Subdivision Surfaces

Thomas Funkhouser
Princeton University
C0S 426, Fall 2000

Course Syllabus

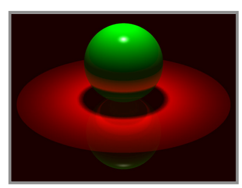
I. Image processing

II. Rendering

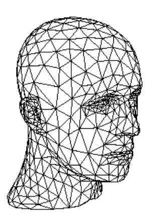
III. Modeling

IV. Animation

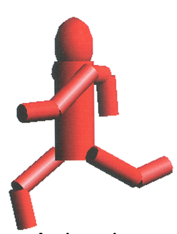
Image Processing
(Rusty Coleman, CS426, Fall99)



Rendering
(Michael Bostock, CS426, Fall99)



Modeling
(Dennis Zorin, CalTech)



Animation
(Angel, Plate 1)

Course Syllabus

I. Image processing

II. Rendering

III. Modeling

IV. Animation

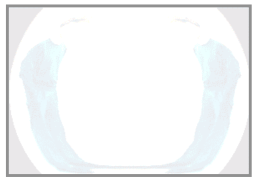
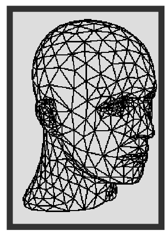
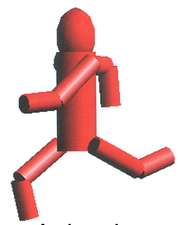


Image Processing
(Rusty Coleman, CS426, Fall99)

Rendering
(Michael Bostock, CS426, Fall99)



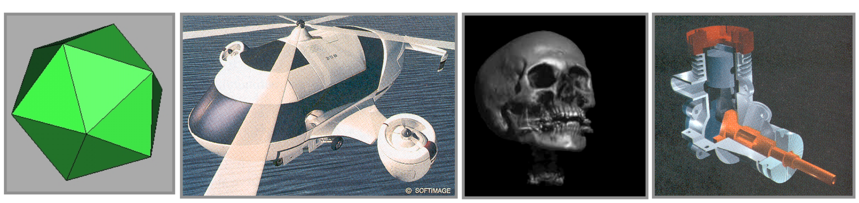
Modeling
(Dennis Zorin, CalTech)



Animation
(Angel, Plate 1)

Modeling

- How do we ...
  - Represent 3D objects in a computer?
  - Construct 3D representations quickly/easily?
  - Manipulate 3D representations efficiently?



Different representations for different types of objects

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
  - Skeleton
  - Application specific

Equivalence of Representations

- Thesis:
  - Each fundamental representation has enough expressive power to model the shape of any geometric object
  - It is possible to perform all geometric operations with any fundamental representation!
- Analogous to Turing-Equivalence:
  - All computers today are turing-equivalent, but we still have many different processors

Computational Differences

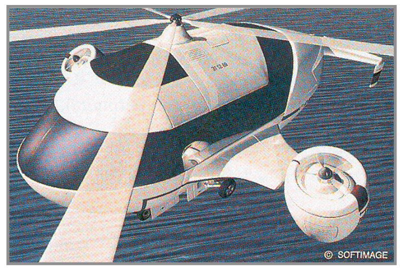
- Efficiency
  - Combinatorial complexity (e.g. O(n log n))
  - Space/time trade-offs (e.g. z-buffer)
  - Numerical accuracy/stability (degree of polynomial)
- Simplicity
  - Ease of acquisition
  - Hardware acceleration
  - Software creation and maintenance
- Usability
  - Designer interface vs. computational engine

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
  - Skeleton
  - 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

Subdivision Surface

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


Pixar

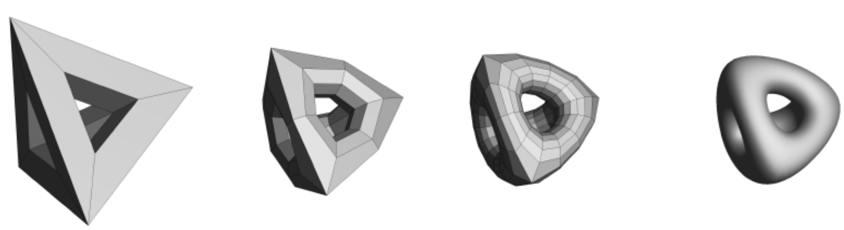
Subdivision

- 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

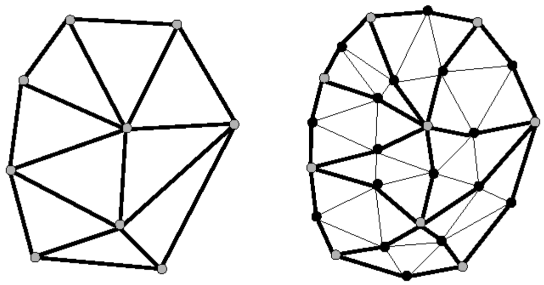


(a) (b) (c) (d)

