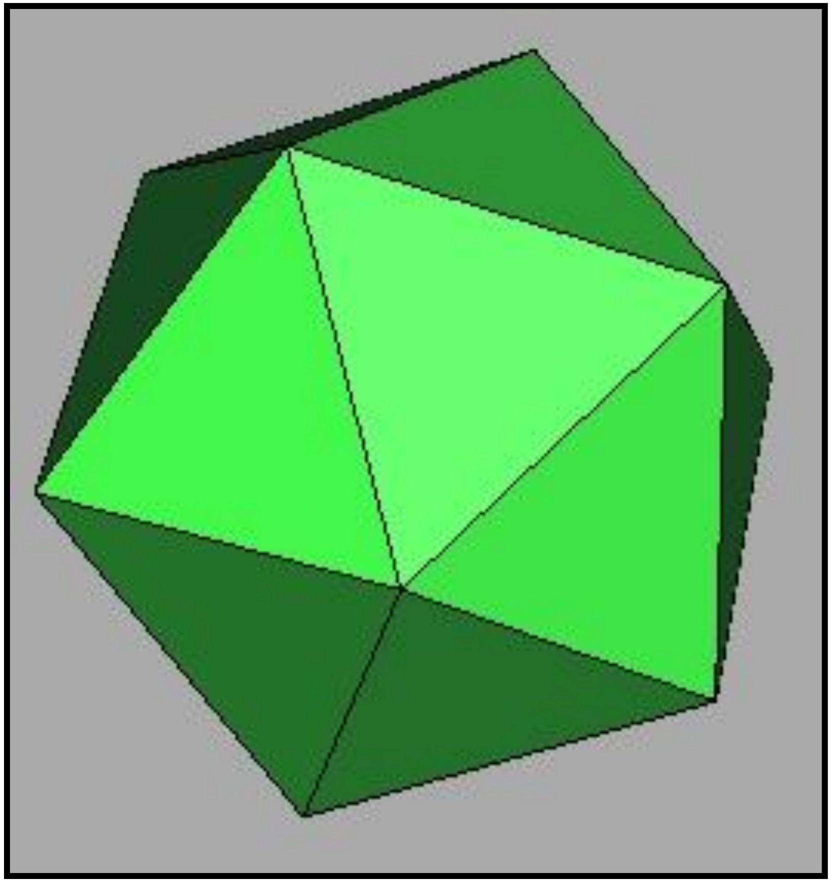
3D Object Representation (Loop) Subdivision Surfaces

Michael Kazhdan

(601.457/657)

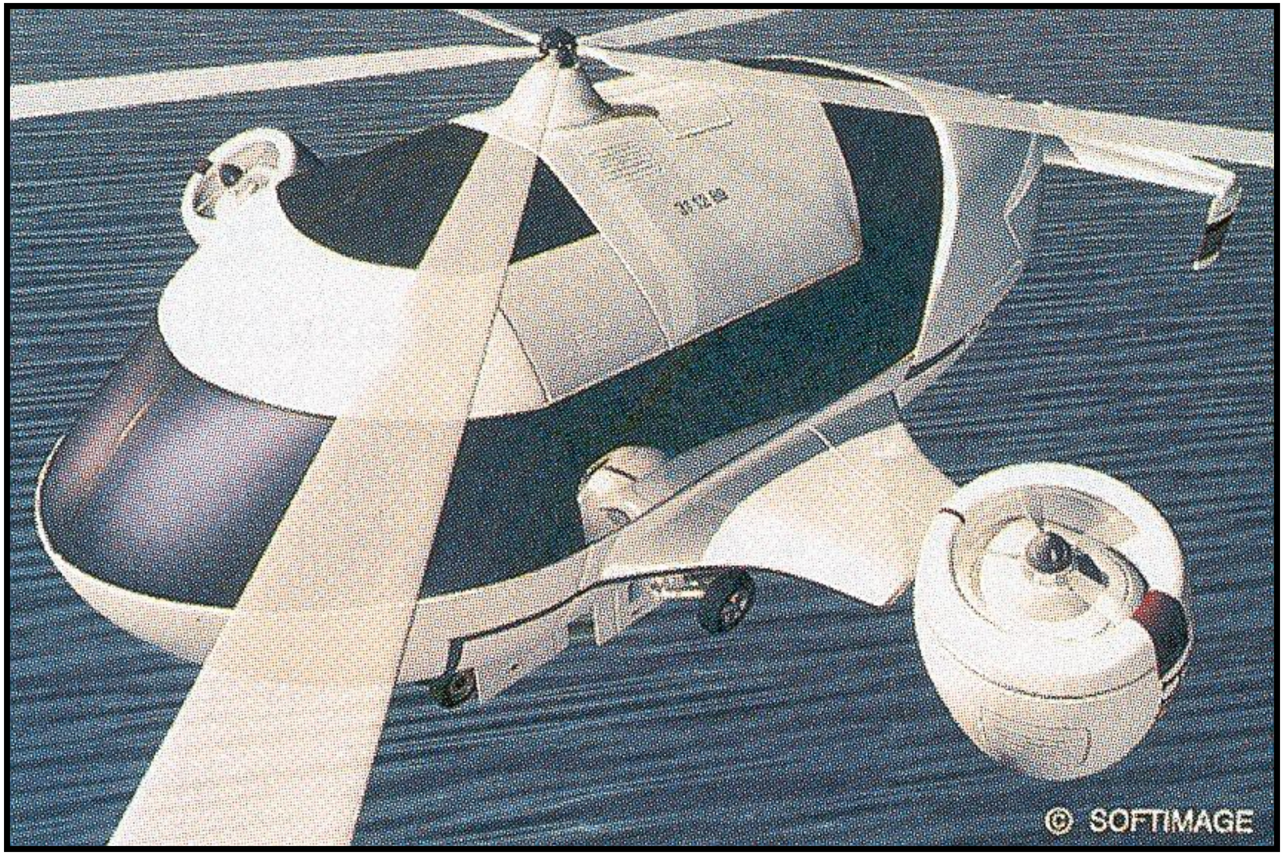
Subdivision for Modeling and Animation, Zorin and Schröder (2000)

3D Objects



How can this object be represented in a computer?

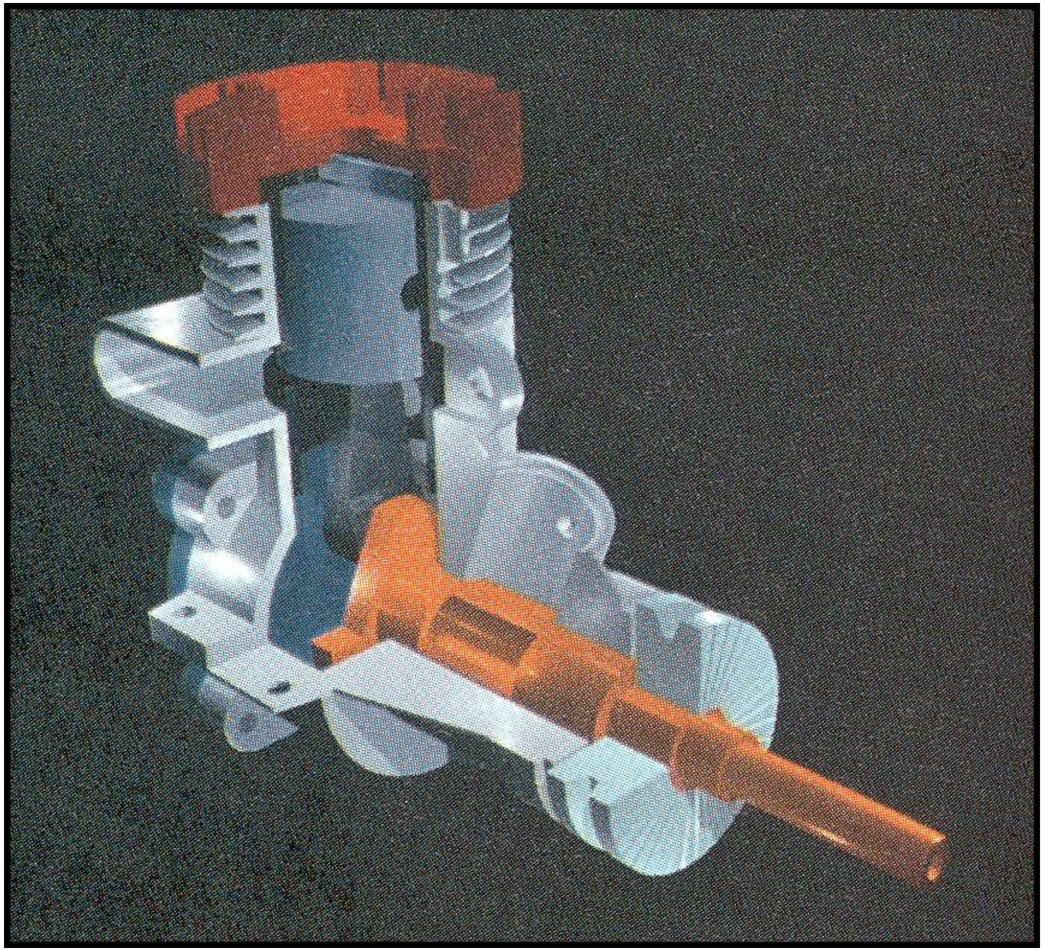
3D Objects



H&B Figure 10.46

This one?

3D Objects



H&B Figure 9.9

This one?

3D Object Representations

Raw data
- Point cloud
- Polygon soup
- Range image

Solids
- Implicit
- Voxels
- CSG

Surfaces
- Mesh
- Subdivision
- Parametric

High-level structures
- Scene graph
- Skeleton
- Application specific

Point Clouds

Unstructured set of 3D point samples

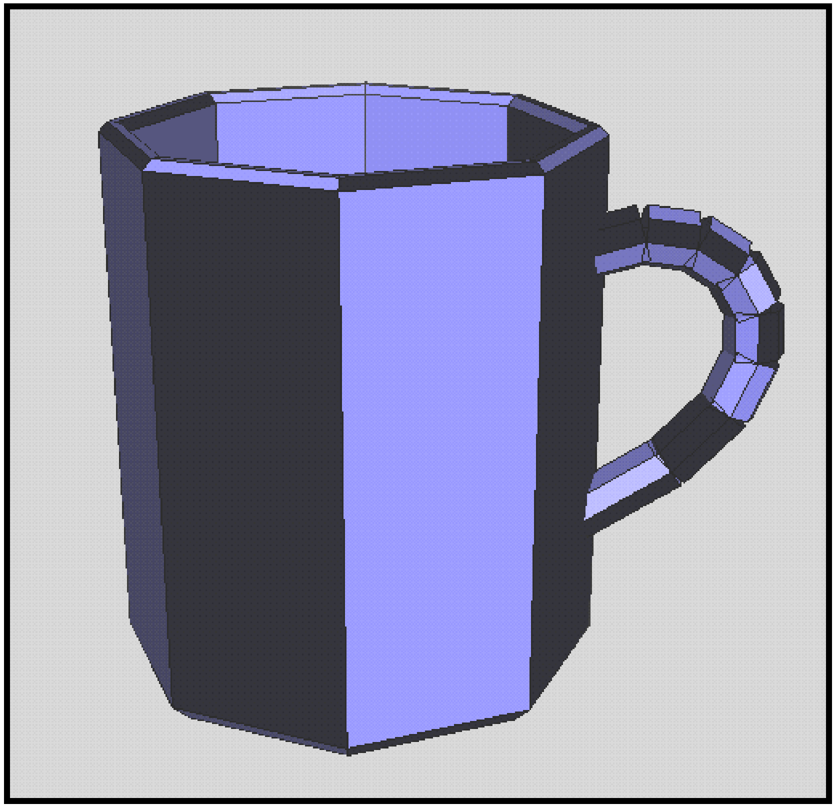
- Acquired from random sampling, particle system implementations, etc.

