Adjacency Data Structures

material from Justin Legakis

Last Time?

- Simple Transformations
- Classes of Transformations
  - homogeneous coordinates
- Representation
- Composition
  - not commutative
- Orthographic & Perspective Projections

Today

- Surface Definitions
- Simple Data Structures
- Fixed Storage Data Structures
- Fixed Computation Data Structures
- Mesh Simplification

Today

- Surface Definitions
  - Well-Formed Surfaces
    - Components Intersect "Properly"
      - Faces are: disjoint, share single Vertex, or share 2 Vertices and the Edge joining them
      - Every edge is incident to exactly 2 vertices
      - Every edge is incident to exactly 2 faces
    - Local Topology is "Proper"
      - Neighborhood of a vertex is homeomorphic to a disk (permits stretching and bending, but not tearing)
      - Called a 2-manifold
      - Boundaries: half-disk, "manifold with boundaries"
    - Global Topology is "Proper"
      - Connected
      - Closed
      - Bounded
  - Orientable Surfaces
  - Computational Complexity
- Simple Data Structures
- Fixed Storage Data Structures
- Fixed Computation Data Structures
- Mesh Simplification

Well-Formed Surfaces

Orientable Surfaces?
Closed Surfaces and Refraction

- Original Teapot model is not "watertight": intersecting surfaces at spout & handle, no bottom, a hole at the spout tip, a gap between lid & base.
- Requires repair before ray tracing with refraction.

Computational Complexity

- Adjacent Element Access Time
  - linear, constant time average case, or constant time?
  - requires loops/recursion/if?
- Memory
  - variable size arrays or constant size?
- Maintenance
  - ease of editing
  - ensuring consistency

Questions?

Today

- Surface Definitions
- Simple Data Structures
  - List of Polygons
  - List of Edges
  - List of Unique Vertices & Indexed Faces:
  - Simple Adjacency Data Structure
- Fixed Storage Data Structures
- Fixed Computation Data Structures
- Mesh Simplification

List of Polygons:

(3,-2,5), (3,6,2), (-6,2,4)
(2,2,4), (0,-1,-2), (9,4,0), (4,2,9)
(1,2,-3), (8,8,7), (-4,-5,1)
(-8,2,7), (-2,3,9), (1,2,-7)

List of Edges:

(3,6,2), (-6,2,4)
(2,2,4), (0,-1,-2)
(9,4,0), (4,2,9)
(8,8,7), (-4,-5,1)
(-8,2,7), (1,2,-7)
(3,0,-3), (-7,4,-3)
(9,4,0), (4,2,9)
(3,6,2), (-6,2,4)
(-3,8,-4), (7,3,-4)
List of Unique Vertices & Indexed Faces:

Vertices: (-1, -1, -1)
(-1, -1, 1)
(-1, 1, -1)
(-1, 1, 1)
(1, -1, -1)
(1, -1, 1)
(1, 1, -1)
(1, 1, 1)

Faces: 1 2 4 3
5 7 8 6
1 5 6 2
3 4 8 7
1 3 7 5
2 6 8 4

Problems with Simple Data Structures

- No Adjacency Information
- Linear-time Searches

Structured
Unstructured

- Adjacency is implicit for structured meshes, but what do we do for unstructured meshes?

Mesh Data

- So, in addition to:
  - Geometric Information (position)
  - Attribute Information (color, texture, temperature, population density, etc.)
- Let’s store:
  - Topological Information (adjacency, connectivity)

Simple Adjacency

- Each element (vertex, edge, and face) has a list of pointers to all incident elements
- Queries depend only on local complexity of mesh
- Data structures do not have fixed size
- Slow! Big! Too much work to maintain!

Questions?

