



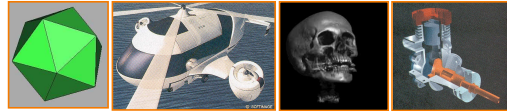
Subdivision Surfaces

Tom Funkhouser
Princeton University
COS 426, Spring 2006



Modeling

- How do we ...
 - Represent 3D objects in a computer?
 - Acquire computer representations of 3D objects?
 - Manipulate computer representations of 3D objects?
 - Analyze computer representations of 3D objects?



Different representations for different types of objects and operations



3D Object Representations

- Raw data
 - Voxels
 - Point cloud
 - Range image
 - Polygons
- Surfaces
 - Mesh
 - Subdivision
 - Parametric
 - Implicit
- Solids
 - Octree
 - BSP tree
 - CSG
 - Sweep
- High-level structures
 - Scene graph
 - Application specific



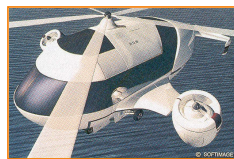
3D Object Representations

- Raw data
 - Voxels
 - Point cloud
 - Range image
 - Polygons
- Surfaces
 - Mesh
 - Subdivision
 - Parametric
 - Implicit
- Solids
 - Octree
 - BSP tree
 - CSG
 - Sweep
- High-level structures
 - Scene graph
 - Application specific



Surfaces

- What makes a good surface representation?
 - Accurate
 - Concise
 - Intuitive specification
 - Local support
 - Affine invariant
 - Arbitrary topology
 - Guaranteed continuity
 - Natural parameterization
 - Efficient display
 - Efficient intersections

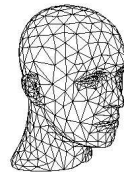


H&B Figure 10.46



Polygon Meshes

- How should we represent a mesh in a computer?
 - Efficient traversal of topology
 - Efficient use of memory
- Mesh Representations
 - Independent faces
 - Vertex and face tables
 - Adjacency lists
 - Winged-Edge

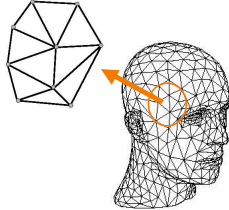


Zorin & Schroeder, SIGGRAPH 99, Course Notes

Polygon Meshes



- How should we represent a mesh in a computer?
 - Efficient traversal of topology
 - Efficient use of memory
- Mesh Representations
 - Independent faces
 - Vertex and face tables
 - Adjacency lists
 - Winged-Edge

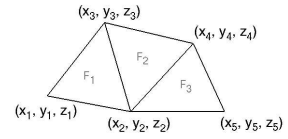
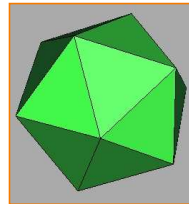


Zorin & Schroeder, SIGGRAPH 99, Course Notes

Independent Faces



- Each face lists vertex coordinates

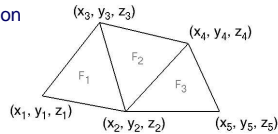
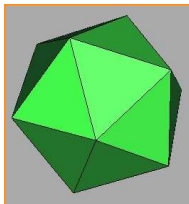


FACE TABLE			
F ₁	(x ₁ , y ₁ , z ₁)	(x ₂ , y ₂ , z ₂)	(x ₃ , y ₃ , z ₃)
F ₂	(x ₂ , y ₂ , z ₂)	(x ₄ , y ₄ , z ₄)	(x ₃ , y ₃ , z ₃)
F ₃	(x ₂ , y ₂ , z ₂)	(x ₅ , y ₅ , z ₅)	(x ₄ , y ₄ , z ₄)

Independent Faces



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

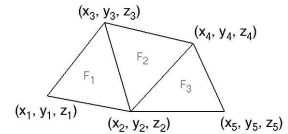
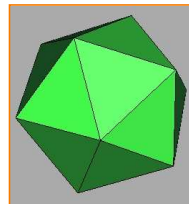


FACE TABLE			
F ₁	(x ₁ , y ₁ , z ₁)	(x ₂ , y ₂ , z ₂)	(x ₃ , y ₃ , z ₃)
F ₂	(x ₂ , y ₂ , z ₂)	(x ₄ , y ₄ , z ₄)	(x ₃ , y ₃ , z ₃)
F ₃	(x ₂ , y ₂ , z ₂)	(x ₅ , y ₅ , z ₅)	(x ₄ , y ₄ , z ₄)