Hoppe

Polygon Soups

Unstructured set of polygons

- Created with interactive modeling systems

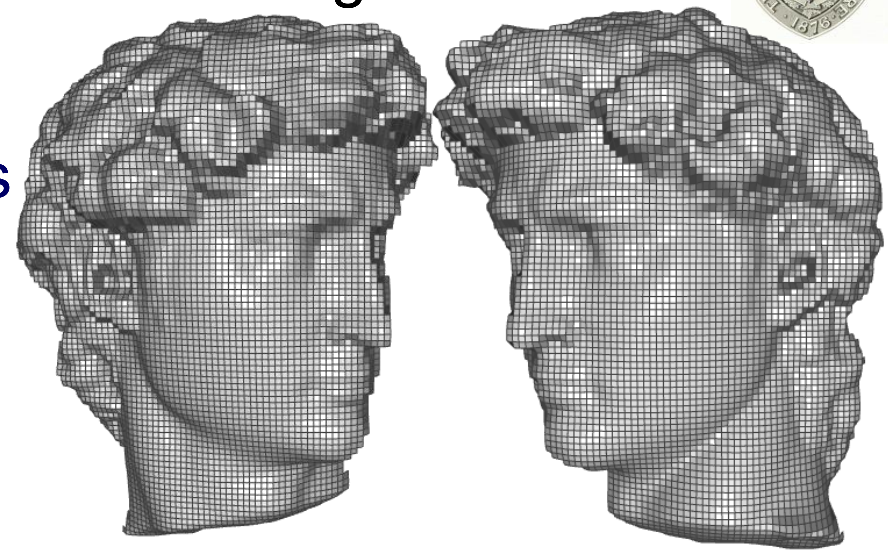


Larson

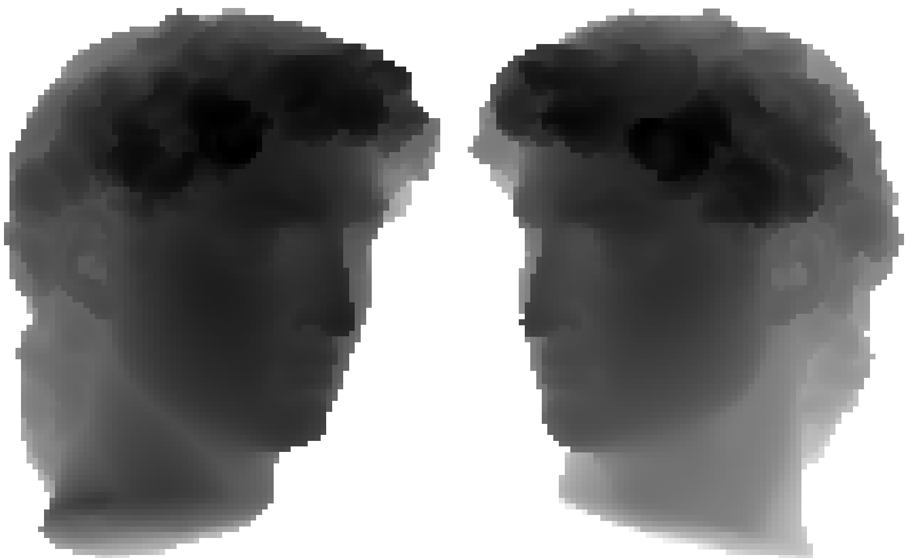
Range/Depth Image

An image storing depth

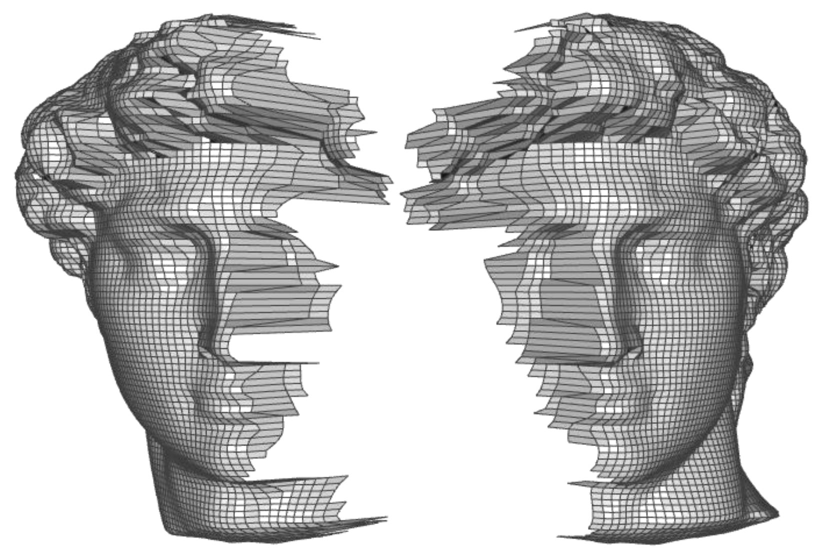
Acquired from 3D scanners



Range Surfaces



Range Images



Aligned Range Surfaces

3D Object Representations

Raw data
- Point cloud
- Polygon soup
- Range image

Solids
- Implicit
- Voxels
- CSG

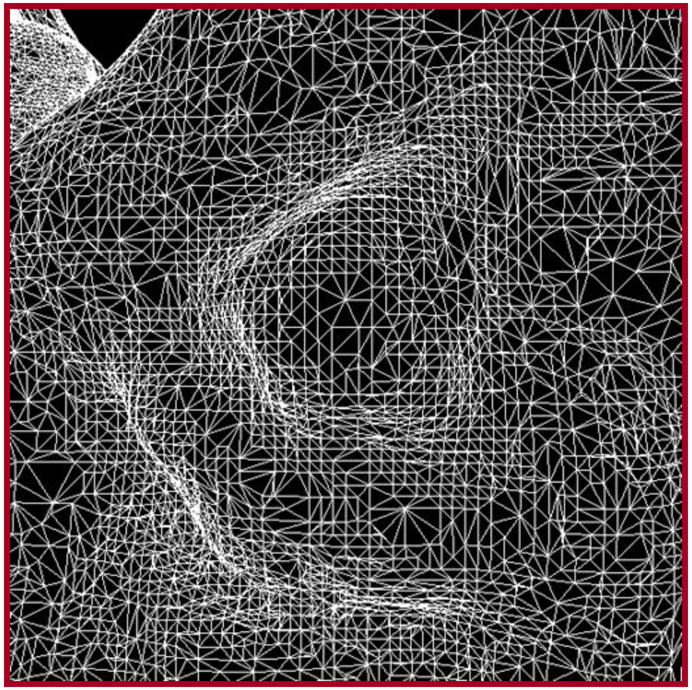
Surfaces
- Mesh
- Subdivision
- Parametric

High-level structures
- Scene graph
- Skeleton
- Application specific

(Manifold) Meshes

Connected set of polygons (usually triangles)

- Merging range images, etc.

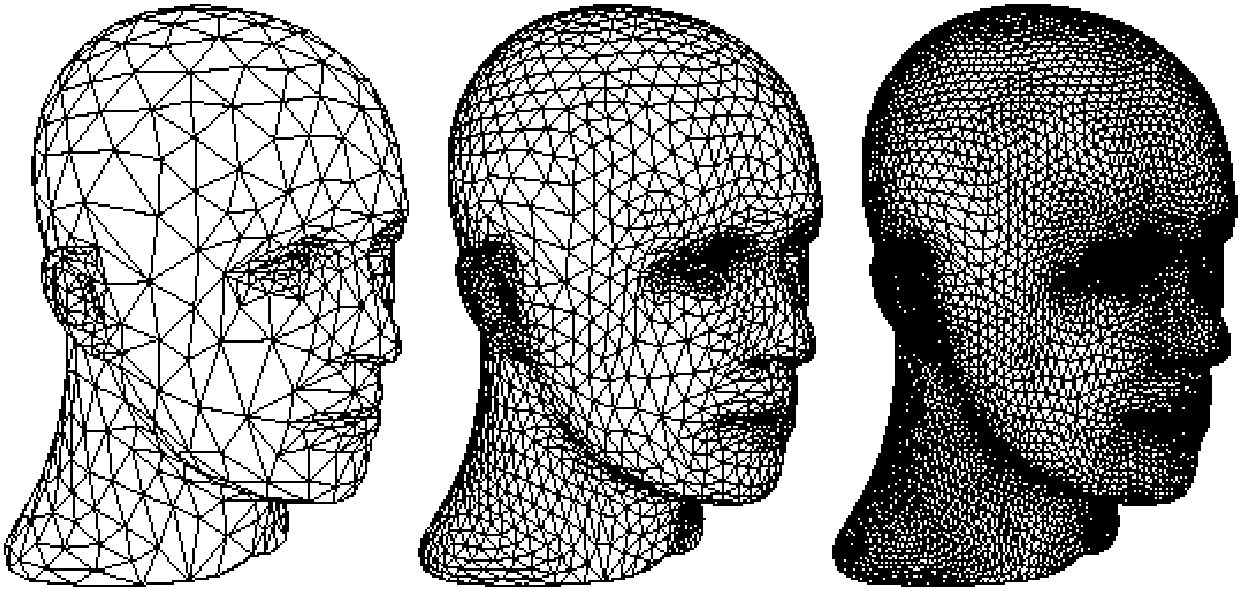


Stanford Graphics Laboratory

Subdivision Surfaces

Coarse mesh & subdivision rule

- Define a smooth surface as limit of a hierarchical sequence of refinements

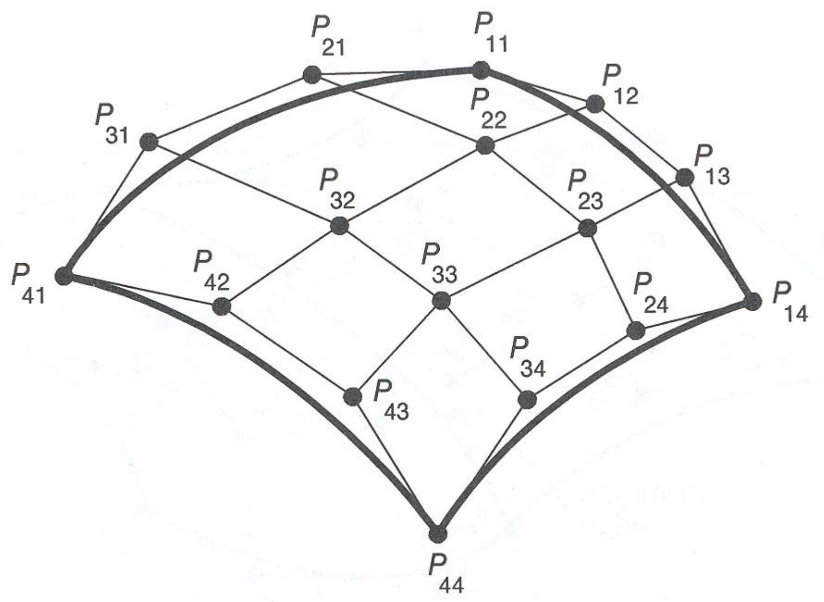


Zorin & Schroeder
SIGGRAPH 99
Course Notes

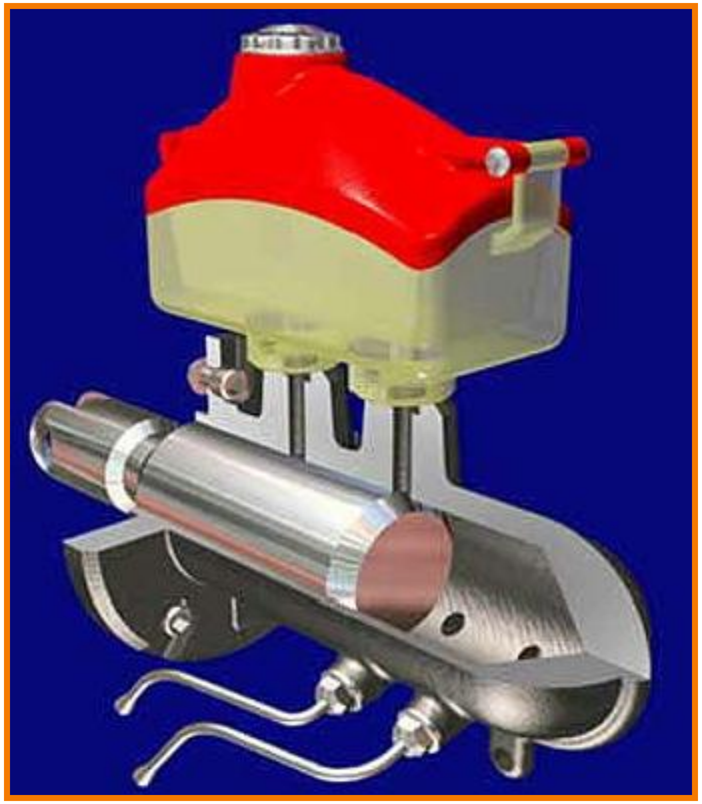
Parametric Surfaces

Tensor product spline patches

Used for real-world simulation



FvDFH Figure 11.44



3D Object Representations

Raw data
- Point cloud
- Polygon soup
- Range image

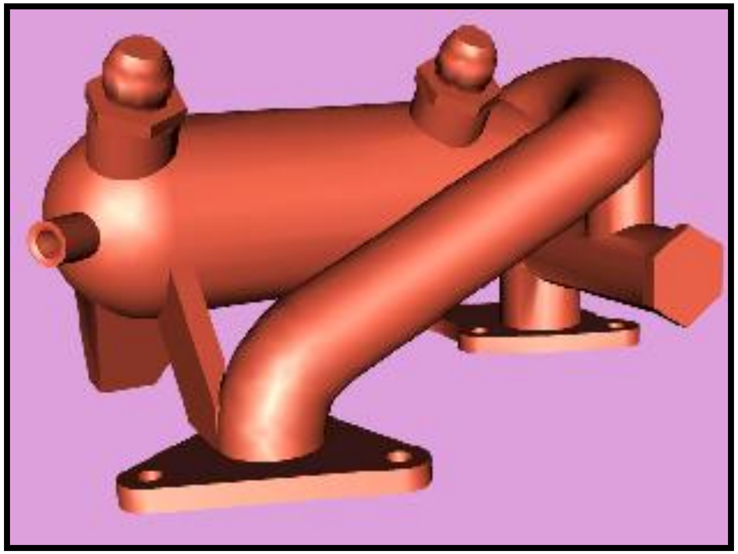
Solids
- Implicit
- Voxels
- CSG

Surfaces
- Mesh
- Subdivision
- Parametric

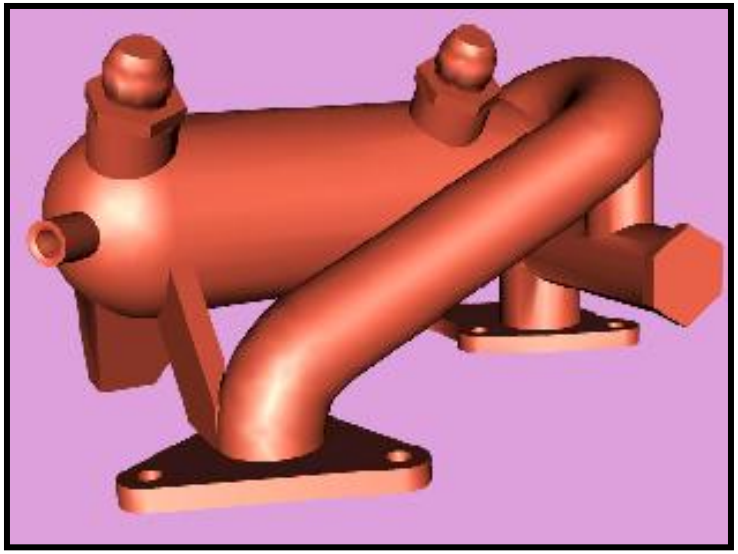
High-level structures
- Scene graph
- Skeleton
- Application specific

Implicit Surfaces

Points satisfying: $F(x, y, z) = 0$



Polygonal Model



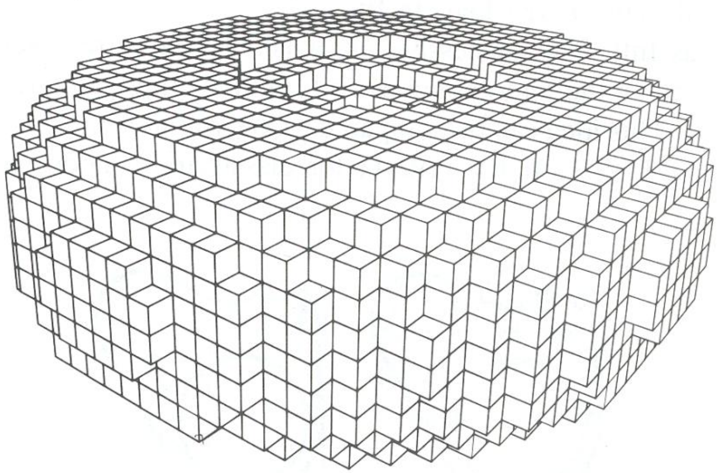
Implicit Model

Bill Lorensen
SIGGRAPH 99
Course #4 Notes

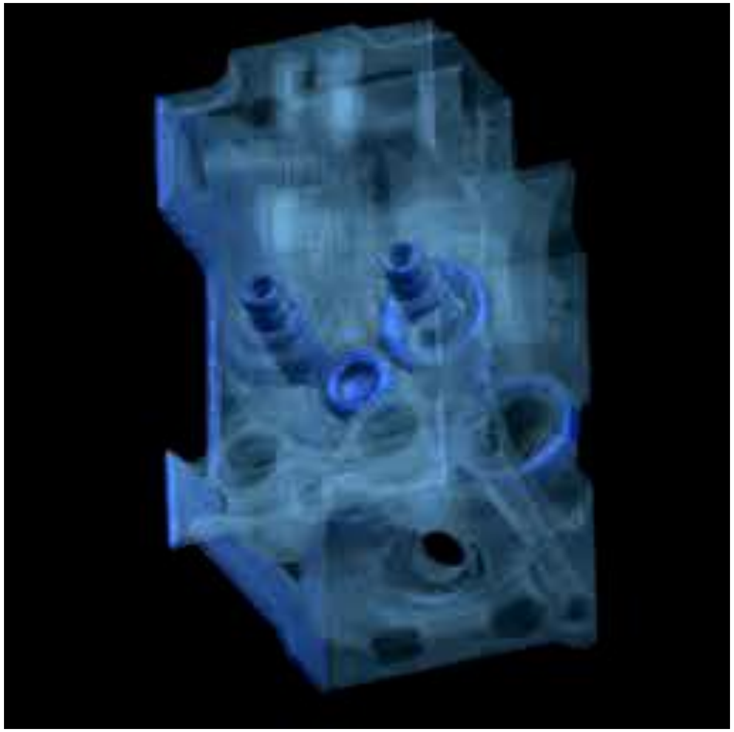
Voxels

Uniform grid of volumetric samples

- Acquired from CT, MRI, etc.