Today

- Surface Definitions
- Simple Data Structures
- Fixed Storage Data Structures
  - Winged Edge (Baumgart, 1975)
- Fixed Computation Data Structures
- Mesh Simplification
**Winged Edge (Baumgart, 1975)**

- Each edge stores pointers to 4 Adjacent Edges
- Vertices and Faces have a single pointer to one incident Edge
- Data Structure Size? Fixed
- How do we gather all faces surrounding one vertex? Messy, because there is no consistent way to order pointers

**HalfEdge (Eastman, 1982)**

- Every edge is represented by two directed HalfEdge structures
- Each HalfEdge stores:
  - vertex at end of directed edge
  - symmetric half edge
  - face to left of edge
  - next points to the HalfEdge counter-clockwise around face on left
- Orientation is essential, but can be done consistently!

**HalfEdge (Eastman, 1982)**

- Loop around a Face:
  ```
  HalfEdgeMesh::FaceLoop(HalfEdge *HE) {
    HalfEdge *loop = HE;
    do {
      loop = loop->Next;
    } while (loop != HE);
  }
  ```
- Loop around a Vertex:
  ```
  HalfEdgeMesh::VertexLoop(HalfEdge *HE) {
    HalfEdge *loop = HE;
    do {
      loop = loop->Next->Sym;
    } while (loop != HE);
  }
  ```

**Today**

- Surface Definitions
- Simple Data Structures
- Fixed Storage Data Structures
- Fixed Computation Data Structures
  - HalfEdge (Eastman, 1982)
  - SplitEdge
  - Corner
  - QuadEdge (Guibas and Stolfi, 1985)
  - FacetEdge (Dobkin and Laszlo, 1987)

**HalfEdge (Eastman, 1982)**

- Starting at a half edge, how do we find:
  - the other vertex of the edge?
  - the other face of the edge?
  - the clockwise edge around the face at the left?
  - all the edges surrounding the face at the left?
  - all the faces surrounding the vertex?

**HalfEdge (Eastman, 1982)**

- Data Structure Size? Fixed
- Data:
  - geometric information stored at Vertices
  - attribute information in Vertices, HalfEdges, and/or Faces
  - topological information in HalfEdges only!
- Orientable surfaces only (no Mobius Strips!)
- Local consistency everywhere implies global consistency
- Time Complexity? linear in the amount of information gathered
SplitEdge Data Structure:

- HalfEdge and SplitEdge are dual structures!

\[
\text{SplitEdgeMesh::FaceLoop()} = \text{HalfEdgeMesh::VertexLoop()}
\]
\[
\text{SplitEdgeMesh::VertexLoop()} = \text{HalfEdgeMesh::FaceLoop()}
\]

Corner Data Structure:

- The Corner data structure is its own dual!

Questions?

Today

- Surface Definitions
- Simple Data Structures
- Fixed Storage Data Structures
- Fixed Computation Data Structures
  - HalfEdge (Eastman, 1982)
  - SplitEdge
  - Corner
  - QuadEdge (Guibas and Stolfi, 1985)
  - FacetEdge (Dobkin and Laszlo, 1987)
- Mesh Simplification

QuadEdge (Guibas and Stolfi, 1985)

- Consider the Mesh and its Dual simultaneously
  - Vertices and Faces switch roles, we just re-label them
  - Edges remain Edges
- Now there are eight ways to look at each edge
  - Four ways to look at primal edge
  - Four ways to look at dual edge

QuadEdge (Guibas and Stolfi, 1985)

- Relations Between Edges: Edge Algebra
- Elements in Edge Algebra:
  - Each of 8 ways to look at each edge
- Operators in Edge Algebra:
  - Rot: Bug rotates 90 degrees to its left
  - Sym: Bug turns around 180 degrees
  - Flip: Bug flips up-side down
  - Oneext: Bug rotates CCW about its origin (either Vertex or Face)
QuadEdge (Guibas and Stolfi, 1985)

- Some Properties of Flip, Sym, Rot, and Onext:
  - $\text{e Rot}^4 = \text{e}$
  - $\text{e Rot}^2 \neq \text{e}$
  - $\text{e Flip}^4 = \text{e}$
  - $\text{e Flip Rot Flip Rot} = \text{e}$
  - $\text{e Rot Flip Rot Flip} = \text{e}$
  - $\text{e Rot Onext Rot Onext} = \text{e}$
  - $\text{e Flip Onext Flip Onext} = \text{e}$
  - $\text{e Flip}^{-1} = \text{e Flip}$
  - $\text{e Sym} = \text{e Rot}^2$
  - $\text{e Rot}^{-1} = \text{e Rot}$
  - $\text{e Rot}^4 = \text{e Flip Rot Flip}$
  - $\text{e Onext}^{-1} = \text{e Rot Onext Rot}$
  - $\text{e Onext}^{-1} = \text{e Flip Onext Flip}$

