Lecture 3
Mesh

Lin ZHANG
School of Software Engineering
Tongji University
Spring 2012
Outline

• Overview
• Mesh data structures
  • Desired characteristics
  • Independent faces
  • Indexed face set
  • Adjacency list
  • Half-edge
• Subdivision surfaces
  • Introduction
  • Linear subdivision—an example
  • Different subdivision schemes overview
Overview

- 3D polygonal mesh
  - Representing a 2D surface embedded in 3D by using a set of polygons

- Why are they of interest?
  - Simple, common representation
  - Rendering with hardware support
  - Output of many acquisition tools
  - Input to many simulation/analysis tools
Overview

- 3D polygonal mesh
Overview

- 3D polygonal mesh

Geometry & topology

Face
Edge
Vertex \((x, y, z)\)
Outline

• Overview

• Mesh data structures
  • Desired characteristics
  • Independent faces
  • Indexed face set
  • Adjacency list
  • Half-edge

• Subdivision surfaces
  • Introduction
  • Linear subdivision—an example
  • Different subdivision schemes overview
Desired characteristics

- Generality
- Compact storage
- Efficient support for operations, e.g.
  - Given a face, find its vertices
  - Given a face, find neighboring faces
  - Given a vertex, find faces touching it
  - Given a vertex, find neighboring vertices
  - Given an edge, find vertices and faces it touches
Independent faces

- Each face lists vertex coordinates
  - Redundant vertices
  - No adjacency information

**Face Table**

- $F_0$: $(x_0, y_0, z_0)$, $(x_1, y_1, z_1)$, $(x_2, y_2, z_2)$
- $F_1$: $(x_3, y_3, z_3)$, $(x_4, y_4, z_4)$, $(x_5, y_5, z_5)$
- $F_2$: $(x_6, y_6, z_6)$, $(x_7, y_7, z_7)$, $(x_8, y_8, z_8)$
Indexed face set

- Each face lists vertex references
  - Shared vertices
  - Still no adjacency information

---

Vertex Table

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Coordinates</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_0$</td>
<td>$(x_0, y_0, z_0)$</td>
</tr>
<tr>
<td>$v_1$</td>
<td>$(x_1, y_1, z_1)$</td>
</tr>
<tr>
<td>$v_2$</td>
<td>$(x_2, y_2, z_2)$</td>
</tr>
<tr>
<td>$v_3$</td>
<td>$(x_3, y_3, z_3)$</td>
</tr>
<tr>
<td>$v_4$</td>
<td>$(x_4, y_4, z_4)$</td>
</tr>
</tbody>
</table>

Face Table

<table>
<thead>
<tr>
<th>Face</th>
<th>Vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_0$</td>
<td>0, 1, 2</td>
</tr>
<tr>
<td>$F_1$</td>
<td>1, 4, 2</td>
</tr>
<tr>
<td>$F_2$</td>
<td>1, 3, 4</td>
</tr>
</tbody>
</table>

Note CCW ordering
Indexed face set

- Each face lists vertex references
  - Storage efficiency?
  - Which operation(s) can be done in $O(1)$ time?

<table>
<thead>
<tr>
<th>Vertex Table</th>
<th>Face Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_0$: $(x_0, y_0, z_0)$</td>
<td>$F_0$: 0, 1, 2</td>
</tr>
<tr>
<td>$v_1$: $(x_1, y_1, z_1)$</td>
<td>$F_1$: 1, 4, 2</td>
</tr>
<tr>
<td>$v_2$: $(x_2, y_2, z_2)$</td>
<td>$F_2$: 1, 3, 4</td>
</tr>
<tr>
<td>$v_3$: $(x_3, y_3, z_3)$</td>
<td></td>
</tr>
<tr>
<td>$v_4$: $(x_4, y_4, z_4)$</td>
<td></td>
</tr>
</tbody>
</table>

Note CCW ordering
Adjacency list

- Store all vertex, face, and edge adjacencies

**Edge Adjacency Table**
- $e_0: v_0, v_1; F_0, \emptyset; \emptyset, e_2, e_1, \emptyset$
- $e_1: v_1, v_2; F_0, F_1; e_5, e_0, e_2, e_6$
- $\vdots$

**Face Adjacency Table**
- $F_0: v_0, v_1, v_2; F_1, \emptyset, \emptyset; e_0, e_2, e_0$
- $F_1: v_1, v_4, v_2; \emptyset, F_0, F_2; e_6, e_1, e_5$
- $F_2: v_1, v_3, v_4; \emptyset, F_1, \emptyset; e_4, e_5, e_3$

**Vertex Adjacency Table**
- $v_0: v_1, v_2; F_0; e_0, e_2$
- $v_1: v_3, v_4, v_2, v_0; F_2, F_1, F_0; e_3, e_5, e_1, e_0$
- $\vdots$
Half-edge

