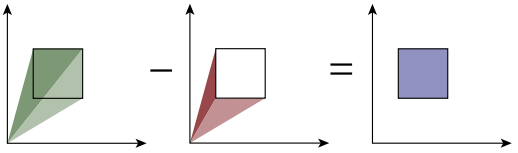


Figure 2: A simple example showing the principle behind the volume calculation. Green elements meet the object where the surface normal points away from the origin; the opposite is the case for the red elements.

where $\hat{\mathbf{n}}(f_i)$ is the face normal of the ith face, $A(f_i)$ is the area of the ith face (given by (3)), and the fraction inside the sum is the midpoint of the face, f_i.

This can be interpreted as the sum of singed volume elements. Each volume element is a tetrahedron with one vertex at the origin and the other three given by the face of the surface. The volume of a tetrahedron is given by

$$V_{\text{tetra}} = \frac{A_{\text{base}} h}{3} \quad (5)$$

where A_{base} is the area of the base, and h is the height form the base to the apex. The choice of base is arbitrary, but in this case it is the selected face of the mesh.The height, h, can be obtained as the dot product between the face normal and a vector starting at the origin and pointing to a point at the face. In this case the middle point, $(\mathbf{v}_1 + \mathbf{v}_2 + \mathbf{v}_3)_{f_i}/3$, is used. The dot product also has the effect that if the normal is pointing in the opposite direction of the vector emanating from the origin, the result becomes negative. By summing these signed volume elements, the volume of the mesh is obtained. This is examplified by Figure 2.

The smoothness of a surface at a point can be quantified by different types of surface curvature measures. The two most prevalent types of curvature are *Gaussian curvature* and *mean curvature*. Both are defined in terms of principal curvatures, κ_1 and κ_2. The principal curvatures are the minimum and maximum curvatures found among all possible normal planes at the given evaluation point.

The Gaussian curvature is defined as

$$K = \kappa_1 \kappa_2 \quad (6)$$

but since this equation has no obvious implementation, this equation is used instead:

$$K = \frac{1}{A_N}\left(2\pi - \sum_{j \in N_1(i)} \theta_j\right) \tag{7}$$

where A_N is the area of the adjacent faces, θ_j is defined in Figure 3, and $N_1(i)$ is the 1-ring neighborhood of the evaluation point. On a flat surface, the sum is naturally 2π since the all the angles live in one plane; this results in a curvature of zero, which makes intuitive sense.

The other curvature measure, the mean curvature, is useful since it can be used to distinguish between concave and convex curvature, unlike the Gaussian curvature. It is defined as

$$H = \frac{\kappa_1 + \kappa_2}{2}. \tag{8}$$

and can be calculated for a triangle model as

$$H = \left\| \frac{1}{4A_N} \sum_{j \in N_1(i)} (\cot \alpha_j + \cot \beta_j)(\mathbf{v}_i - \mathbf{v}_j) \right\| \tag{9}$$

where the vectors and angles in the equation are defined by Figure 3.

The estimates provided by (7) and (9) can both be improved by replacing the area, A_N, with the Voronoi area:

$$A_v = \frac{1}{8} \sum_{j \in N_1(i)} (\cot \alpha_j + \cot \beta_j) \left\| \mathbf{v}_i - \mathbf{v}_j \right\|^2. \tag{10}$$

The *genus* of a mesh is a topological quantity, that in some sense can be seen as the number of holes of a surface. To calculate the genus, G, of a mesh, the Euler–Poincaré formula can be used:

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

where S is the number of shells, V is the number of vertices, E is the number of edges, F is the number of faces, and L is the number of loops. The number of edges, E, in a half-edge mesh are half as many as the number of half-edges. In a triangle mesh, each triangle forms exactly one loop; consequently, the number of loops, L, is the same as the number of faces, F.

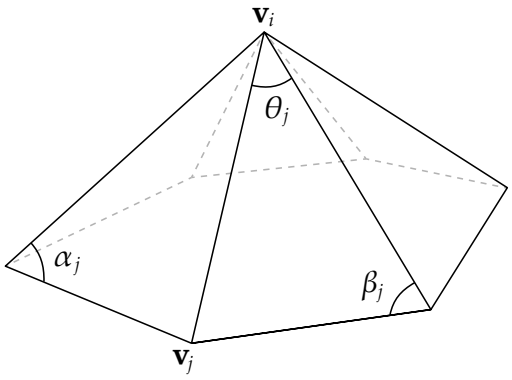


Figure 3: Angles used in curvature calculations.