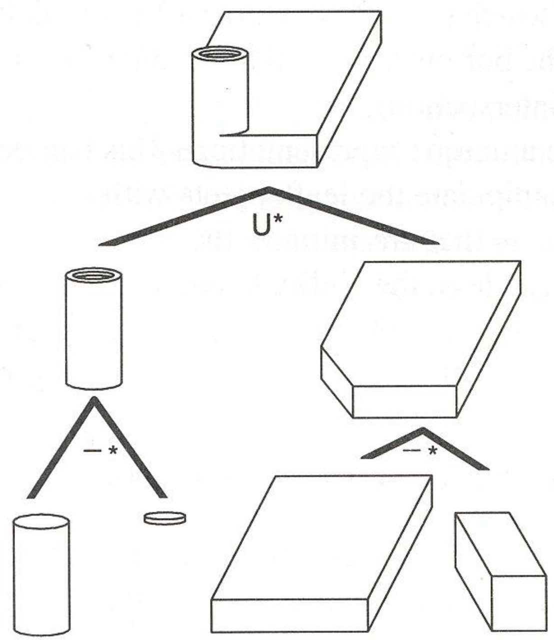
FvDFH Figure 12.20



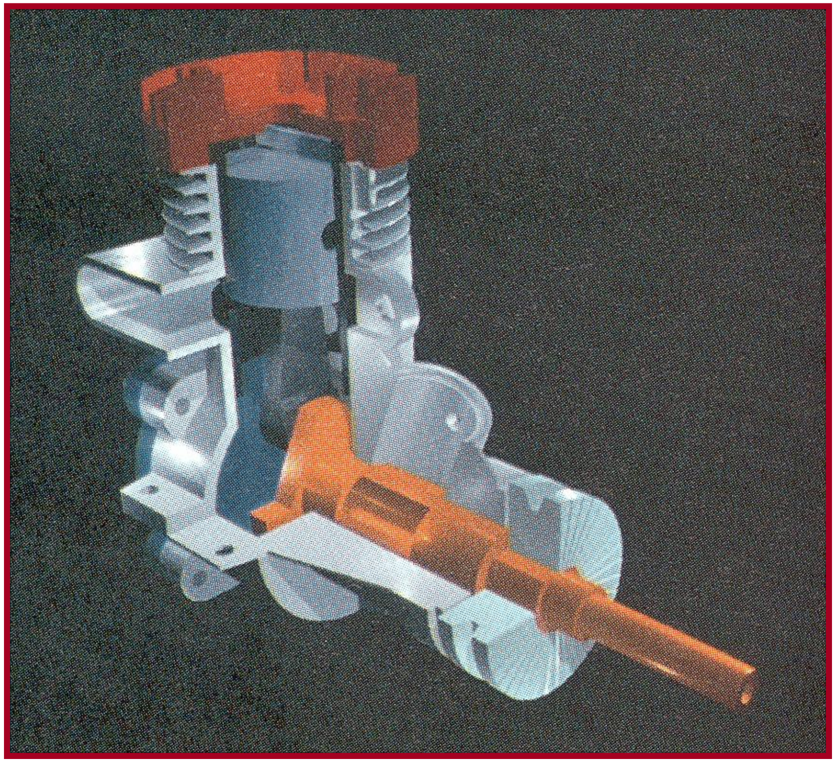
Stanford Graphics Laboratory

Constructive Solid Geometry (CSG)

Hierarchies of boolean set operations (union, difference, intersect) applied to simple shapes



FvDFH Figure 12.27



H&B Figure 9.9

3D Object Representations

Raw data
- Point cloud
- Polygon soup
- Range image

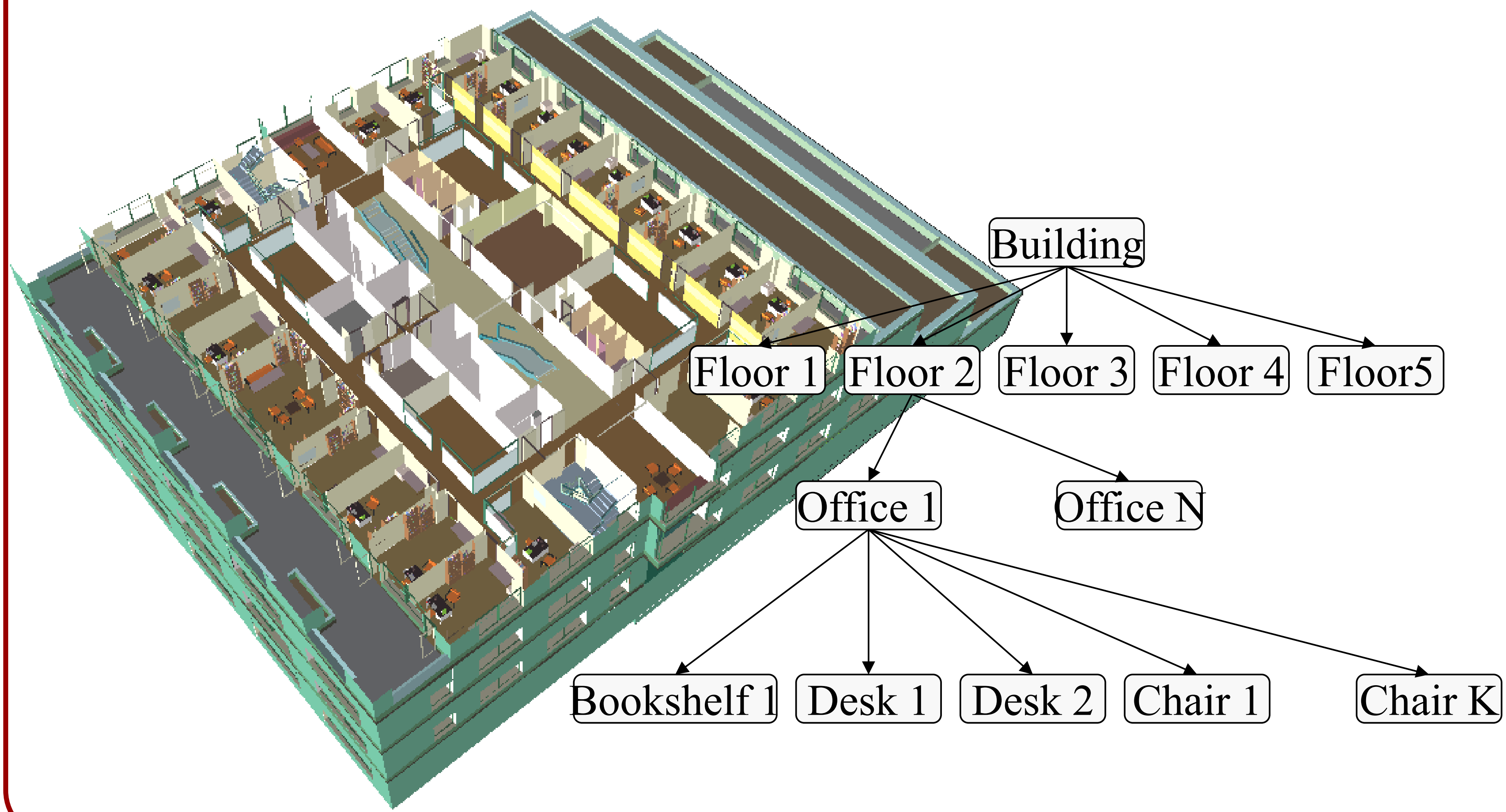
Solids
- Implicit
- Voxels
- CSG

Surfaces
- Mesh
- Subdivision
- Parametric

High-level structures
- Scene graph
- Skeleton
- Application specific

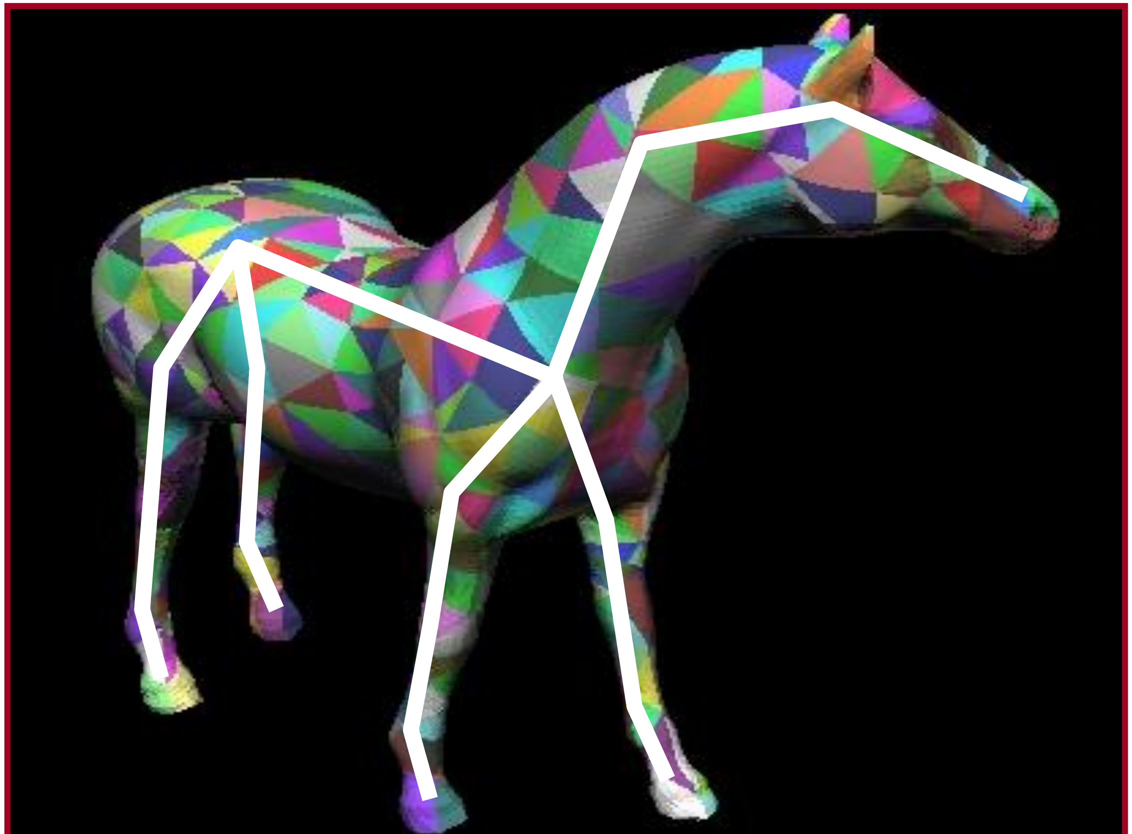
Scene Graphs

Union of objects at leaf nodes



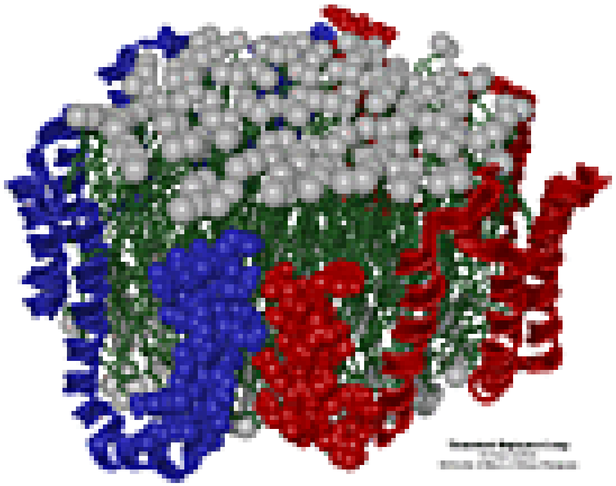
Skeletons

Graph of curves with geometry associated to individual curve positions



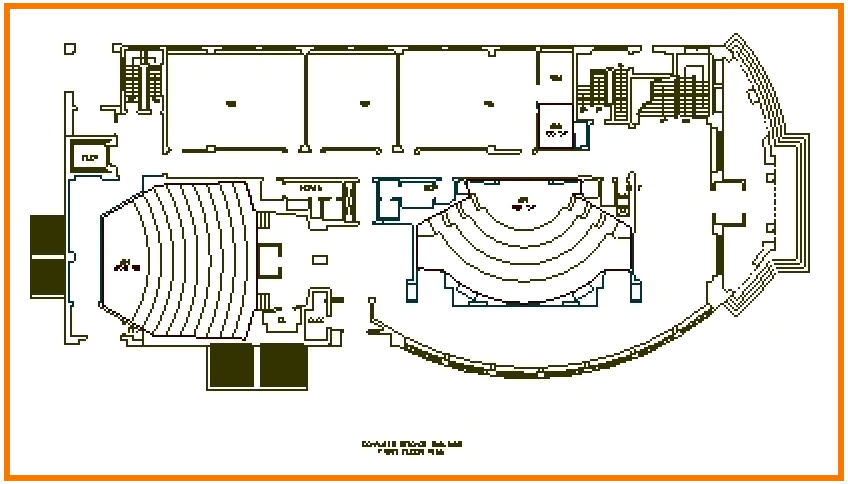
Stanford Graphics Laboratory

Application Specific



Apo A-1

(Theoretical Biophysics Group,
University of Illinois at Urbana-Champaign)



Architectural Floorplan

Surfaces

What makes a good surface representation?

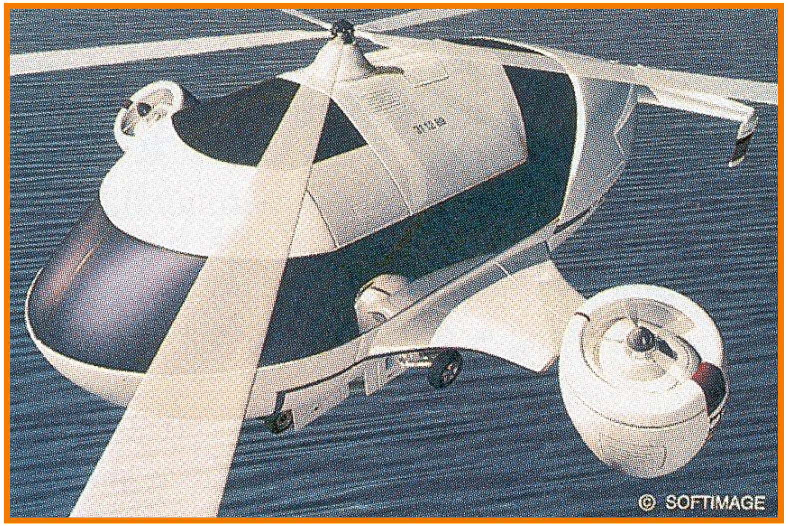
- Concise
- Local support
- Affine invariant
- Arbitrary topology
- Guaranteed smoothness
- Natural parameterization
- Efficient display
- Efficient intersections

Surfaces

What makes a good surface representation?

- Concise
- Local support
- Affine invariant
- Arbitrary topology
- Guaranteed smoothness
- Natural parameterization
- Efficient display
- Efficient intersections

smooth ≠ complex



H&B Figure 10.46

Surfaces

What makes a good surface representation?

- Concise
- Local support
- Affine invariant
- Arbitrary topology
- Guaranteed smoothness
- Natural parameterization
- Efficient display
- Efficient intersections

edits are localized

Not Local Support

Surfaces

What makes a good surface representation?