- Instead of a single edge, 2 **directed** “half edges”
- For each half-edge, we store a reference to
  - The vertex it points to
  - Its adjacent face (a zero pointer, if it is a boundary half-edge)
  - The next half-edge of the face or boundary (in counter-clockwise direction)
  - Its inverse (the opposite) half-edge
- For each face, we store a reference to
  - One of its half-edges
- For each vertex, we store a reference to
  - One of its outgoing half-edges
Half-edge

to_vertex
next_halfedge
opposite_halfedge
face
A basic half-edge structure can be realized using the following classes

```c
struct Halfedge
{
    HalfedgeRef next_halfedge;
    HalfedgeRef opposite_halfedge;
    FaceRef face;
    VertexRef to_vertex;
};
```

```c
struct Face
{
    HalfedgeRef halfedge;
};
```

```c
struct Vertex
{
    HalfedgeRef outgoing_halfedge;
};
```
Example: The following procedure enumerates all vertices that are adjacent to a given center vertex

```cpp
Enumerate_1_ring(VertexRef center)
{
    HalfedgeRef h = center.outgoing_halfedge;
    HalfedgeRef hstop = h;
    do
    {
        VertexRef v = h.to_vertex;
        // do something with v
        h = h. opposite_halfedge.next_halfedge;
    }
    while ( h != hstop );
}
```
Outline

• Overview
• Mesh data structures
  • Desired characteristics
  • Independent faces
  • Indexed face set
  • Adjacency list
  • Half-edge
• Subdivision surfaces
  • Introduction
  • Linear subdivision—an example
  • Different subdivision schemes overview
Subdivision surfaces

- Coarse mesh + subdivision rule
  - Smooth surface = limit of sequence of refinements
- Advantages
  - Simple (only need subdivision rule)
  - Local (only look at nearby vertices)
  - Arbitrary topology (since only local)
Subdivision surfaces

- How do you make a surface with guaranteed continuity?
Subdivision surfaces

- Repeated application of
  - Topology refinement (splitting faces)
  - Geometry refinement (weighted averaging)
Subdivision surfaces—an example

- Base mesh
Subdivision surfaces—an example

- Topology refinement
Subdivision surfaces—an example

- Geometry refinement
Subdivision surfaces—an example

- Topology refinement
Subdivision surfaces—an example

- Geometry refinement
Subdivision surfaces—an example

- Limit surface
Subdivision surfaces—an example

- Base mesh + limit surface
Outline

• Overview

• Mesh data structures
  • Desired characteristics
  • Independent faces
  • Indexed face set
  • Adjacency list
  • Half-edge

• Subdivision surfaces
  • Introduction
    • Linear subdivision—an example
    • Different subdivision schemes overview
Design of subdivision rules

- What types of input?
  - Quad meshes, triangle meshes, etc.
- How to refine topology?
  - Simple implementations
- How to refine geometry?
  - Smoothness guarantees in limit surface ($C^0, C^1, C^2 ...$)

Lin ZHANG, SSE, 2012
An example—linear subdivision

- Types of input
  - Quad mesh—four-sided polygon
  - Any number of quads may touch each vertex
- How to refine topology?
  - Split every quad into four at midpoints
- How to refine geometry?
  - Average vertex positions
An example—linear subdivision

```
LinearSubivision \((F_0, V_0, k)\)

for i = 1 \ldots k levels

\((F_i, V_i) = \text{RefineTopology}(F_{i-1}, V_{i-1})\)

\text{RefineGeometry}(F_i, V_i)

return \((F_k, V_k)\)
```
An example—linear subdivision

RefineTopology \((F, V)\)

\[
\begin{align*}
\text{new}V &= V \\
\text{new}F &= \{\} \\
\text{for each face } F_i \\
&\quad \text{Insert new vertex } c \text{ at centroid of } F_i \text{ into } \text{new}V \\
&\quad \text{for } j = 1 \text{ to } 4 \\
&\quad \quad \text{Insert new vertex } e_j \text{ at centroid of each edge } (F_{ij}, F_{ij+1}) \text{ into } \text{new}V \\
&\quad \text{for } j = 1 \text{ to } 4 \\
&\quad \quad \text{Insert new face } (F_{ij}, e_j, c, e_{j-1}) \text{ into } \text{new}F \\
\text{return } (\text{new}F, \text{new}V)
\end{align*}
\]
An example—linear subdivision

RefineGeometry( F, V )

newV = 0 * V

val = array of 0 whose size is number of vertices

newF = F

for each face Fi

cent = centroid for Fi

newV[Fi] += cent // syntax: repeat for all vtx indices in Fi

val[Fi] += 1 // syntax: repeat for all vtx indices in Fi

for each vertex newV[i]

newV[i] /= val[i]

return (newF, newV)

What is its physical meaning?
An example—linear subdivision

Example

Input mesh
An example—linear subdivision

Example

Topology refinement
An example—linear subdivision

Example

Geometry refinement
An example—linear subdivision

Example

Topology refinement
An example—linear subdivision

Example

Geometry refinement
An example—linear subdivision

Example

Topology refinement
An example—linear subdivision

Example

Geometry refinement
An example—linear subdivision

Example

Final result
Outline

• Overview
• Mesh data structures
  • Desired characteristics
  • Independent faces
  • Indexed face set
  • Adjacency list
  • Half-edge
• Subdivision surfaces
  • Introduction
  • Linear subdivision—an example
  • Different subdivision schemes overview

Lin ZHANG, SSE, 2012
Subdivision schemes comparison

- Common subdivision schemes
  - Loop [1]
  - Catmull-Clark [2]
- Differ in:
  - Input topology
  - How to refine topology
  - How to refine geometry

Subdivision schemes comparison

Loop

Catmull-Clark

Loop

Catmull-Clark
Thanks for your attention