- Other Useful Definitions:
  - $\text{e Lnext} = \text{e Rot}^{-1} \text{Onext Rot}$
  - $\text{e Rnext} = \text{e Rot Onext Rot}^{-1}$
  - $\text{e Dnext} = \text{e Sym Onext Sym}$
  - $\text{e Oprev} = \text{e Onext}^{-1} = \text{e Rot Onext Rot}$
  - $\text{e Lprev} = \text{e Lnext}^{-1} = \text{e Rot Sym}$
  - $\text{e Rprev} = \text{e Rnext}^{-1} = \text{e Sym Sym}$
  - $\text{e Dprev} = \text{e Dnext}^{-1} = \text{e Rot Sym}$

- All of these functions can be expressed as a constant number of Rot, Sym, Flip, and Onext operations independent of the local topology and the global size and complexity of the mesh.

FacetEdge (Dobkin and Laszlo, 1987)

- QuadEdge (2D, surface) → FacetEdge (3D, volume)
- Faces → Polyhedra / Cells
- Edge → Polygon & Edge pair

Questions?

Today

- Surface Definitions
- Simple Data Structures
- Fixed Storage Data Structures
- Fixed Computation Data Structures
- Mesh Simplification

Progressive Meshes

- Mesh Simplification
  - vertex split / edge collapse
  - geometry & discrete/scalar attributes
  - priority queue
- Level of Detail
  - geomorphs
- Progressive Transmission
- Mesh Compression
- Selective Refinement
  - view dependent

Hugues Hoppe “Progressive Meshes” SIGGRAPH 1996
Selective Refinement

- Remove a vertex & surrounding triangles, re-triangulate the hole
- Merge Nearby Vertices
  - will likely change the topology...

Other Simplification Strategies

- Remove a vertex & surrounding triangles, re-triangulate the hole
- Merge Nearby Vertices
  - will likely change the topology...

Preserving Discontinuity Curves

- Contract (merge) vertices $v_i$ and $v_j$ if:
  - $(v_i, v_j)$ is an edge, or
  - $\|v_i - v_j\| < t$, where $t$ is a threshold parameter
- Track cumulative error by summing 4x4 quadric error matrices after each operation:
  \[
  \Delta e = \sum_{p \in \text{operation}} v'(p)p'v
  \]
  \[
  = \sum_{p \in \text{operation}} \nu^2 p
  \]
  \[
  = \nu^2 \left( \sum_{p \in \text{operation}} \kappa_p \right)
  \]
  \[
  \kappa_p = \begin{bmatrix}
  a & b & c & d \\
  e & f & g & h \\
  i & j & k & l \\
  m & n & o & p
  \end{bmatrix}
  \]


When to Preserve Topology?

- How “equilateral” are the elements?
  - How are the elements distributed?
  - For Triangles:
    - Ratio of area to perimeter
    - Smallest angle
  - For Tetrahedra:
    - Ratio of volume to surface area
    - Smallest solid angle
  - Other Factors:
    - Aspect ratio of element
    - Ratio of distance to boundary

Judging Element Quality

- How “equilateral” are the elements?
  - For Triangles:
    - Ratio of area to perimeter
    - Smallest angle
  - For Tetrahedra:
    - Ratio of volume to surface area
    - Smallest solid angle
  - Other Factors:
    - Aspect ratio of element
    - Ratio of distance to boundary
Readings for Tuesday (1/20) *pick one*

- "Free-form deformation of solid geometric models", Sederberg & Parry, SIGGRAPH 1986
- "Teddy: A Sketching Interface for 3D Freeform Design", Igarashi et al., SIGGRAPH 1999
- Post a comment or question on the LMS discussion by 10am on Tuesday