- Concise
- Local support
- Affine invariant
- Arbitrary topology
- Guaranteed smoothness
- Natural parameterization
- Efficient display
- Efficient intersections

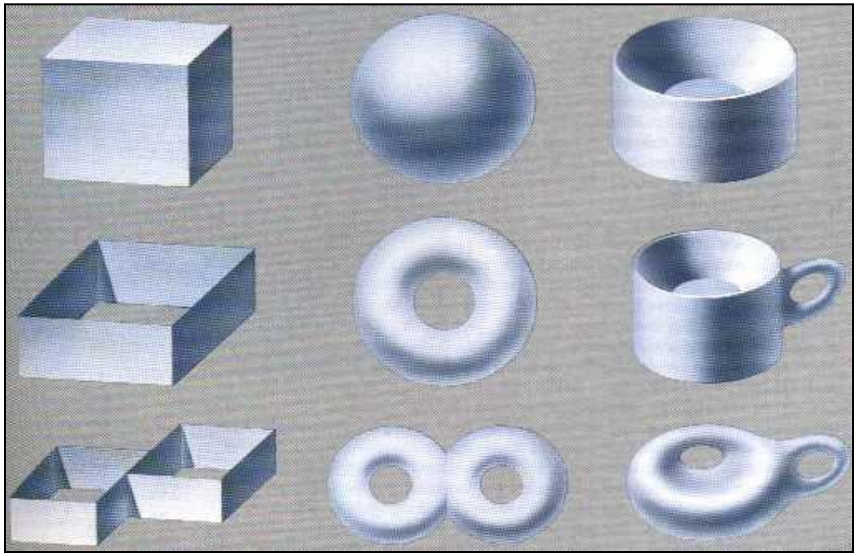
applying an affine transformation (linear+translation) to the surface does not fundamentally change its representation.

Surfaces

What makes a good surface representation?

- Concise
- Local support
- Affine invariant
- Arbitrary topology
- Guaranteed smoothness
- Natural parameterization
- Efficient display
- Efficient intersections

can represent surfaces with arbitrary on topology



Topological Genus Equivalences

Surfaces

What makes a good surface representation?

- Concise
- Local support
- Affine invariant
- Arbitrary topology
- Guaranteed smoothness
- Natural parameterization
- Efficient display
- Efficient intersections

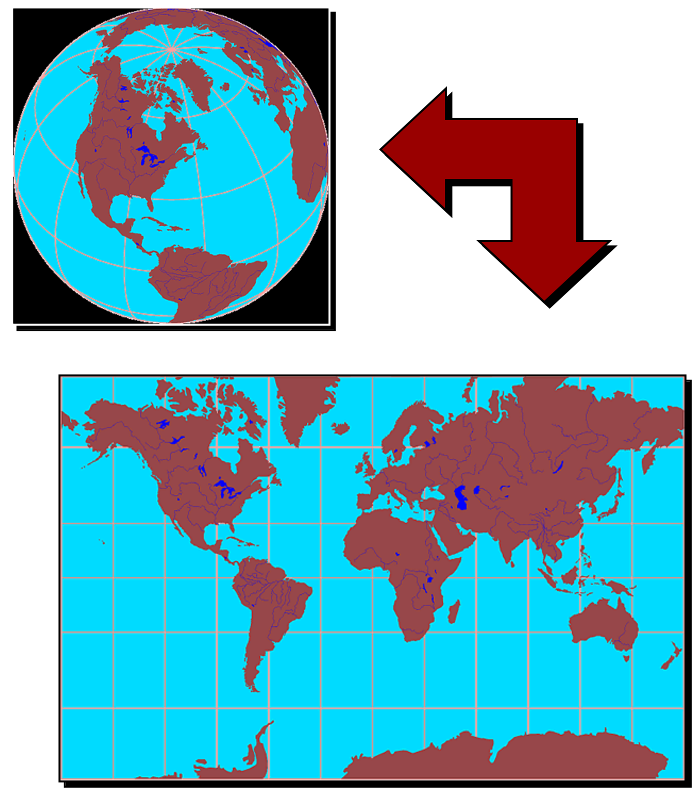
positions/normal vary continuously/smoothly over the surface

Surfaces

What makes a good surface representation?

- Concise
- Local support
- Affine invariant
- Arbitrary topology
- Guaranteed smoothness
- Natural parameterization
- Efficient display
- Efficient intersections

supports texture mapping



A Parameterization (not necessarily natural)

Surfaces

What makes a good surface representation?

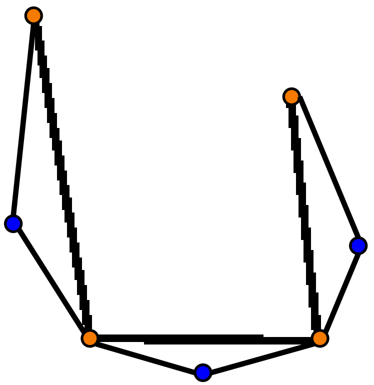
- Concise
- Local support
- Affine invariant
- Arbitrary topology
- Guaranteed smoothness
- Natural parameterization
- Efficient display
- Efficient intersections

supports efficient ray-tracing and/or real-time rendering

Subdivision

Q: How can we interpret a coarse set of samples as a smooth curve?

A: Introduce new in-between vertices that smooth out the severe angles

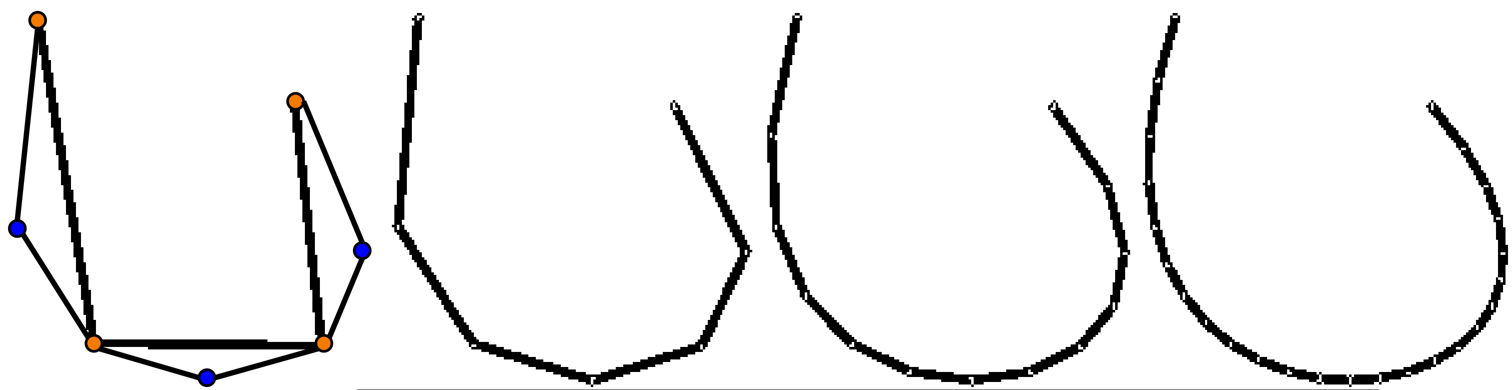


Zorin & Schroeder
SIGGRAPH 99
Course Notes

Subdivision

Q: How can we interpret a coarse set of samples as a smooth curve?

A: Introduce new in-between vertices that smooth out the severe angles



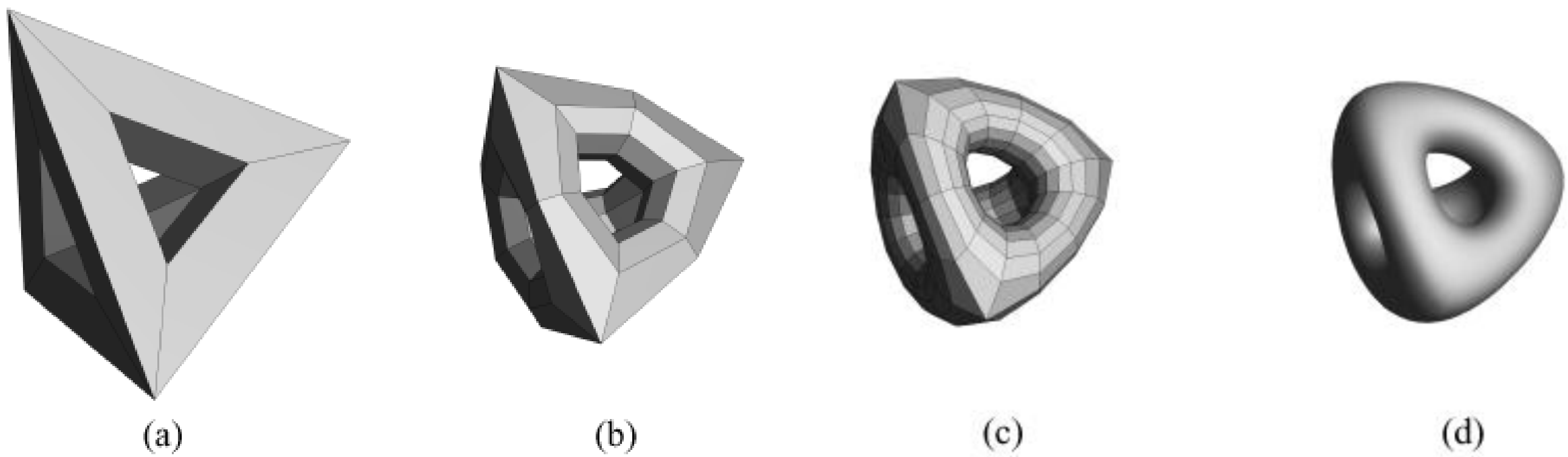
User: Specifies coarse geometry
Algorithm: Defines refined geometry

Zorin & Schroeder
SIGGRAPH 99
Course Notes

Subdivision Surfaces

Coarse mesh & subdivision rule

- Define smooth surface as limit of a sequence of refinements



Zorin & Schroeder
SIGGRAPH 99
Course Notes

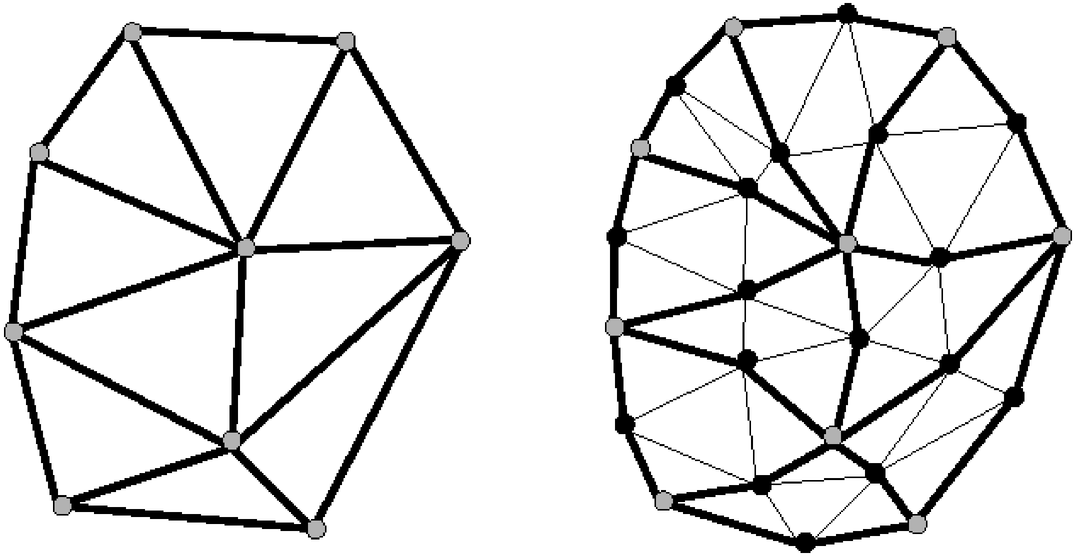
Key Questions

How to subdivide the mesh?

Aim for properties like smoothness

How to store the mesh? (Next time)

Aim for efficiency of implementing subdivision rules



Zorin & Schroeder
SIGGRAPH 99
Course Notes

General Subdivision Scheme

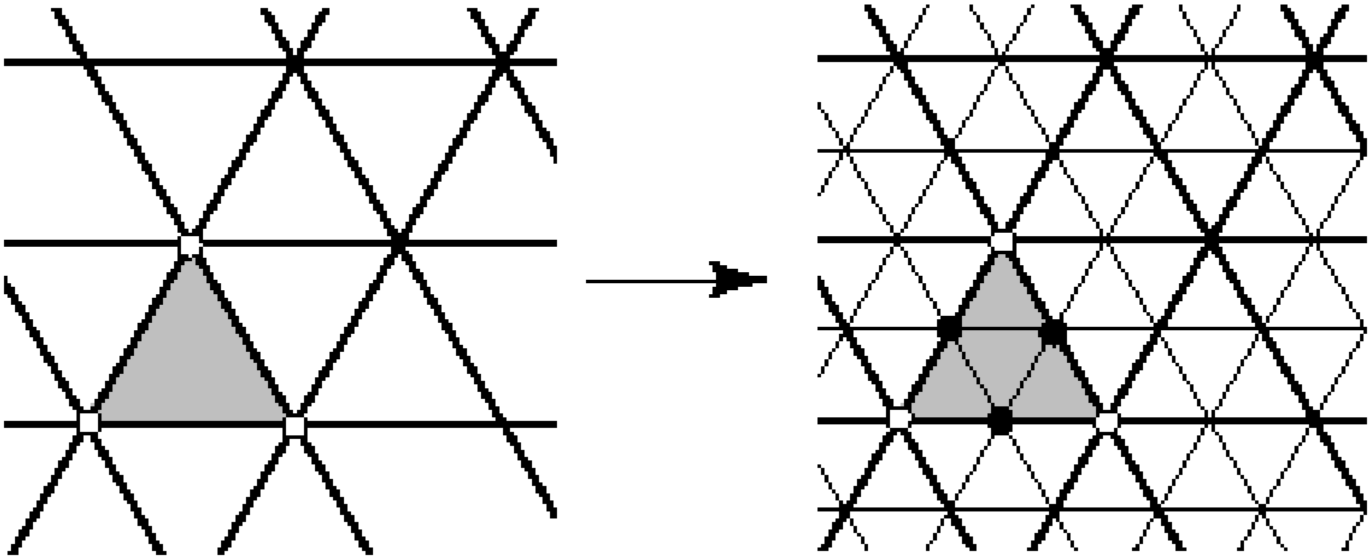
- How to subdivide the mesh?
  - Two parts:
    - Refinement (topology):
      - Add new vertices and connect
    - Smoothing (geometry):
      - Move vertex positions

Loop Subdivision Scheme

How to subdivide the mesh?

Refinement:

Subdivide each triangle into 4 by introducing edge mid-points and connecting the vertices



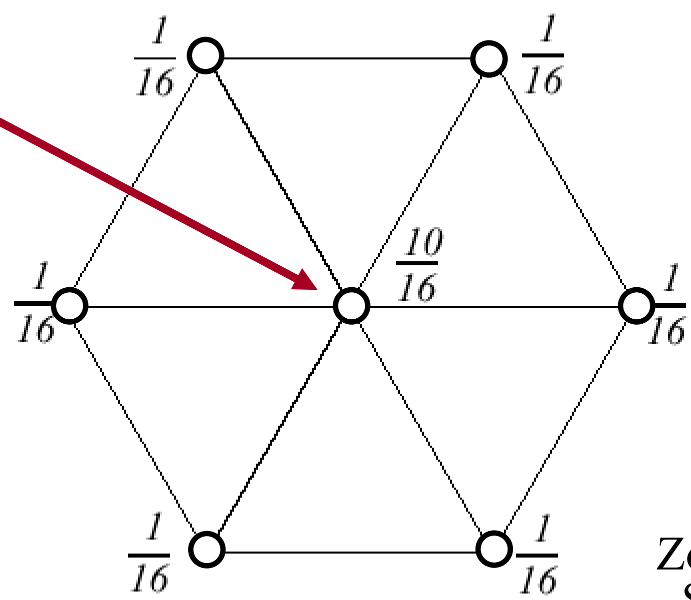
Zorin & Schroeder
SIGGRAPH 99
Course Notes

Loop Subdivision Scheme

How to subdivide the mesh?

