Working with Meshes

Surfaces/Meshes

We’ll stick to triangles
**Discrete Surfaces**

Setup
- topology & geometry
- simplicial complex: “triangle mesh”  \( K = \{V, E, T\} \)
- 2-manifold  
  \[ V = \{v_i\} \quad E = \{e_{ij}\} \quad T = \{t_{ijk}\} \]
- Euler characteristic  
  \[ F - E + V = 2(1 - g) = \chi \]

---

**What’s a Mesh?**

Formally
- abstract simplicial complex \( K \)
- singletons, pairs, triples, ... of integers
  \[ V = \{1, 2, 3, \ldots\} \quad E = \{\{i, j\}, \{k, l\}, \ldots\} \]
- containment property  
  \[ F = \{\{i, j, k\}, \{j, i, l\}, \ldots\} \]
- part \( \rho \in K \land \sigma \subseteq \rho \Rightarrow \sigma \in K_{\text{face}}, \emptyset \)
S I M P L I C I A L  C O M P L E X

Topological realization
- identify $V$ with unit vectors in $\mathbb{R}^N$
- subset topology of ambient space
- closure, star, and link

$|K| = \bigcup_{\sigma \in K} |\sigma|$

$ClL = \{p | p \preceq \sigma, \sigma \in L\}$

$StL = \{p | p \preceq \rho, \sigma \in L\} \setminus L - 0$

T O P O L O G I C A L  S T R U C T U R E

2-manifold (with boundary)
- every point has an open, (half-) disklike subset surrounding it

$|K| \text{ 2-manifold iff } |St v| \approx \mathbb{R}^2$

$|St \sigma| = \bigcup_{\rho \in St \sigma} \text{int}|\rho|$
Topological Invariants

Euler characteristic
- for surfaces: $F - E + V = \chi = 2(1 - g)$
- not required to be simplicial
- more generally for simplicial complexes
- proof by induction (shelling)

\[
\chi(K) = \sum_{\emptyset \neq \rho \subset K} (-1)^{\dim \rho}
\]

Simplicial Complex

Geometric realization
- the concrete embedding $\pi_v(|K|)$
\[\pi_v : \mathbb{R}^n \rightarrow \mathbb{R}^3\]
- vertex images specify everything
- piecewise linear approximation
- presumably approximation of underlying smooth surface
**Mesh Structure**

**Input**
- typically
  - list of vertices (how long?)
  - list of triangles (until EOF)
- need to build mesh structure
  - infer topology
  - check topology
  - oriented (orientable?)

---

**Building the Mesh**

**What do we need?**
- array of pointers to vertices
- choices for basic topology primitive
  - (half-)edges
  - triangles
- we’ll use triangles
Types of Operations

What do we need to support?
- iterate over all vertices (easy)
- iterate over all triangles (easy)
- for a triangle visit
  - incident vertices (easy)
  - incident triangles (easy)
- for a vertex visit
  - star \( \forall v_i : \{t_{ijk} \} \subseteq T \)
  - link \( \forall v_i : \{e_{jk} | t_{ijk} \in T \} \)
  - different flavors \( \forall v_i : \{v_j | e_{ij} \in E \} \)
- need back pointer
  - vertex points to one incident triangle
  - careful at boundary!
TYPES OF OPERATIONS

What about edges?
- visit all edges
- not explicitly represented...
- do we need edges? Yes!
  - discover triangle adjacencies
  - map pairs of integers to triangles
    \[ e_{ij} \mapsto \{t_{ijk}, t_{jil}\} \]

OPERATIONS TO SUPPORT

For later (think about it now...)
- edge collapse
  - legality?
- edge flip
**Data Structures**

**Triangles**
- consistent ordering of vertex and triangle incidences

```c
Triangle{
    Vertex *v[3];
    Triangle *t[3];
}
```
- triangles across from vertices

---

**What Data Where?**

**Attributes**
- normal, color, texture coordinates
- later: forces, velocities, mass
- why not just lay everything out in arrays?
- OK, but ...
- changes in structure!
- very hard to debug...
**EXAMPLES**

**Vertex normals**

- gradient of volume
  \[ n_i = \frac{1}{2} \sum_{t_{ijk}} (p_j - p_i) \times (p_k - p_i) \]

\[ N_i = \frac{n_i}{|n_i|} \]

\[ \forall v_i : n_i = \vec{0} \]

\[ \forall t_{ijk} : a_{ijk} = (p_j - p_i) \times (p_k - p_i) \]

\[ \forall t_{ijk} : \begin{cases} 
  n_{i}^+ = a_{ijk} \\
  n_{j}^+ = a_{ijk} \\
  n_{k}^+ = a_{ijk} 
\end{cases} \]

\[ \forall v_i : N_i = \frac{n_i}{|n_i|} \]

---

**EXAMPLE**

**Gaussian curvature**

\[ \forall v_i : K_i = 2\pi - \sum_{t_{ijk}} \alpha_{ijk}^i \]

\[ \forall v_i \in V \setminus \partial V : K_i = 2\pi - \ldots \]

\[ \forall v_i \in \partial V : K_i = \pi - \ldots \]

\[ \forall t_{ijk} : \begin{cases} 
  K_{i}^- = \text{atan2}(|a_{ijk}|, (p_j - p_i) \cdot (p_k - p_i)) \\
  K_{j}^- = \text{atan2}(|a_{ijk}|, (p_k - p_j) \cdot (p_i - p_j)) \\
  K_{k}^- = \text{atan2}(|a_{ijk}|, (p_i - p_k) \cdot (p_j - p_k)) 
\end{cases} \]
PRINCIPLES

As you write code...
- assumptions are ok, but you must assert them explicitly
- orientability
- 2-manifold property
- avoid storing the same information multiple times
- nasty to keep current under changes

OTHER TRICKS

As you write code
- use two sided lighting
- abstract the iterators!
  - what about boundary vertices?
- keep iterators sorted
  - interior then boundary vertices
  - interior then boundary triangles