Zorin & Schroeder
SIGGRAPH 99
Course Notes

Key Questions

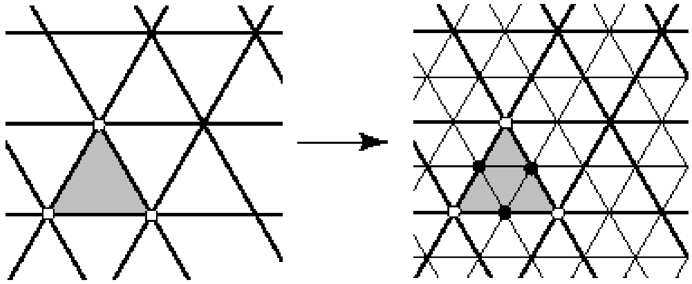
- How refine mesh?
  - Aim for properties like smoothness
- How store mesh?
  - Aim for efficiency for implementing subdivision rules



Zorin & Schroeder
SIGGRAPH 99
Course Notes

Loop Subdivision Scheme

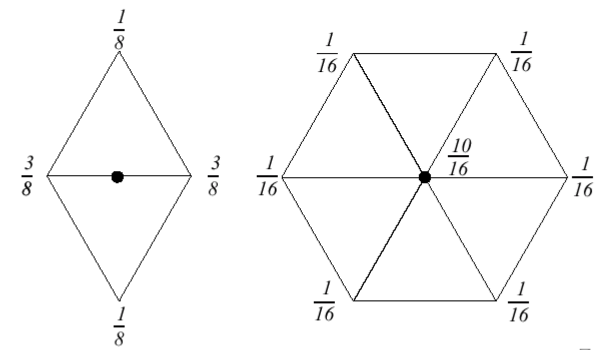
- How refine mesh?
  - Refine each triangle into 4 triangles by splitting each edge and connecting new vertices



Zorin & Schroeder
SIGGRAPH 99
Course Notes

Loop Subdivision Scheme

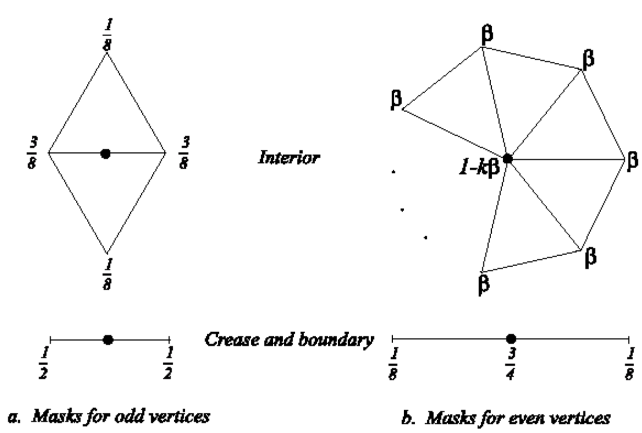
- How position new vertices?
  - Choose locations for new vertices as weighted average of original vertices in local neighborhood



Zorin & Schroeder
SIGGRAPH 99
Course Notes

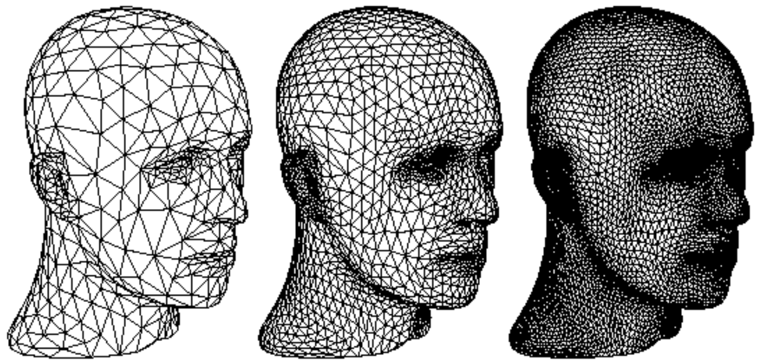
Loop Subdivision Scheme

- Different rules for boundaries:



Zorin & Schroeder
SIGGRAPH 99
Course Notes

Loop Subdivision Scheme



Limit surface has provable smoothness properties!