- Refinement
- Smoothing (existing vertices):
  - Choose *new* location as weighted average of *original* vertex and its neighbors

Existing vertex being moved from one level to the next



Zorin & Schroeder
SIGGRAPH 99
Course Notes

Loop Subdivision Scheme

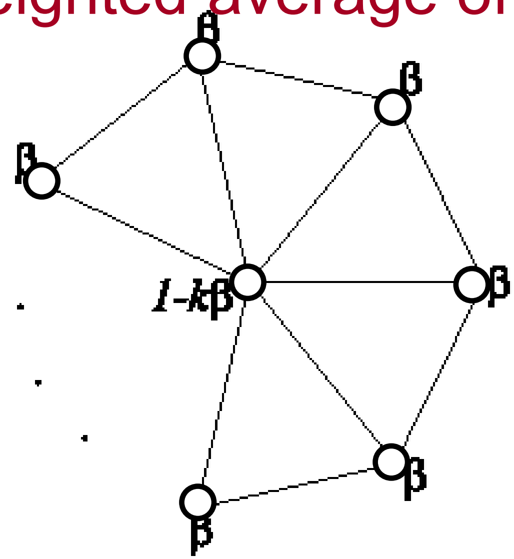
How to subdivide the mesh?

Refinement

Smoothing (existing vertices):

Choose *new* location as weighted average of *original* vertex and its neighbors

What about *extraordinary* vertices with more/less than 6 neighboring faces?



new_position = $(1\ -\ k\beta)$original_position + sum(β**each_original_vertex*)

Zorin & Schroeder
SIGGRAPH 99
Course Notes

Loop Subdivision Scheme

How to subdivide the mesh?

Refinement

Smoothing (existing vertices):

Choose *new* location as weighted average of *original* vertex and its neighbors

$\underline{0 \leq \beta \leq 1/k}$:

As β increases, the contribution from adjacent vertices plays a more important role.

If $\beta = 0$, the subdivision is interpolatory.

new_position = $(1 - k\beta)$original_position + sum(β**each_original_vertex*)

Zorin & Schroeder
SIGGRAPH 99
Course Notes

Loop Subdivision Scheme

<u>Challenge</u>:

Choose β so that the limit surface has guaranteed smoothness properties

Original Loop

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

Warren

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

Why are valence-6 vertices "regular"?

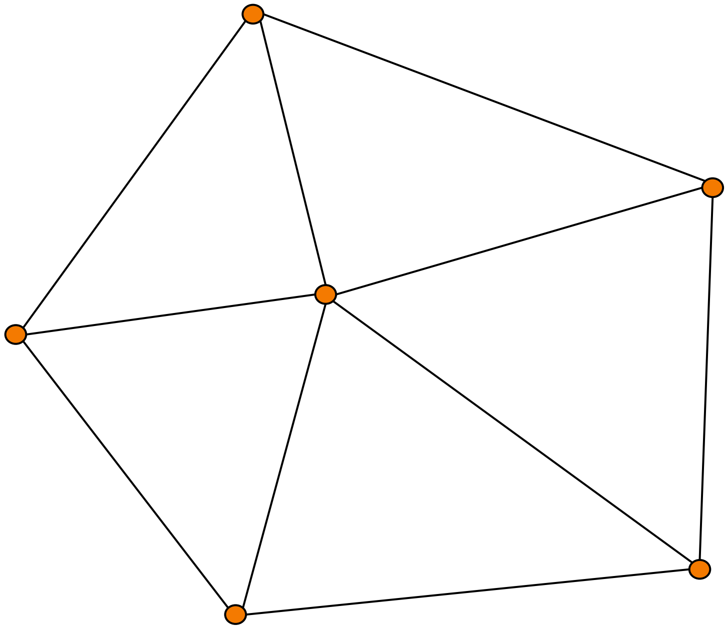
Definition:

Given an undirected graph, the *valence* of a vertex/node in the graph is the number of edges emanating from it.

Why are valence-6 vertices "regular"?

Subdivision:

Q: What happens after we refine?



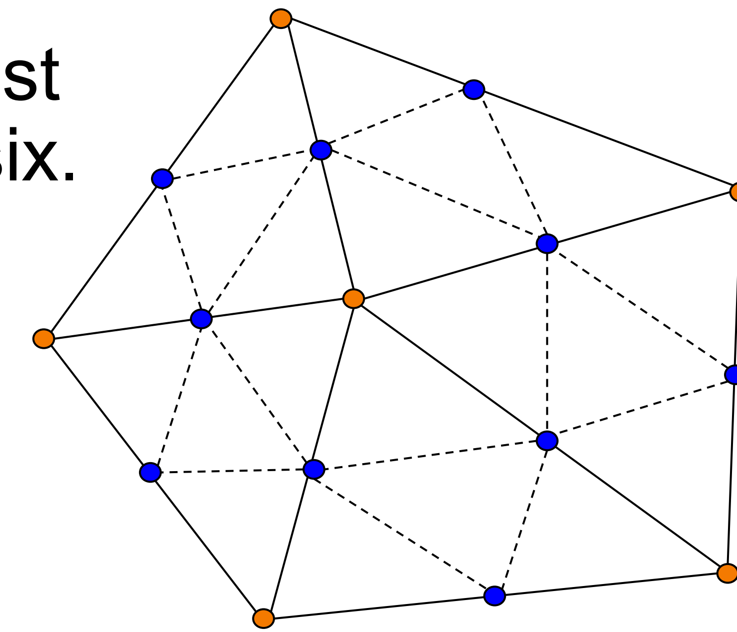
Why are valence-6 vertices "regular"?

Subdivision:

Q: What happens after we refine?

A: Valence of old vertices is unchanged.
Valence of new vertices is six.

⇒ As we continue refining most vertices will have valence six.



Why are valence-6 vertices "regular"?

Euler Characteristic:

For connected, water-tight meshes, the number of vertices, edges, and faces satisfy:

$$|V| - |E| + |F| = 2 - 2g$$

where g is the genus of the surface (how many topological holes it has).

For water-tight triangle meshes, each face has three edges and each edge is shared by two faces, so the number of edges is

$$|E| = \frac{3}{2}|F|$$

Why are valence-6 vertices "regular"?

Euler Characteristic:

$$|V| - |E| + |F| = 2 - 2g$$

For water-tight triangle meshes:

$$|E| = \frac{3}{2}|F|$$

Putting this together we get:

$$|V| - |E| + \frac{2}{3}|E| = 2 - 2g$$

$$|V| - \frac{1}{3}|E| = 2 - 2g$$

$$3|V| \approx |E|$$

Why are valence-6 vertices "regular"?

$$3|V| \approx |E|$$

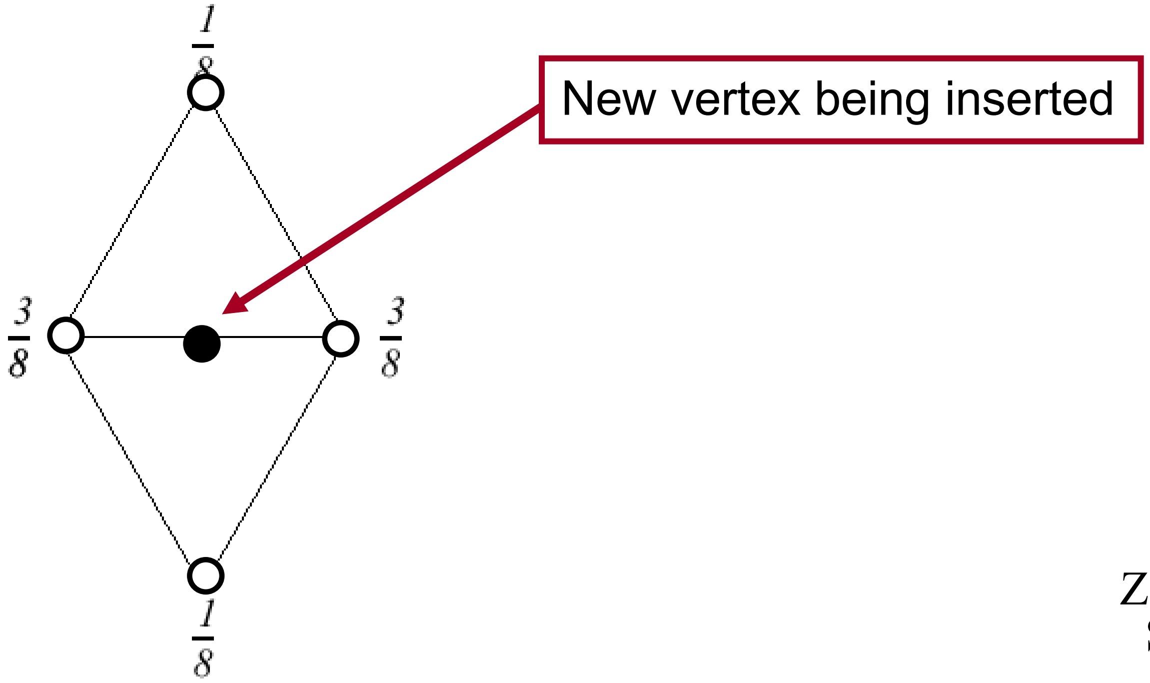
$$\Downarrow$$

$$\begin{aligned} Average\ Valence &= \frac{1}{|V|}\sum_{v\in V} valence(v) \\ &= \frac{1}{|V|}(2|E|) \\ &\approx \frac{1}{|V|}(6|V|) \\ &= 6 \end{aligned}$$

Loop Subdivision Scheme

How to subdivide the mesh?

- Refinement
- Smoothing (inserted vertices):
  - Choose location as weighted average of *original* vertices in local neighborhood

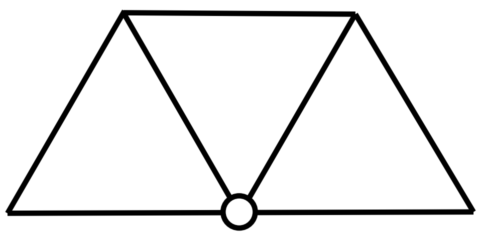


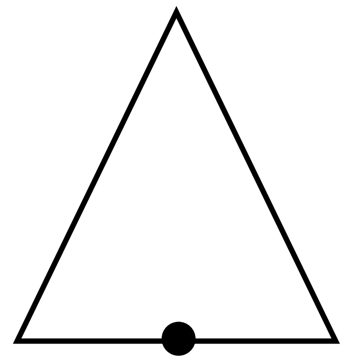
Zorin & Schroeder
SIGGRAPH 99
Course Notes

Boundary Cases?

What about *boundary vertices / edges?*

- Existing vertex adjacent to an incomplete "triangle fan"
- New vertex bordered by only one triangle

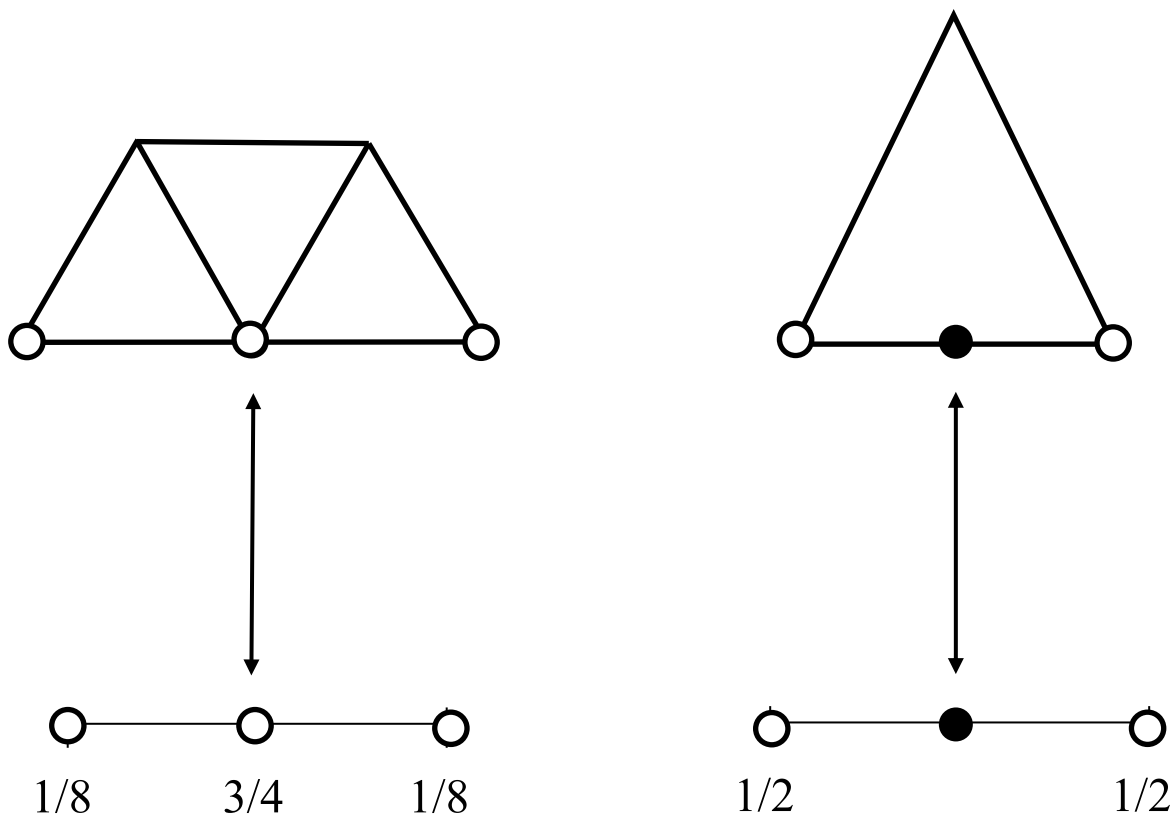




Boundary Cases?