Vertex and Face Tables



- Each face lists vertex references
 - Shared vertices



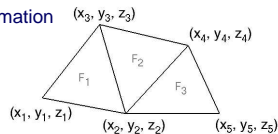
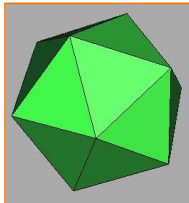
VERTEX TABLE			
V ₁	x ₁	y ₁	z ₁
V ₂	x ₂	y ₂	z ₂
V ₃	x ₃	y ₃	z ₃
V ₄	x ₄	y ₄	z ₄
V ₅	x ₅	y ₅	z ₅

FACE TABLE			
F ₁	V ₁	V ₂	V ₃
F ₂	V ₂	V ₄	V ₃
F ₃	V ₂	V ₅	V ₄

Vertex and Face Tables



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



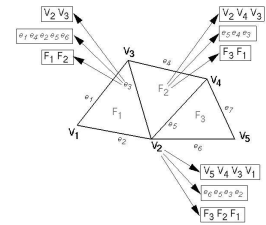
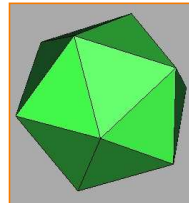
VERTEX TABLE			
V ₁	x ₁	y ₁	z ₁
V ₂	x ₂	y ₂	z ₂
V ₃	x ₃	y ₃	z ₃
V ₄	x ₄	y ₄	z ₄
V ₅	x ₅	y ₅	z ₅

FACE TABLE			
F ₁	V ₁	V ₂	V ₃
F ₂	V ₂	V ₄	V ₃
F ₃	V ₂	V ₅	V ₄

Adjacency Lists



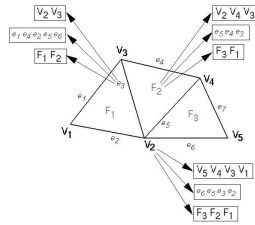
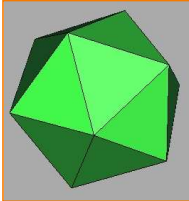
- Store all vertex, edge, and face adjacencies
 - Efficient topology traversal



Adjacency Lists



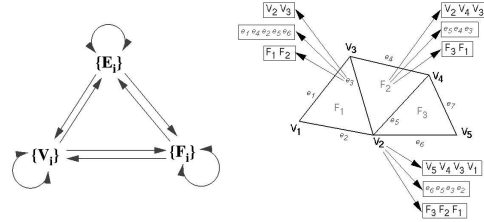
- Store all vertex, edge, and face adjacencies
 - Efficient topology traversal
 - Extra storage



Partial Adjacency Lists



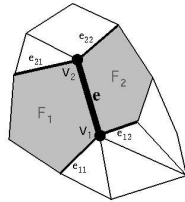
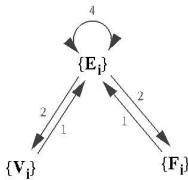
- Can we store only some adjacency relationships and derive others?



Winged Edge



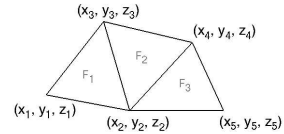
- Adjacency encoded in edges
 - All adjacencies in $O(1)$ time
 - Little extra storage (fixed records)
 - Arbitrary polygons



Winged Edge



- Example:



VERTEX TABLE					EDGE TABLE							FACE TABLE		
	v_1	x_1	y_1	z_1	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}
v_1	x_1	y_1	z_1	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	
v_2	x_2	y_2	z_2	e_6	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	
v_3	x_3	y_3	z_3	e_3	e_2	e_3	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	
v_4	x_4	y_4	z_4	e_5	e_4	e_5	e_6	e_7	e_8	e_9	e_{10}	e_1	e_2	
v_5	x_5	y_5	z_5	e_6	e_5	e_6	e_7	e_8	e_9	e_{10}	e_1	e_2	e_3	
					e_1	v_1	v_2	v_3	F_1	e_2	e_2	e_4	e_3	
					e_2	v_1	v_2	F_1	e_1	e_1	e_3	e_6	e_6	
					e_3	v_2	v_3	F_2	e_2	e_5	e_1	e_4	e_4	
					e_4	v_3	v_4	F_2	e_1	e_3	e_7	e_5	e_5	
					e_5	v_2	v_4	F_2	e_3	e_6	e_4	e_7	e_7	
					e_6	v_2	v_5	F_3	e_5	e_2	e_7	e_7	e_7	
					e_7	v_4	v_5	F_3	e_4	e_5	e_6	e_6	e_6	