Zorin & Schroeder
SIGGRAPH 99
Course Notes

Subdivision Schemes

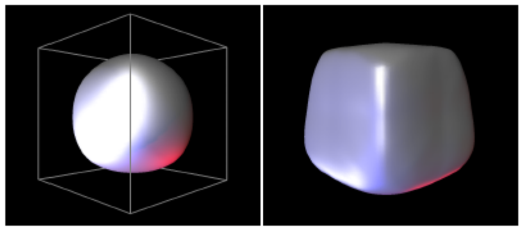
- There are different subdivision schemes
  - Different methods for refining topology
  - Different rules for positioning vertices
    - » Interpolating versus approximating

Face split		
	Triangular meshes	*Quad. meshes*
Approximating	Loop (C^2)	Catmull-Clark (C^2)
Interpolating	Mod. Butterfly (C^1)	Kobbelt (C^1)

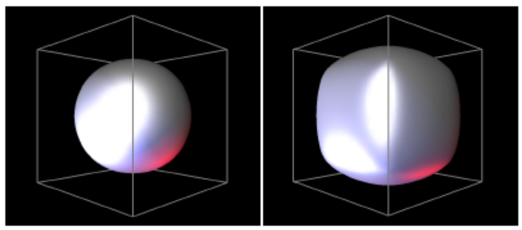
Vertex split
Doo-Sabin, Midedge (C^1)
Biquartic (C^2)

Zorin & Schroeder
SIGGRAPH 99
Course Notes

Subdivision Schemes



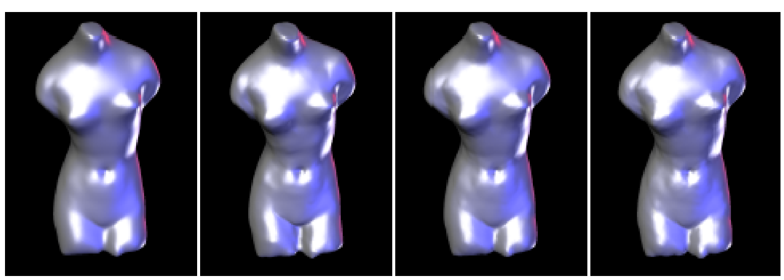
Loop *Butterfly*



Catmull-Clark *Doo-Sabin*

Zorin & Schroeder
SIGGRAPH 99
Course Notes

Subdivision Schemes

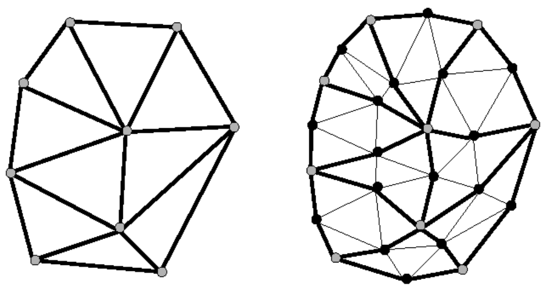


Loop *Butterfly* *Catmull-Clark* *Doo-Sabin*

Zorin & Schroeder
SIGGRAPH 99
Course Notes

Key Questions

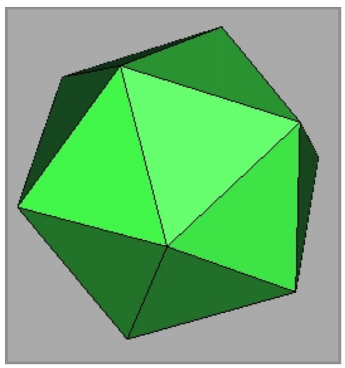
- How refine mesh?
  - Aim for properties like smoothness
- **How store mesh?**
  - Aim for efficiency for implementing subdivision rules



Zorin & Schroeder
SIGGRAPH 99
Course Notes

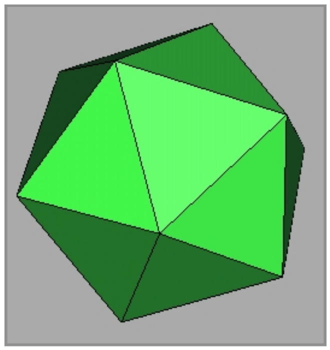
Polygon Meshes

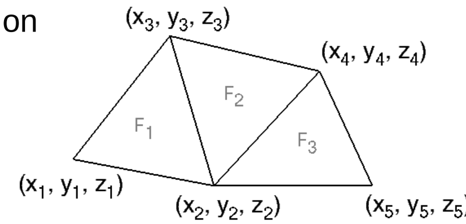
- Mesh Representations
  - Independent faces
  - Vertex and face tables
  - Adjacency lists
  - Winged-Edge