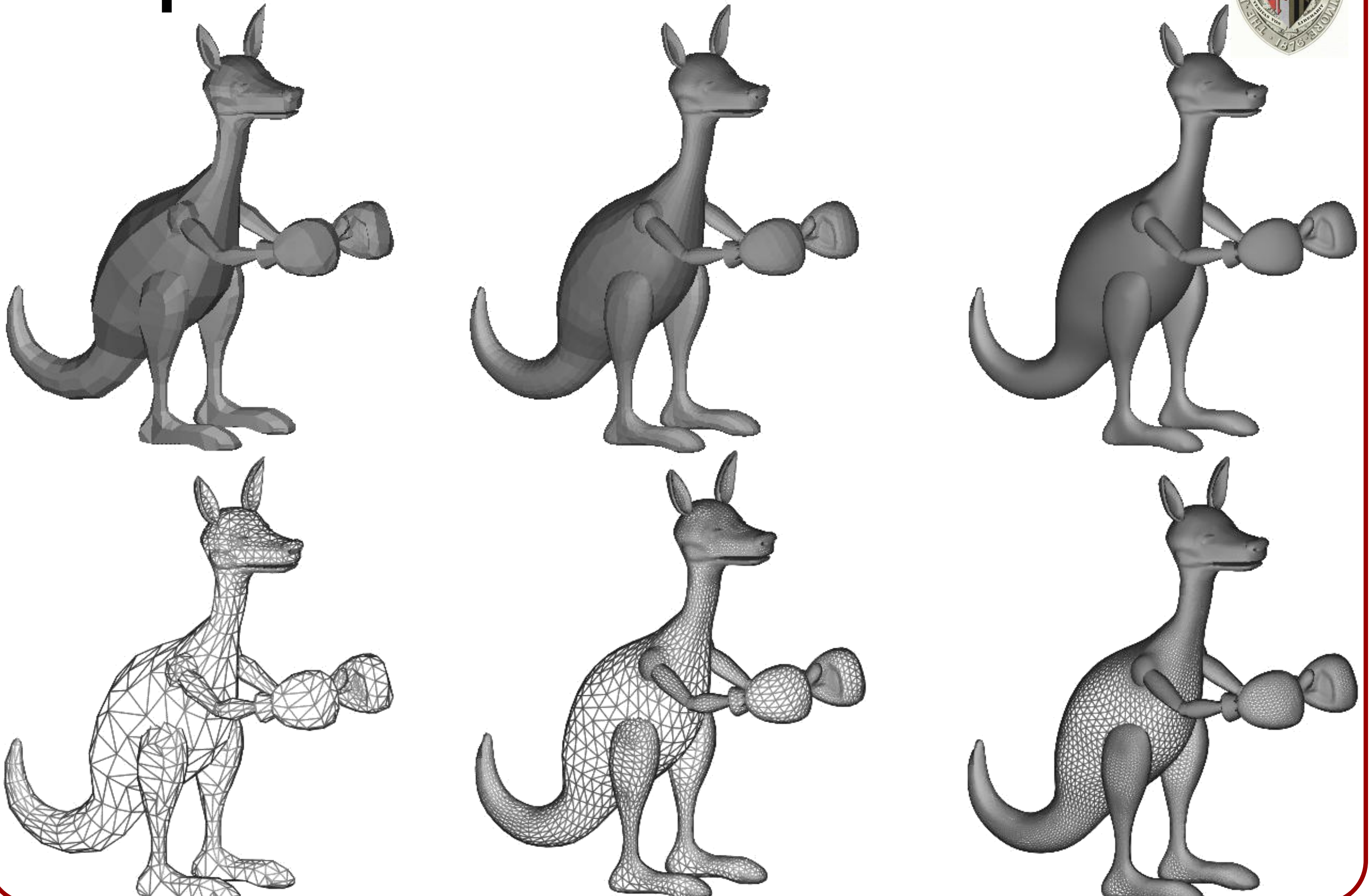
What about *boundary vertices / edges?*

Refine as though the vertices/edges are on the (boundary) curve



Zorin & Schroeder SIGGRAPH 99 Course Notes

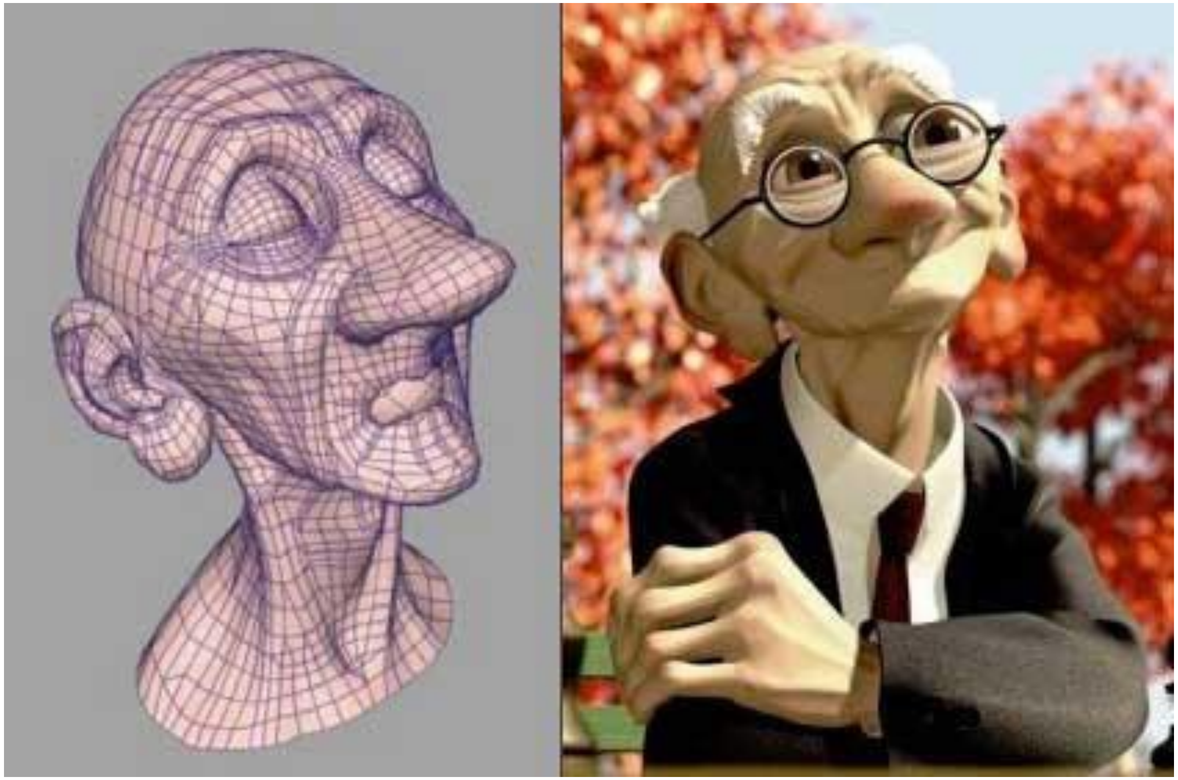
Loop Subdivision Scheme



Loop Subdivision Scheme

Geri’s Game, *Pixar*

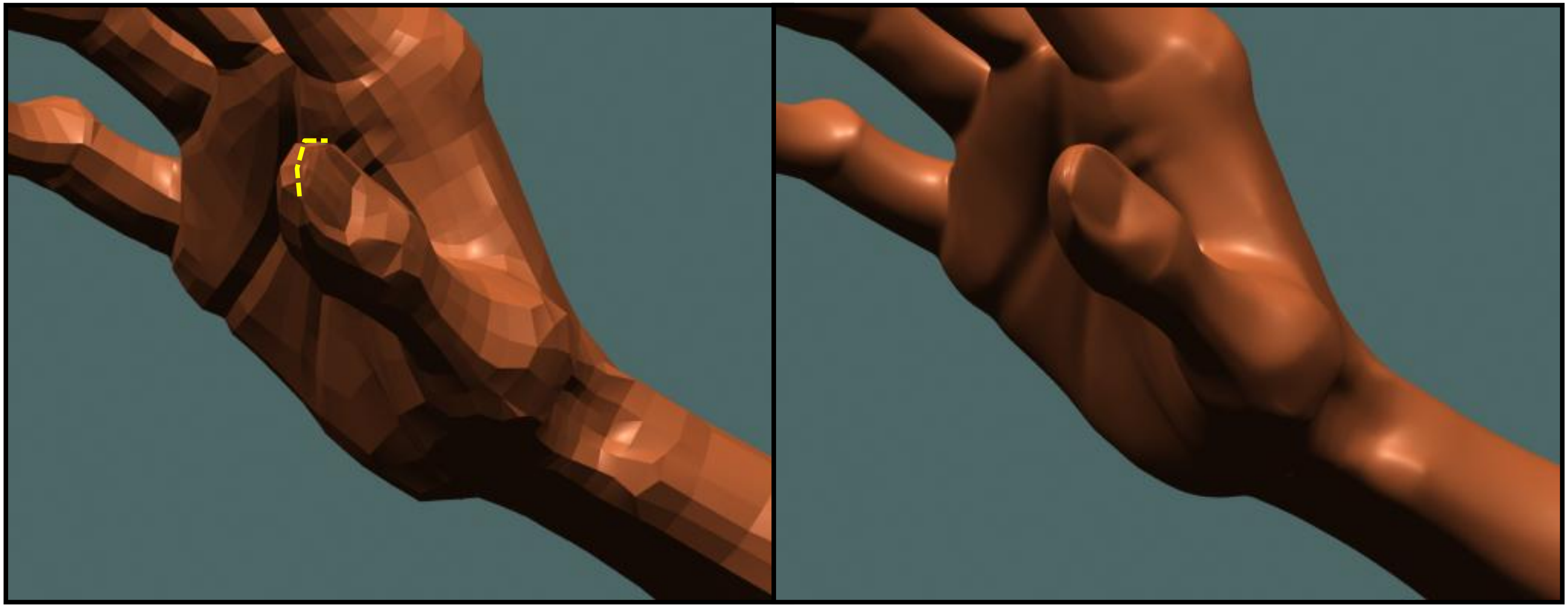
Loop Subdivision Scheme



Pixar

Smooth surfaces can be constructed from coarse meshes!

Loop Subdivision Scheme



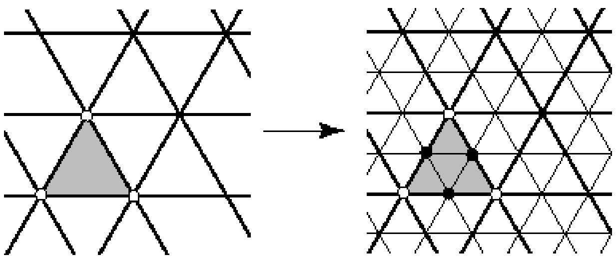
Pixar

Sharp creases can be specified by specifying that certain curves should subdivide as boundary curves.

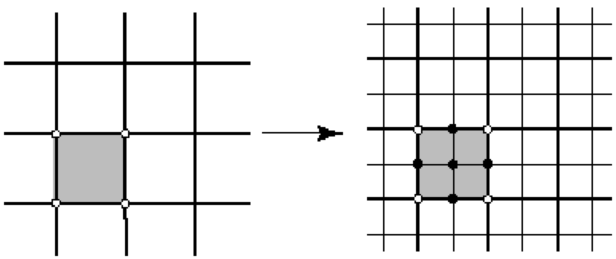
Subdivision Schemes

There are different subdivision schemes

- Different methods for refining topology
- Different rules for positioning vertices
  - Interpolating versus approximating



Face split for triangles



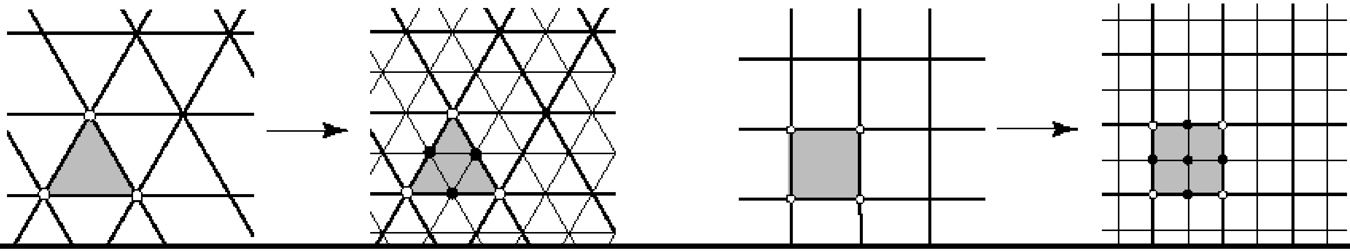
Face split for quads

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

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

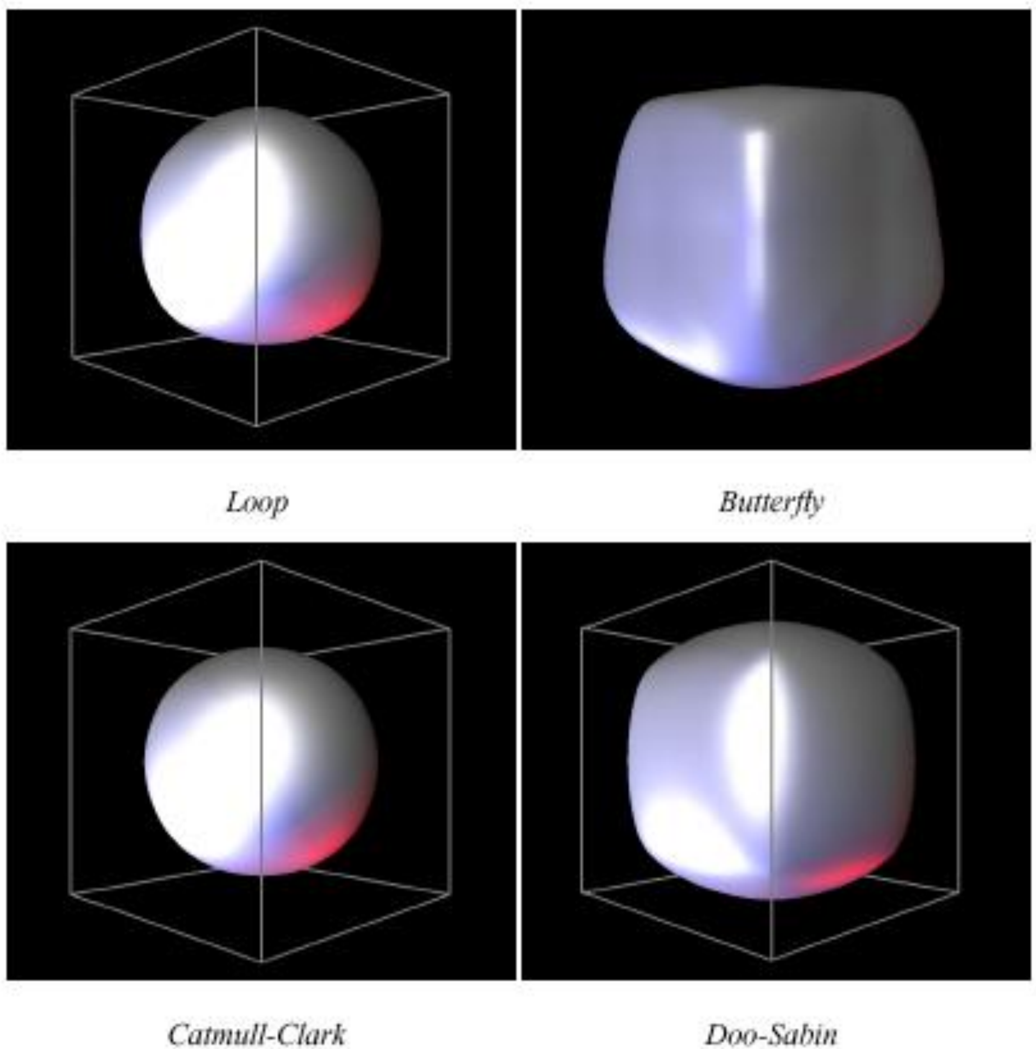


In general, forcing the subdivision to be interpolating removes degrees of freedom, making the solution less smooth.