3D Object Representations



- Raw data
 - Voxels
 - Point cloud
 - Range image
 - Polygons
- Surfaces
 - Mesh
 - **Subdivision**
 - Parametric
 - Implicit
- Solids
 - Octree
 - BSP tree
 - CSG
 - Sweep
- High-level structures
 - Scene graph
 - Application specific

Gerry's Game

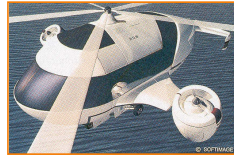


Pixar

Surfaces



- What makes a good surface representation?
 - Accurate
 - Concise
 - Intuitive specification
 - Local support
 - Affine invariant
 - Arbitrary topology
 - ~~Guaranteed continuity~~
 - Natural parameterization
 - Efficient display
 - Efficient intersections

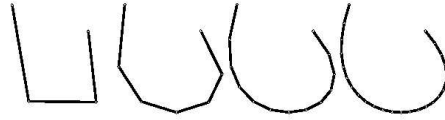


H&B Figure 10.46

Subdivision



- How do you make a smooth curve?

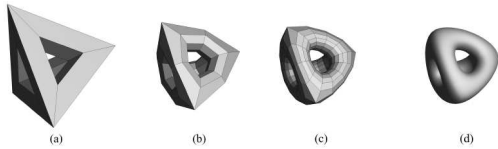


Zorin & Schroeder
SIGGRAPH 99
Course Notes

Subdivision Surfaces



- Coarse mesh & subdivision rule
 - Define smooth surface as limit of sequence of refinements

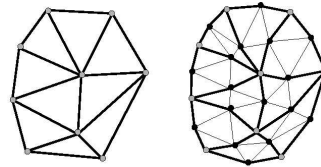


Zorin & Schroeder
SIGGRAPH 99
Course Notes

Key Questions



- How refine mesh?
 - Aim for properties like smoothness
- How store mesh?
 - Aim for efficiency for implementing subdivision rules

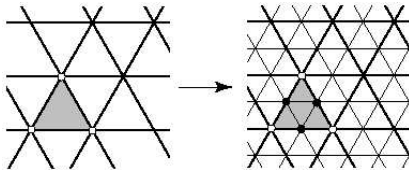


Zorin & Schroeder
SIGGRAPH 99
Course Notes

Loop Subdivision Scheme



- How refine mesh?
 - Refine each triangle into 4 triangles by splitting each edge and connecting new vertices
 - Need rules for "even / odd" (white / black) vertices

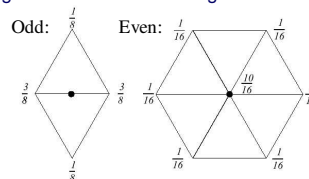


Zorin & Schroeder
SIGGRAPH 99
Course Notes

Loop Subdivision Scheme



- How position new vertices?
 - Choose locations for new vertices as weighted average of original vertices in local neighborhood



What if odd vertex only touches one triangle?

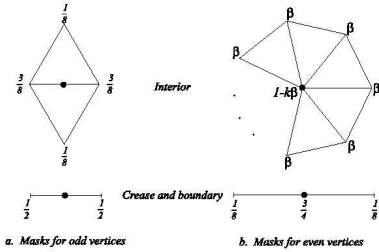
What if even vertex does not have degree 6?

Zorin & Schroeder
SIGGRAPH 99
Course Notes

Loop Subdivision Scheme



- Rules for extraordinary vertices and boundaries:



Zorin & Schroeder
SIGGRAPH 99
Course Notes

Loop



- How to choose β ?
 - Analyze properties of limit surface
 - Interested in continuity of surface and smoothness
 - Involves calculating eigenvalues of matrices

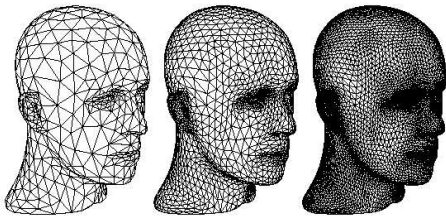
» Original Loop

$$\beta = \frac{1}{n} \left(\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4} \cos \frac{2\pi}{n} \right)^2 \right)$$

» Warren

$$\beta = \begin{cases} \frac{3}{8n} & n > 3 \\ \frac{3}{16} & n = 3 \end{cases}$$

Loop Subdivision Scheme



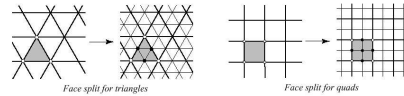
Limit surface has provable smoothness properties!