The number of shells, S, of a mesh can be found by starting at a vertex and iteratively tagging all vertices that could be reached. If unvisited vertices remains, a new starting point out of the unvisited vertices is chosen and the procedure is restarted. This is done until all points have been visited. The number of shells is then given by the number of times a point from the unvisited nodes had to be chosen (including the first one).

2 Results

The results of the lab are presented in this chapter. The performance measurements were done on an 8-core M1 MacBook Air with 16 GB of RAM.

2.1 Surface Normals

Figure 5 shows a comparison between the vertex normals and the face normals. As seen, the vertex normals can be used with Gouraud shading to provide a much more natural look for smooth areas, see for example the donut. As demonstrated by the pyramid, sharp edges are not handled well; the face and vertex normals for a pyramid are visualized directly in Figure 5c. The averaging of the face normals implicitly assumes that the surface should be smooth.

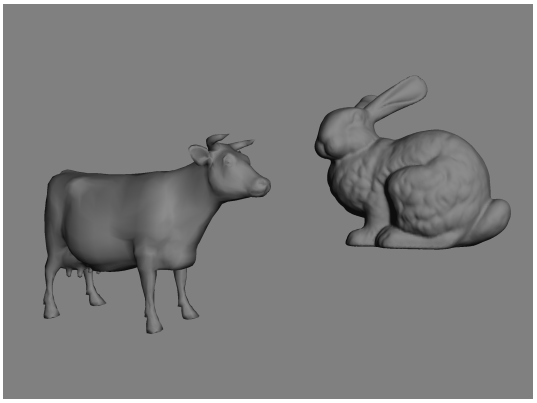


Figure 4: The models used to measure normal calculation times.

Table 1: Time taken to calculate vertex normals for `HalfEdgeMesh` and `SimpleMesh` mesh. The times are given in milliseconds.

Model	Vertices	`HalfEdge` time	`Simple` time
`cow`	2,903	3	121
`bunny_medium`	35,034	45	16,564

The performance of the vertex normal operation is drastically improved when compared to the `SimpleMesh` implementation, as seen in Table 1. The models used are shown in Figure 4.

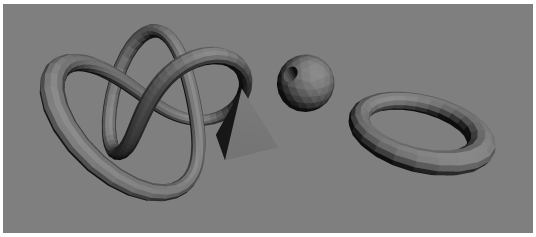
2.2 Surface Curvature

Comparisons between different curvature measures can be found in Figure 6. They are all shown on a unit sphere.

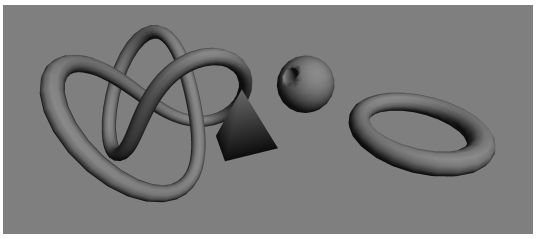
2.3 Surface Area and Mesh Volume

The volume and surface area of perfect spheres are well-known, so they are suitable for comparisons. Table 2 show the calculated areas and volumes for two different spheres.

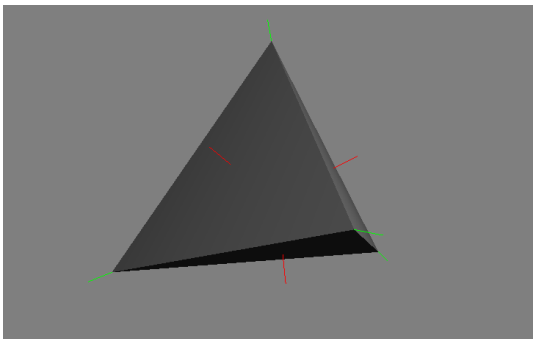
The volume calculations were also tested using different points on the face, i.e. with $(\mathbf{v}_1 + \mathbf{v}_2 + \mathbf{v}_3)_{f_i}/3$ in (4) replaced with one of the vertex points: $\mathbf{v}_1$, $\mathbf{v}_2$, or $\mathbf{v}_3$. The different expressions all produced the same results: 4.15192 for the sphere with radius 1, and 0.0534869 for the cow model.