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

Zorin & Schroeder
SIGGRAPH 99
Course Notes

Subdivision Schemes



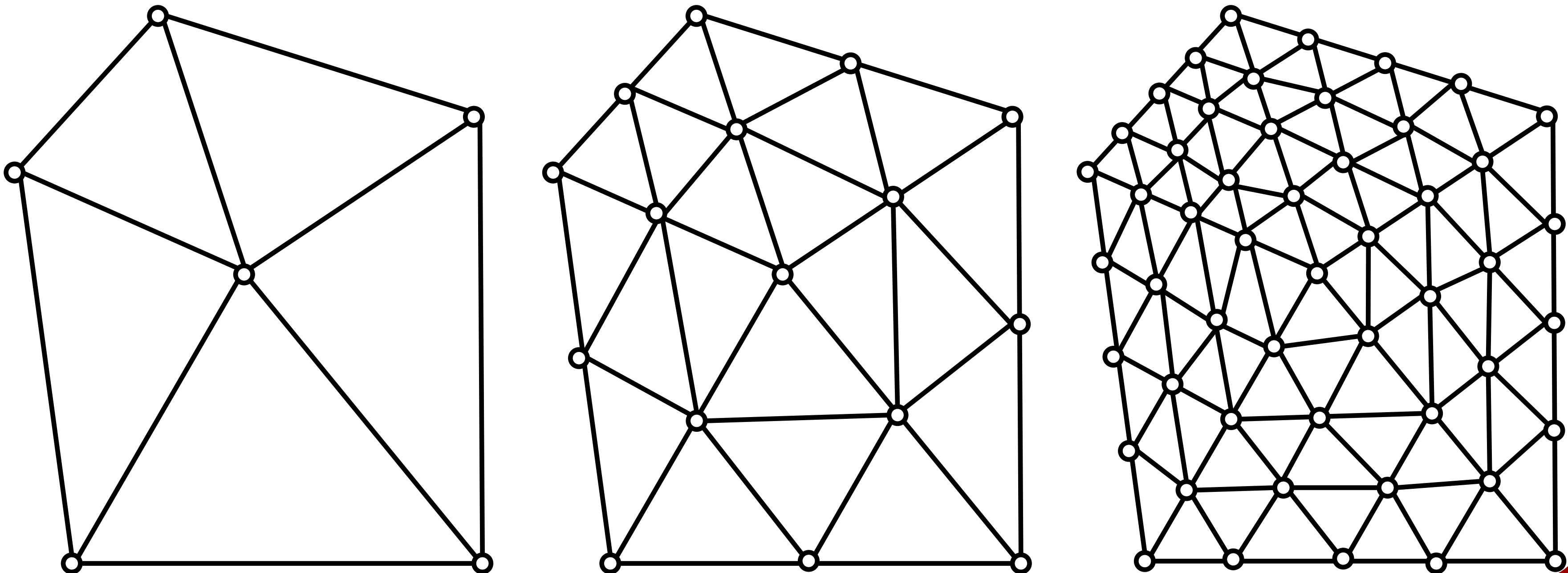
Zorin & Schroeder
SIGGRAPH 99
Course Notes

Subdivision Surfaces

- Properties:
  - ✓ Concise
  - ○ Local support
  - ○ Affine invariant
  - ○ Arbitrary topology
  - ○ Guaranteed smoothness
  - ○ Natural parameterization
  - ○ Efficient display
  - ○ Efficient intersections

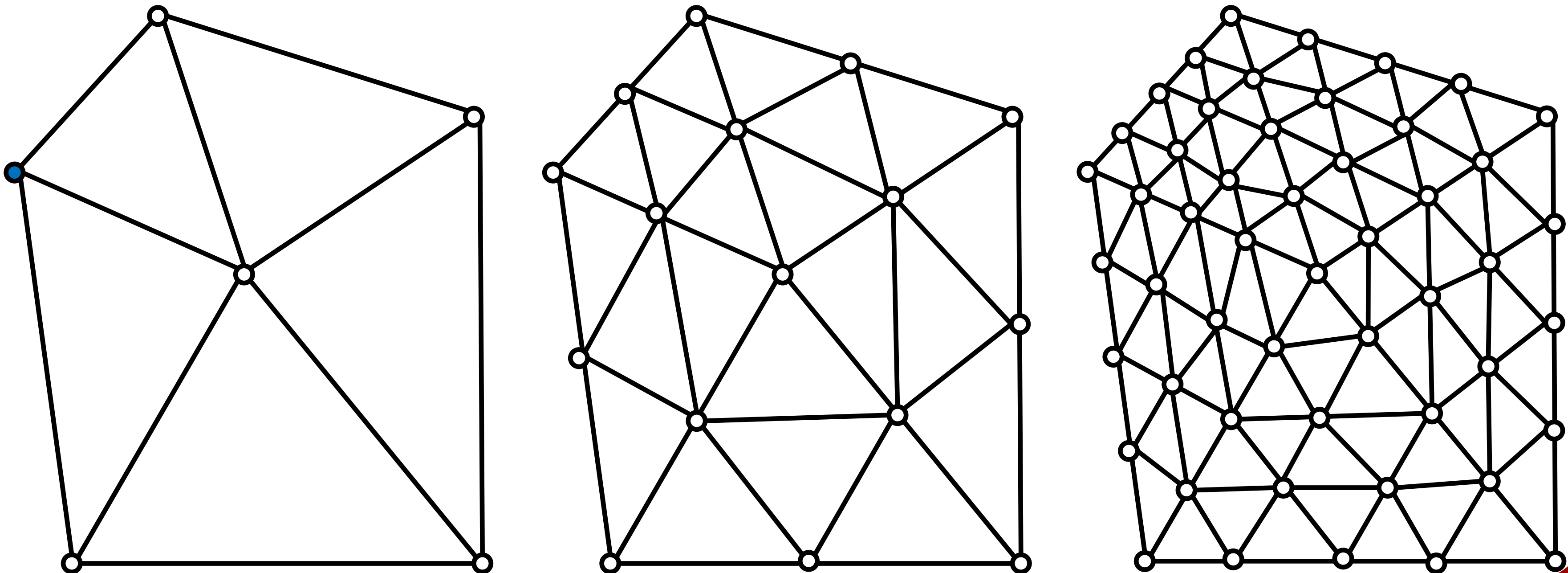
Pixar

Local support?



Local support?

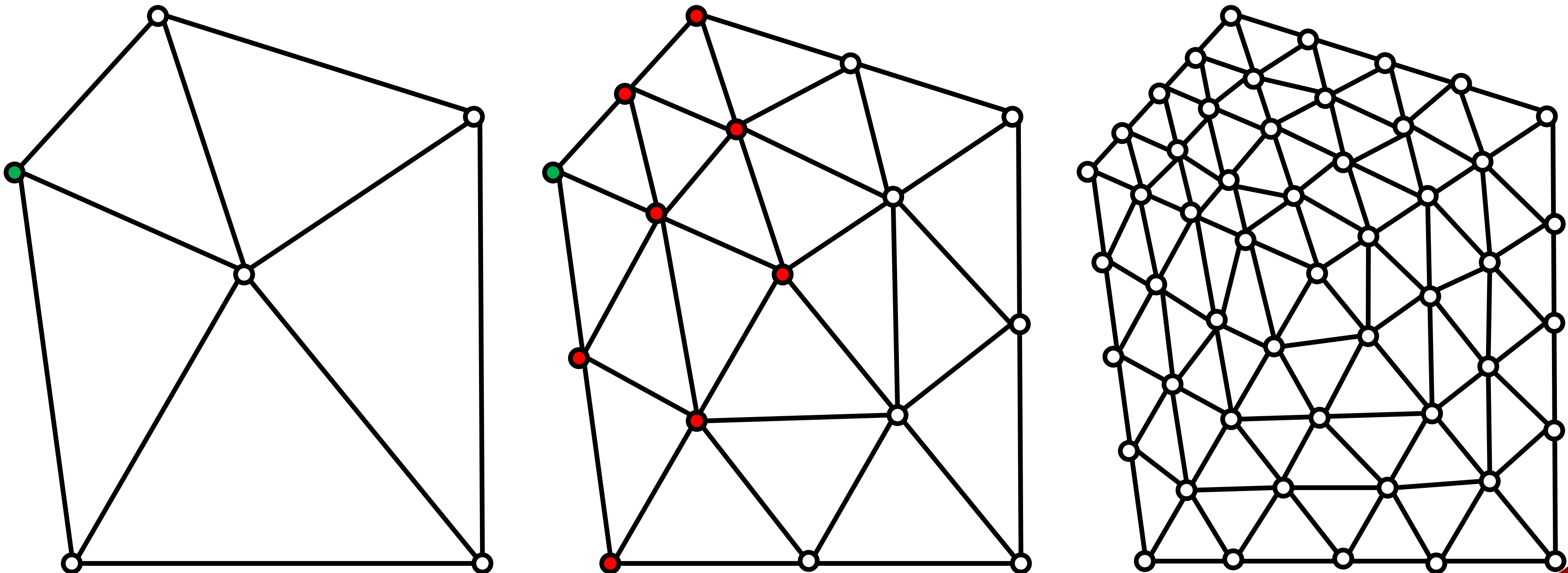
Modifying a vertex position at the coarser level



Local support?

Modifying a vertex position at the coarser level

We modify positions in the one-ring at the next level



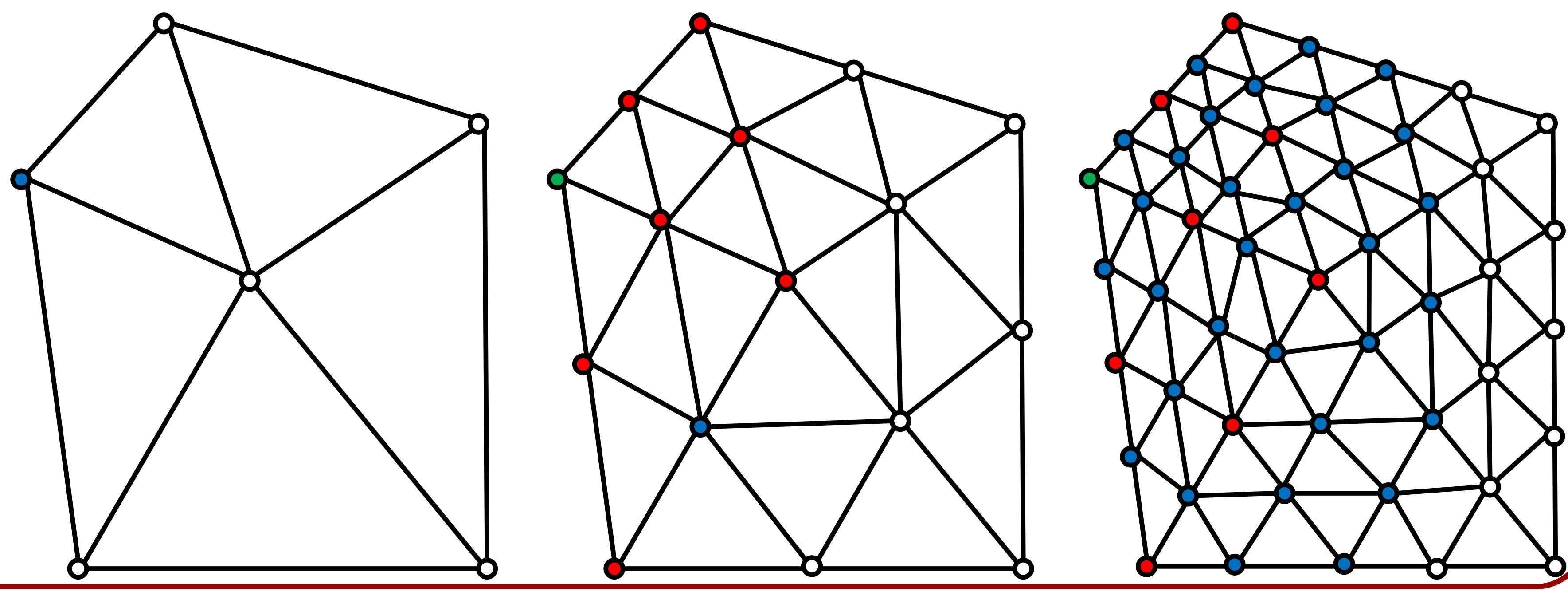
Local support?

Modifying a vertex position at the coarser level

We modify positions in the one-ring at the next level

Which modifies positions in the one-ring at the next level

Because we refine by a factor of two at each level, the effects are limited within the two-ring at the original level.



Subdivision Surfaces

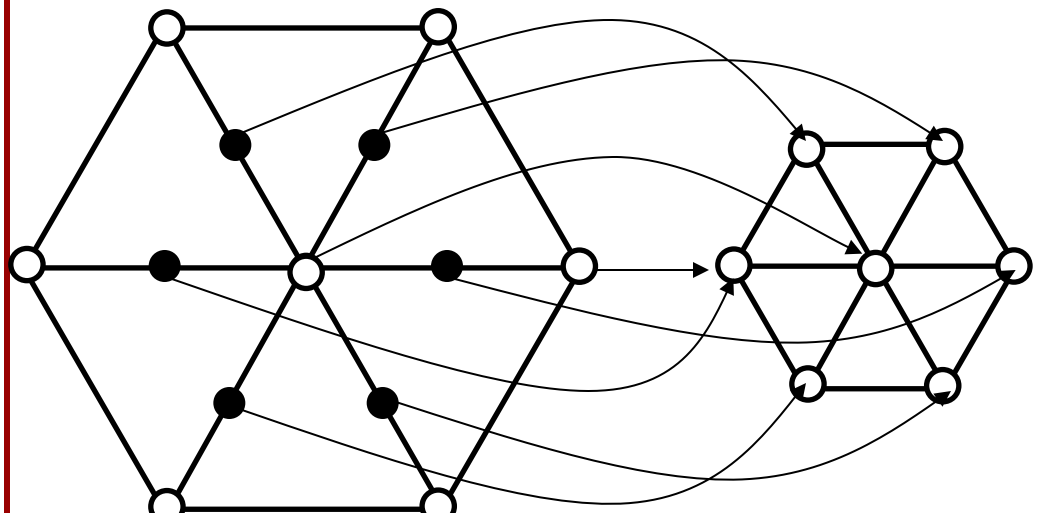
- Properties:
  - ✓ Concise
  - ✓ Local support
  - ✓ Affine invariant
  - ✓ Arbitrary topology
  - ◦ Guaranteed smoothness
  - ◦ Natural parameterization
  - ◦ Efficient display
  - ◦ Efficient intersections

Pixar

Guaranteed Smoothness?

To determine the smoothness of the subdivision:

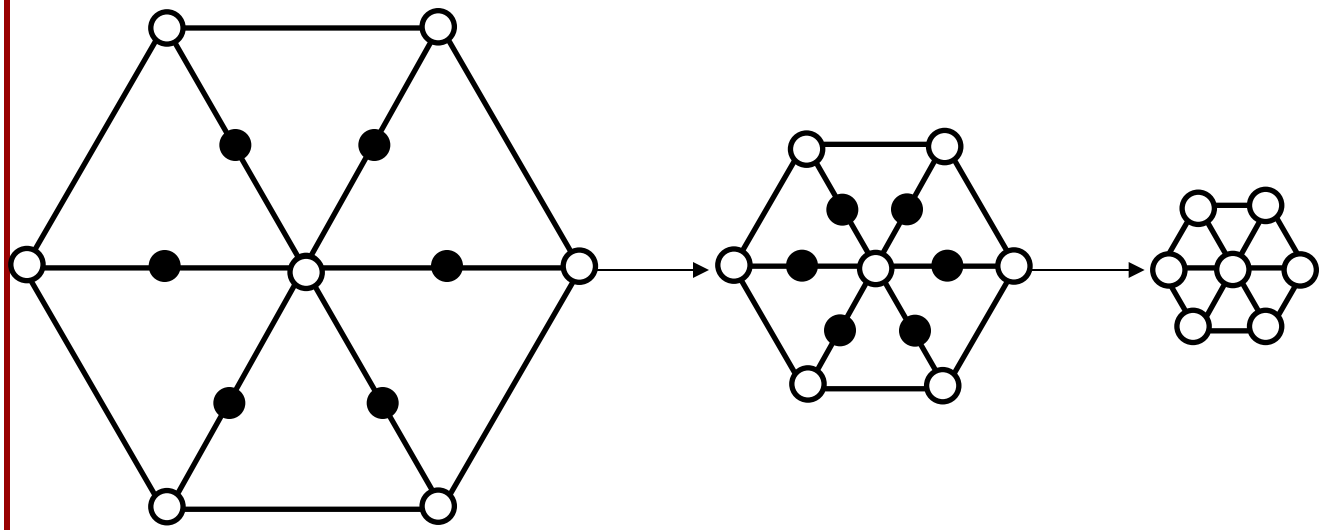
- Repeatedly apply the subdivision scheme
- Look at the neighborhood in the limit



Guaranteed Smoothness?