Zorin & Schroeder
SIGGRAPH 99
Course Notes

Subdivision Schemes



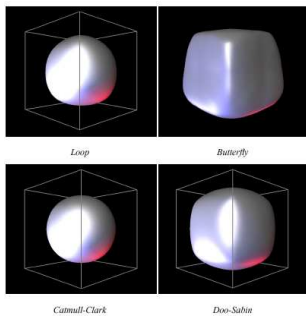
- There are different subdivision schemes
 - Different methods for refining topology
 - Different rules for positioning vertices
 - Interpolating versus approximating



Face split		Vertex split	
	Triangular meshes	Quad meshes	
Approximating	Loop (C^2)	Catmull-Clark (C^2)	Doo-Sabin, Midedge (C^1)
Interpolating	Mod. Butterfly (C^1)	Kobbelt (C^1)	Biquartic (C^2)

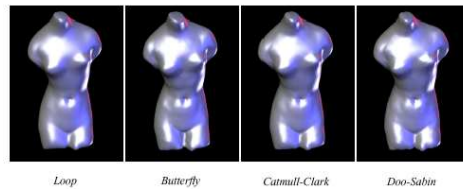
Zorin & Schroeder, SIGGRAPH 99, Course Notes

Subdivision Schemes



Zorin & Schroeder
SIGGRAPH 99
Course Notes

Subdivision Schemes



Zorin & Schroeder
SIGGRAPH 99
Course Notes

Subdivision Surfaces



- Properties:
 - Accurate
 - Concise
 - Intuitive specification
 - Local support
 - Affine invariant
 - Arbitrary topology
 - Guaranteed continuity
 - Natural parameterization
 - Efficient display
 - Efficient intersections

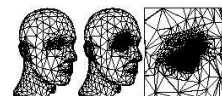


Pixar

Subdivision Surfaces



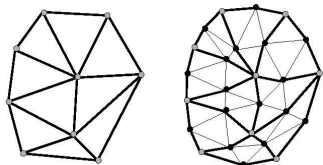
- Advantages:
 - Simple method for describing complex surfaces
 - Relatively easy to implement
 - Arbitrary topology
 - Local support
 - Guaranteed continuity
 - Multiresolution
- Difficulties:
 - Intuitive specification
 - Parameterization
 - Intersections



Key Questions



- How refine mesh?
 - Aim for properties like smoothness
- How store mesh?
 - Aim for efficiency for implementing subdivision rules

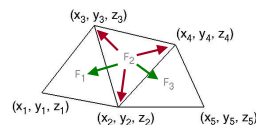


Zorin & Schroeder
SIGGRAPH 99
Course Notes

Triangle Meshes



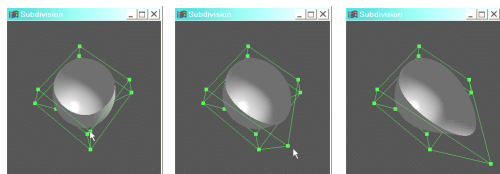
- Relevant properties:
 - Exactly 3 vertices per face
 - Any number of faces per vertex
- Useful adjacency structure for Loop subdivision:
 - Do not represent edges explicitly
 - Faces store refs to vertices and neighboring faces
 - Vertices store refs to adjacent faces and vertices



Assignment 3



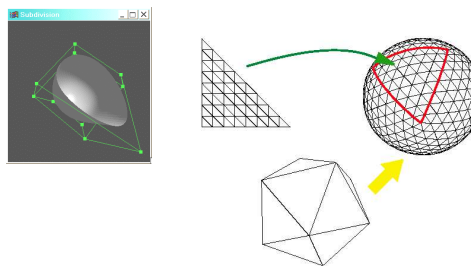
- Interactive editing of subdivision surfaces
 - Loop subdivision scheme
 - Partial adjacency list mesh representation
 - Interactive vertex dragging



Assignment 3



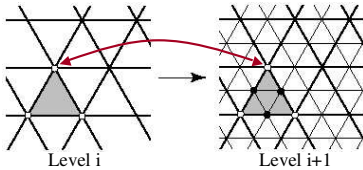
- Edit coarse mesh while display subdivided mesh



Assignment 3



- Store hierarchy of meshes
 - Full triangle mesh at every level
 - Vertices store references to counterparts one level up and one level down
 - Enables efficient re-positioning of mesh vertices after interactive dragging



Summary



Feature	Polygonal Mesh	Subdivision Surface
Accurate	No	Yes
Concise	No	Yes
Intuitive specification	No	No
Local support	Yes	Yes
Affine invariant	Yes	Yes
Arbitrary topology	Yes	Yes
Guaranteed continuity	No	Yes
Natural parameterization	No	No
Efficient display	Yes	Yes
Efficient intersections	No	No