(a) Flat shading, using the face normals.



(b) Gouraud shading using the calculated vertex normals.



(c) Face normals (red), and vertex normals (green).

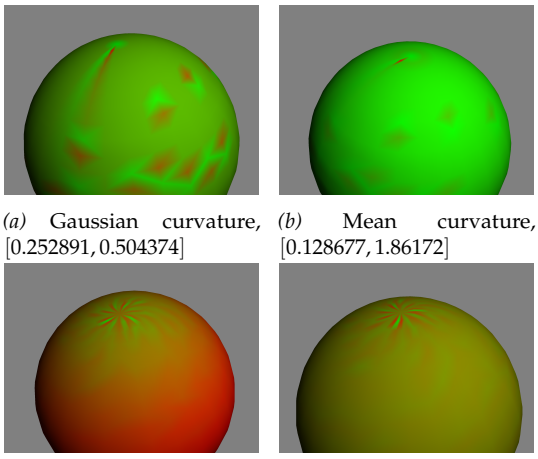
Figure 5: A demonstration of the difference between face normals and the implementation of vertex normals described in this report.

2.4 Topology

The genus and shell calculations were tested on three different models: a sphere, a donut, and the model `genus_test` shown in Figure 5a. The number of shells calculated was 1 for the sphere and the donut, and 4 for `genus_test`. The calculated genus was 0 for the sphere, 1 for the donut, and 3 for `genus_test`.

Table 2: Calculated areas and volumes compared with the exact values for perfect spheres.

Model	Radius	Vertices	Calc. area	Calc. volume	Exact area	Exact volume
`sphere_0.1`	0.1	994	0.12511	0.0041519	$0.04\pi \approx 0.12566$	$\frac{4}{3}\pi 0.1^3 \approx 0.0041887$
`sphere_1.0`	1	994	12.511	4.15192	$4\pi \approx 12.56637$	$\frac{4}{3}\pi \approx 4.18879$



(a) Gaussian curvature, $[0.252891, 0.504374]$

(b) Mean curvature, $[0.128677, 1.86172]$

(c) Gaussian curvature with voronoi, $[0.996891, 1.00619]$

(d) Mean curvature with voronoi, $[0.997823, 1.00227]$

Figure 6: Comparison between different curvature measures. The extent of the values over the entire surface is used to specify the range of the color scale, and is provided in the caption. The model used is a unit sphere, so curvature should be one.

3 Conclusion

The half-edge mesh trades some memory efficiency for efficiency when it comes to finding the neighboring vertices or faces.

3.1 Surface Normals

The normal calculation times, as seen in Table 1, are drastically improved for the half-edge data structure, especially when the number of vertices is high. This clearly exemplifies the effects of the adjacency information; all adjecent vertices and faces are easily reached through just a few pointers. In the `SimpleMesh` implementation, a full search through all vertices has to be done for each vertex, massively increasing the runtime.

3.2 Surface Curvature

The usage of the Voronoi area, divides the mesh into non-overlapping surface patches, and provide a highly significant improvement to the curvature estimates. The estimates calculated with the full area of the neighboring faces provide a very poor result, both in terms of deviation from the true value, and in terms of inconsistency over the surface area.

3.3 Surface Area and Mesh Volume

It is worth noting that the since the sphere models are made up of triangles, they do not actually have the exact same area and volume as a perfect sphere. Thus it makes sense that the calculated values deviate from those of a perfect sphere.

The volume calculation using different evaluation points in the calculation provides the same results. This makes sense since the length of the projection of the position vector onto the normal, should produce the same results, as the position vector always points at the triangle face, which is orthogonal to the normal.

3.4 Topology

The shell and genus calculations worked as expected.

Lab Partner and Grade

The lab was done together with Viktor Sjögren. All lab tasks were finished and the report aims for grade 5.