Independent Faces

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

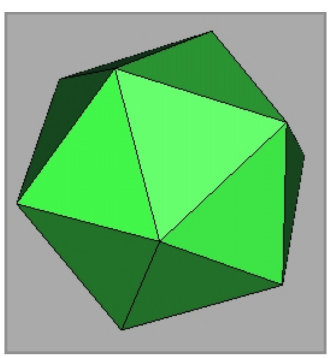


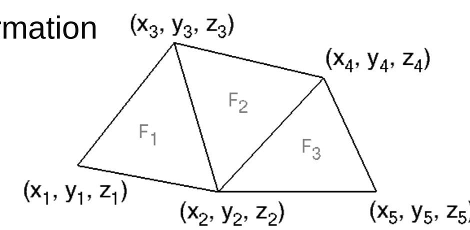


FACE TABLE	
F_1	(x_1, y_1, z_1) (x_2, y_2, z_2) (x_3, y_3, z_3)
F_2	(x_2, y_2, z_2) (x_4, y_4, z_4) (x_3, y_3, z_3)
F_3	(x_2, y_2, z_2) (x_5, y_5, z_5) (x_4, y_4, z_4)

Vertex and Face Tables

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



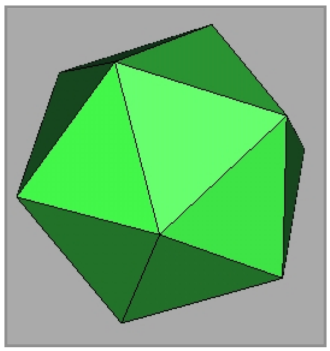


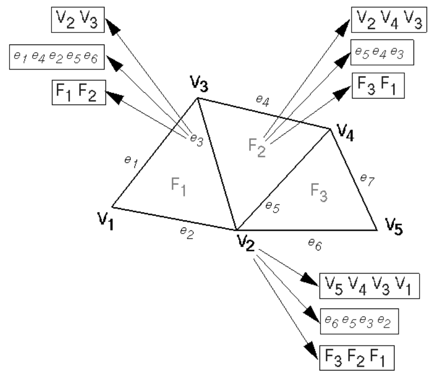
VERTEX TABLE			
V_1	X_1	Y_1	Z_1
V_2	X_2	Y_2	Z_2
V_3	X_3	Y_3	Z_3
V_4	X_4	Y_4	Z_4
V_5	X_5	Y_5	Z_5

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

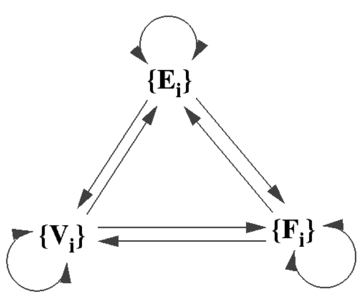
- Store all vertex, edge, and face adjacencies
  - Efficient topology traversal
  - Extra storage

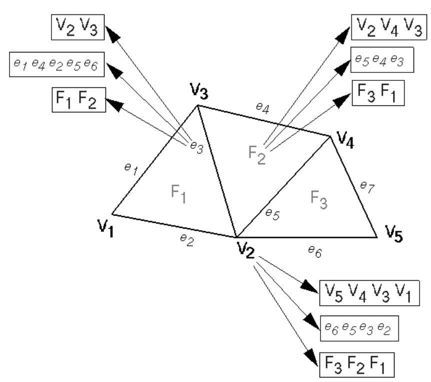




Partial Adjacency Lists

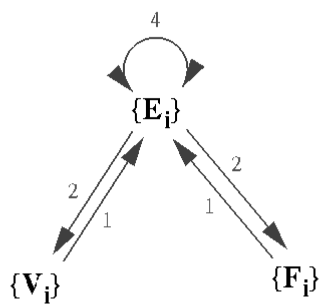
- Can we store only some adjacency relationships and derive others?