To determine the smoothness of the subdivision:

- Repeatedly apply the subdivision scheme
- Look at the neighborhood in the limit.

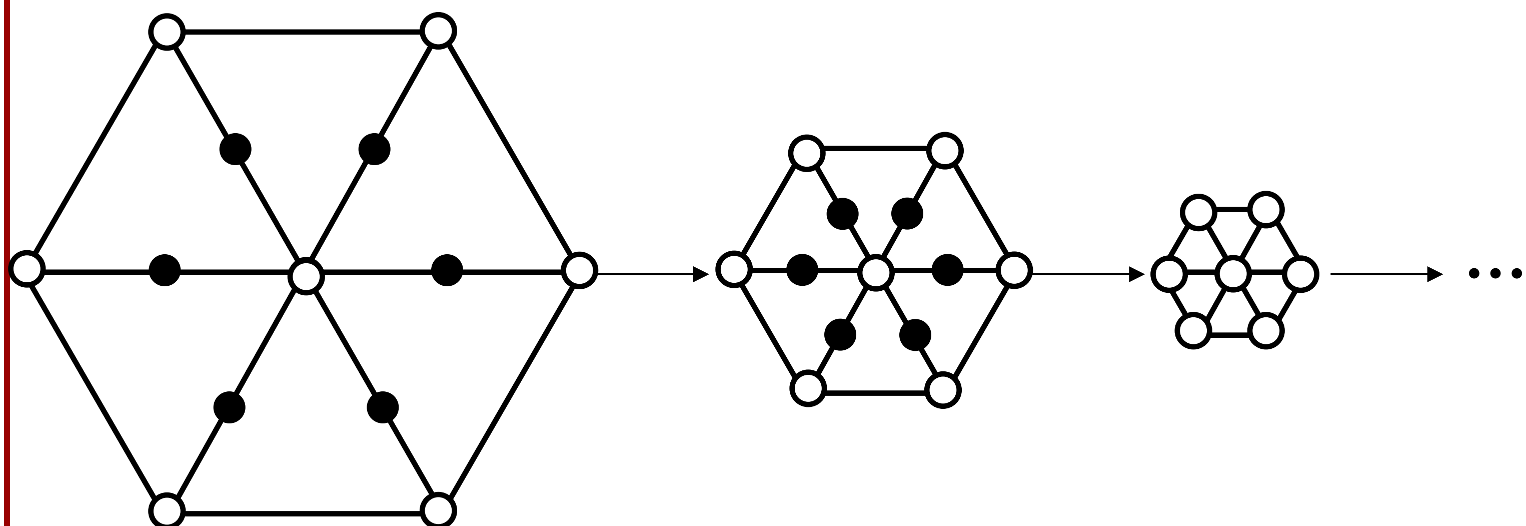


Guaranteed Smoothness?

To determine the smoothness of the subdivision:

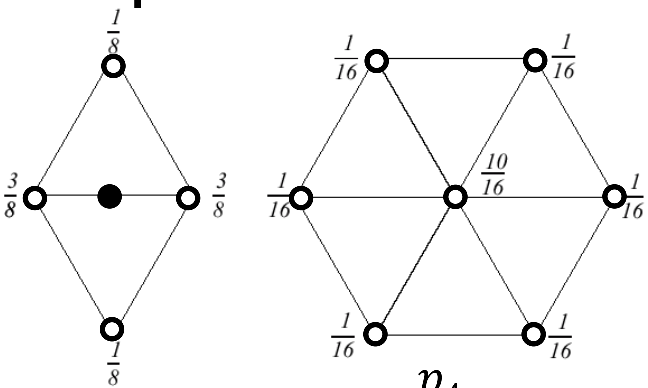
- Repeatedly apply the subdivision scheme
- Look at the neighborhood in the limit.

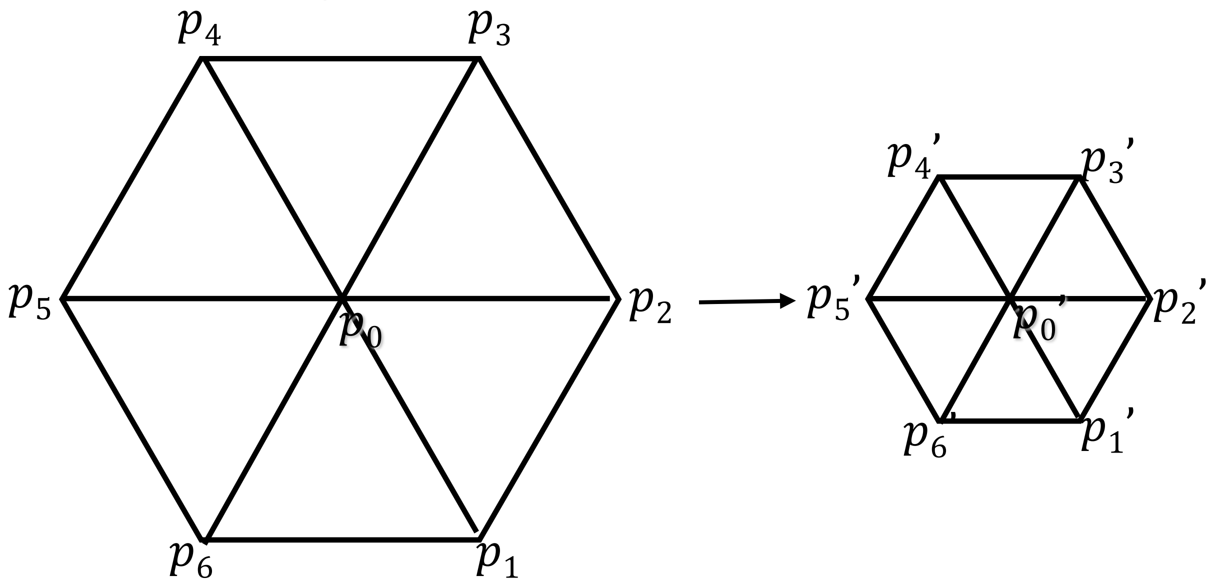
What happens after infinitely many iterations?



Guaranteed Smoothness?

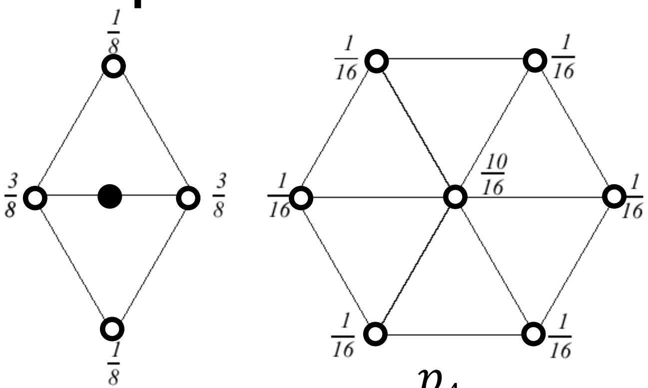
Compute the new positions as a linear combination of previous ones.





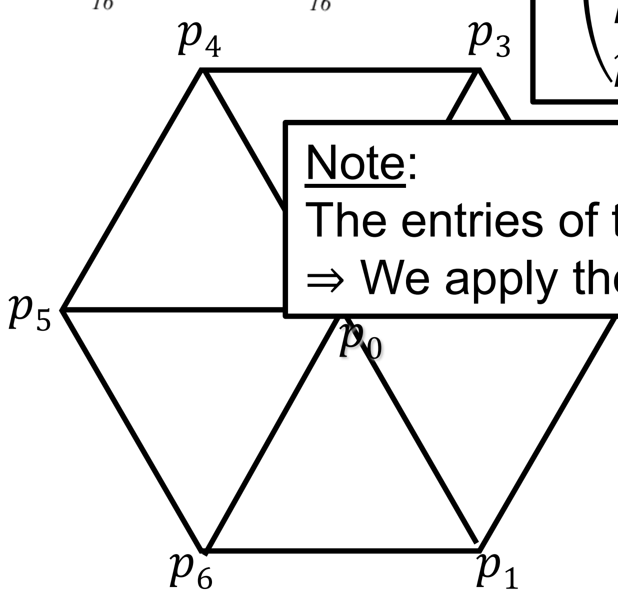
Guaranteed Smoothness?

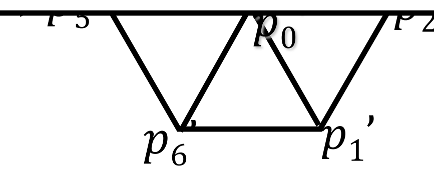
Compute the new positions as a linear combination of previous ones.



Subdivision Matrix

$$
\begin{pmatrix} p_0' \\ p_1' \\ p_2' \\ p_3' \\ p_4' \\ p_5' \\ p_6' \end{pmatrix}
= \frac{1}{16}
\begin{pmatrix}
10 & 1 & 1 & 1 & 1 & 1 & 1 \\
6 & 6 & 2 & 0 & 0 & 0 & 2 \\
6 & 2 & 6 & 2 & 0 & 0 & 0 \\
6 & 0 & 2 & 6 & 2 & 0 & 0 \\
6 & 0 & 0 & 2 & 6 & 2 & 0 \\
6 & 0 & 0 & 0 & 2 & 6 & 2 \\
6 & 2 & 0 & 0 & 0 & 2 & 6
\end{pmatrix}
\begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{pmatrix}
$$





<u>Note</u>:
The entries of the left and right vectors are 3D positions.
⇒ We apply the matrix to each coordinate independently

Guaranteed Smoothness?

Compute the new positions as a linear combination of previous ones.

To find the limit position of p_0, repeatedly apply the **subdivision matrix.**

Use eigenvalue decomposition to compute the n^{th} power of the matrix efficiently.

$$\begin{pmatrix} p_0^{(n)} \\ p_1^{(n)} \\ p_2^{(n)} \\ p_3^{(n)} \\ p_4^{(n)} \\ p_5^{(n)} \\ p_6^{(n)} \end{pmatrix} = \Bigg[\underbrace{\frac{1}{16}\begin{pmatrix} 10 & 1 & 1 & 1 & 1 & 1 & 1 \\ 6 & 6 & 2 & 0 & 0 & 0 & 2 \\ 6 & 2 & 6 & 2 & 0 & 0 & 0 \\ 6 & 0 & 2 & 6 & 2 & 0 & 0 \\ 6 & 0 & 0 & 2 & 6 & 2 & 0 \\ 6 & 0 & 0 & 0 & 2 & 6 & 2 \\ 6 & 2 & 0 & 0 & 0 & 2 & 6 \end{pmatrix}}_{\mathbf{S}}\Bigg]^n \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{pmatrix}$$

Guaranteed Smoothness?

If, after a change of basis we have $\mathbf{S} = \mathbf{A}^{-1}\mathbf{D}\mathbf{A}$, where $\mathbf{D}$ is a diagonal matrix, then:

$$\begin{aligned}\mathbf{S}^n &= (\mathbf{A}^{-1}\mathbf{D}\mathbf{A})(\mathbf{A}^{-1}\mathbf{D}\mathbf{A})\cdots(\mathbf{A}^{-1}\mathbf{D}\mathbf{A})(\mathbf{A}^{-1}\mathbf{D}\mathbf{A}) \\ &= \mathbf{A}^{-1}\mathbf{D}^n\mathbf{A}\end{aligned}$$

Since $\mathbf{D}$ is diagonal, raising $\mathbf{D}$ to the n^{th} power is the same as raising each of the diagonal entries of $\mathbf{D}$ to the n^{th} power.

decomp

compute

power o

$$\mathbf{D}^n = \begin{pmatrix} \lambda_0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_6 \end{pmatrix}^n = \begin{pmatrix} \lambda_0^n & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_6^n \end{pmatrix}$$

$$\begin{matrix} 1 & 1 \\ 0 & 2 \\ 0 & 0 \\ 0 & 0 \end{matrix}\Bigg)\Bigg]^n \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{pmatrix}$$

- If $|\lambda_i| > 1$ <u>for any</u> $0 \le i \le 6$, $\mathbf{D}^n$ blows up as $n \to \infty$.
- If $|\lambda_i| < 1$ <u>for all</u> $0 \le i \le 6$, $\mathbf{D}^n$ collapses as $n \to \infty$.
- If $\lambda_i = -1$ for any $0 \le i \le 6$, $\mathbf{D}^n$ does not converge as $n \to \infty$.

Guaranteed Smoothness?

Set $\boldsymbol{S}^{\infty}$ to be the matrix:

$$\mathbf{S}^{\infty} = \mathbf{A}^{-1}\begin{pmatrix} \lambda_0^{\infty} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_6^{\infty} \end{pmatrix}\mathbf{A}$$

with $\lambda_i^{\infty} = 1$ if $\lambda_i = 1$, and $\lambda_i^{\infty} = 0$ otherwise.

The limit of the point p_0 and its 1-ring neighborhood under repeated subdivision is:

$$\begin{pmatrix} p_0^{\infty} \\ \hline \vdots \\ \hline p_6^{\infty} \end{pmatrix} = \boldsymbol{S}^{\infty}\begin{pmatrix} p_0 \\ \hline \vdots \\ \hline p_6 \end{pmatrix}$$

Note that if the subdivision scheme is continuous:
$p_0^{\infty} = p_1^{\infty} = p_2^{\infty} = p_3^{\infty} = p_4^{\infty} = p_5^{\infty} = p_6^{\infty}$

Guaranteed Smoothness?

Set $\boldsymbol{S}^{\infty}$ to be the matrix:

$$\mathbf{S}^{\infty} = \mathbf{A}^{-1} \begin{pmatrix} \lambda_0^{\infty} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_6^{\infty} \end{pmatrix} \mathbf{A}$$

with $\lambda_i^{\infty} = 1$ if $\lambda_i = 1$, and $\lambda_i^{\infty} = 0$ otherwise.

The limit of the point p_0 and its 1-ring neighborhood under repeated subdivision is:

Using a similar approach we can derive an expression for the normal at the limit point.
⇒ For the normal to be well-defined, we get additional constraints on diagonal values.

Subdivision Surfaces

- Properties:
  - ✓ Concise
  - ✓ Local support
  - ✓ Affine invariant
  - ✓ Arbitrary topology
  - ✓ Guaranteed smoothness
  - ✓ Natural parameterization
  - ○ Efficient display
  - ○ Efficient intersections

Given texture coordinates at the vertices of the base mesh, the weights used to set the positions at the subdivision level can also be used to set the texture coordinates.

Note:
Could be problematic if using a multi-chart atlas.

Pixar

Subdivision Surfaces

- Properties:
  - ✓ Concise
  - ✓ Local support
  - ✓ Affine invariant
  - ✓ Arbitrary topology
  - ✓ Guaranteed smoothness
  - ✓ Natural parameterization
  - ✓ Efficient display
  - ◦ Efficient intersections

Can refine so that triangle projections are pixel-sized. (Can even use the limit positions as the vertex coordinates.)

Pixar

Subdivision Surfaces

- Properties:
  - ✓ Concise
  - ✓ Local support
  - ✓ Affine invariant
  - ✓ Arbitrary topology
  - ✓ Guaranteed smoothness
  - ✓ Natural parameterization
  - ✓ Efficient display
  - ✗ Efficient intersections

Given a ray, cannot tell where it would intersect the limit surface.
(But can ray-trace sufficiently refined mesh.)

Pixar