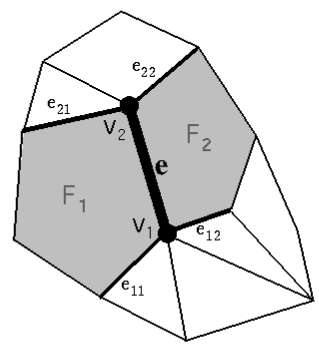




Winged Edge

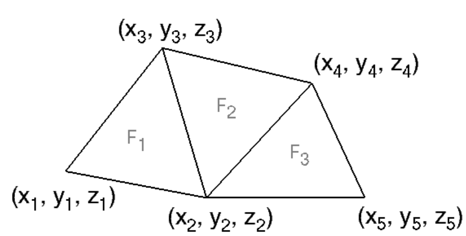
- Adjacency encoded in edges
  - All adjacencies in O(1) time
  - Little extra storage (fixed records)
  - Arbitrary polygons





Winged Edge

- Example:



VERTEX TABLE

V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

EDGE TABLE

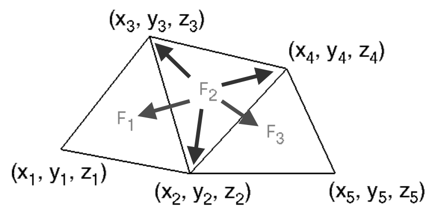
					11	12	21	22
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	V3	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

FACE TABLE

F_1	e_1
F_2	e_3
F_3	e_5

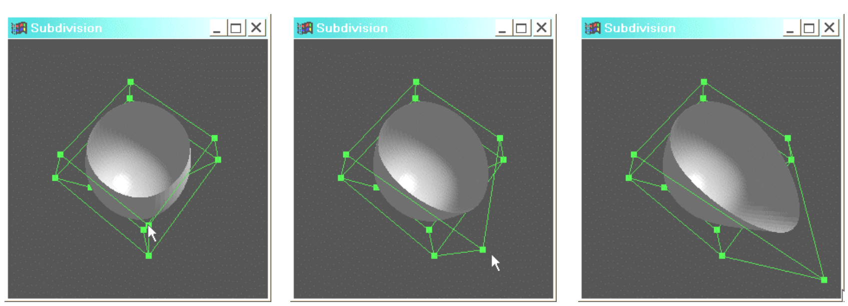
Triangle Meshes

- 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



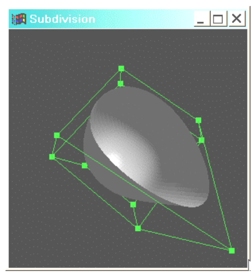
Assignment 4

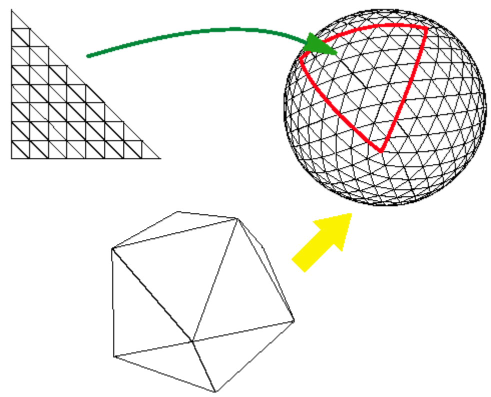
- Interactive editing of subdivision surfaces
  - Loop subdivision scheme
  - Partial adjacency list mesh representation
  - Interactive vertex dragging



Assignment 4

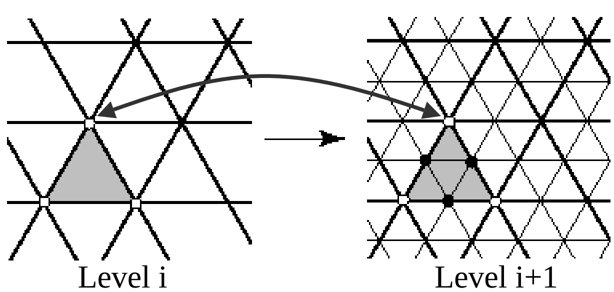
- Edit coarse mesh while display subdivided mesh





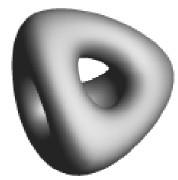
Assignment 4

- 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

- Advantages:
  - Simple method for describing complex surfaces
  - Relatively easy to implement
  - Arbitrary topology
  - Smoothness guarantees
  - Multiresolution



- Difficulties:
  - Intuitive specification
  - Parameterization
  - Intersections