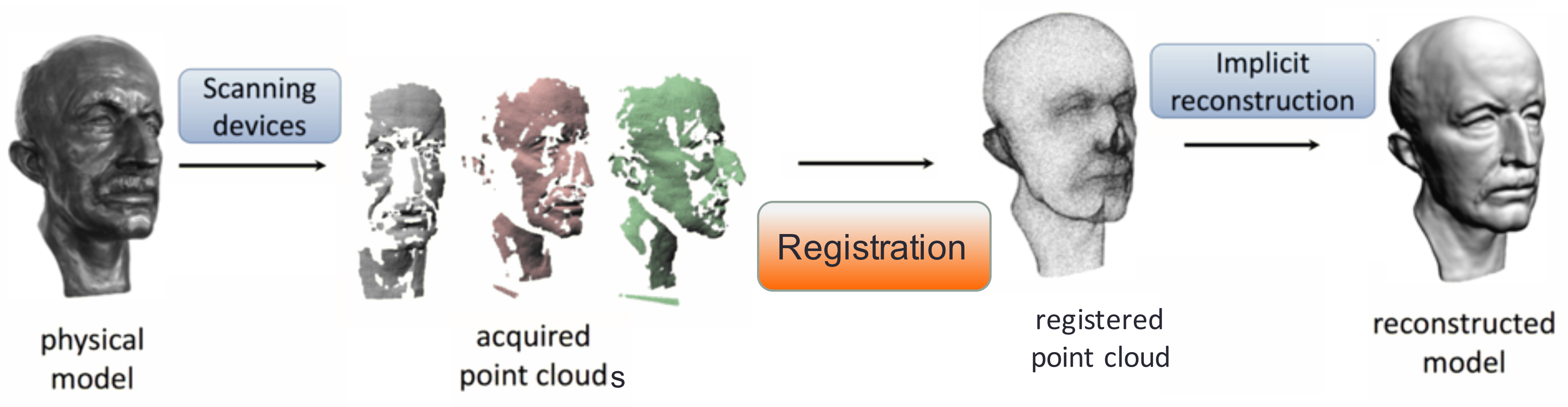
Digital Geometry Processing

Algorithms for Representing, Analyzing and Comparing 3D shapes

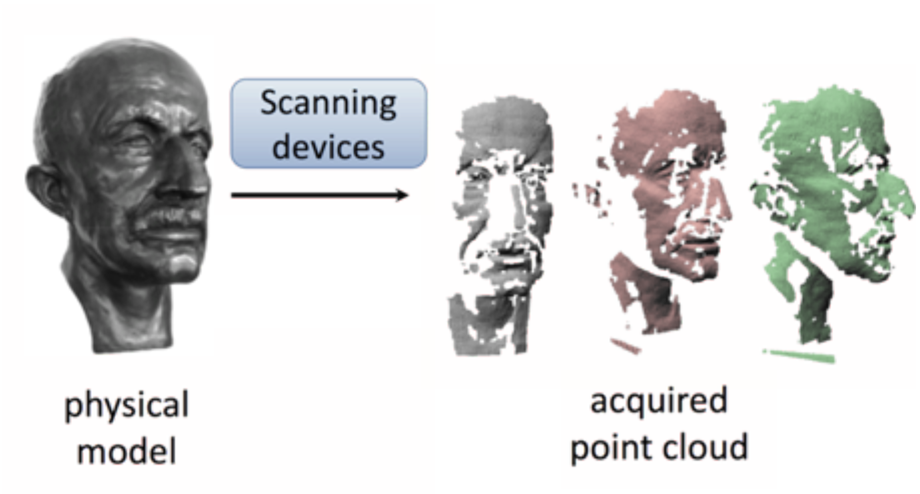
Today

- Previous lecture summary
- Triangle mesh basics
- Shape Simplification
- Shape Subdivision

Last Time


Scanning devices
physical model
acquired point clouds
Registration
registered point cloud
Implicit reconstruction
reconstructed model

Last Time



Types of 3D scanners:

- Time of Flight
  - Delay-based
  - Frequency-based
- Triangulation
  - Laser-based (single line)
  - Structured light (multiple patterns)
- Computer Vision-based
  - Depth-from-stereo
  - Depth-from-blur
- Example: Microsoft Kinect

Last Time

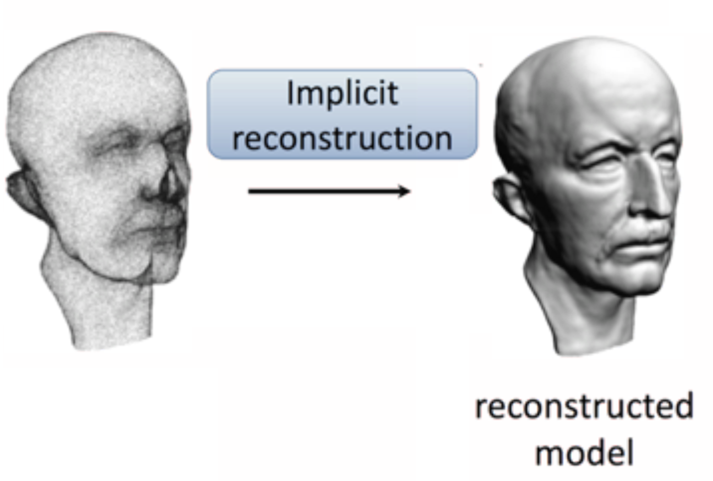
Partial Scans -> Single Point Cloud

Main Approach: Iterative Closest Point.

At each iteration:

1) Find nearest neighbor
2) Find best transformation
   a. Point-to-point (closed form solution)
   b. Point-to-plane (local linearization)

Last Time



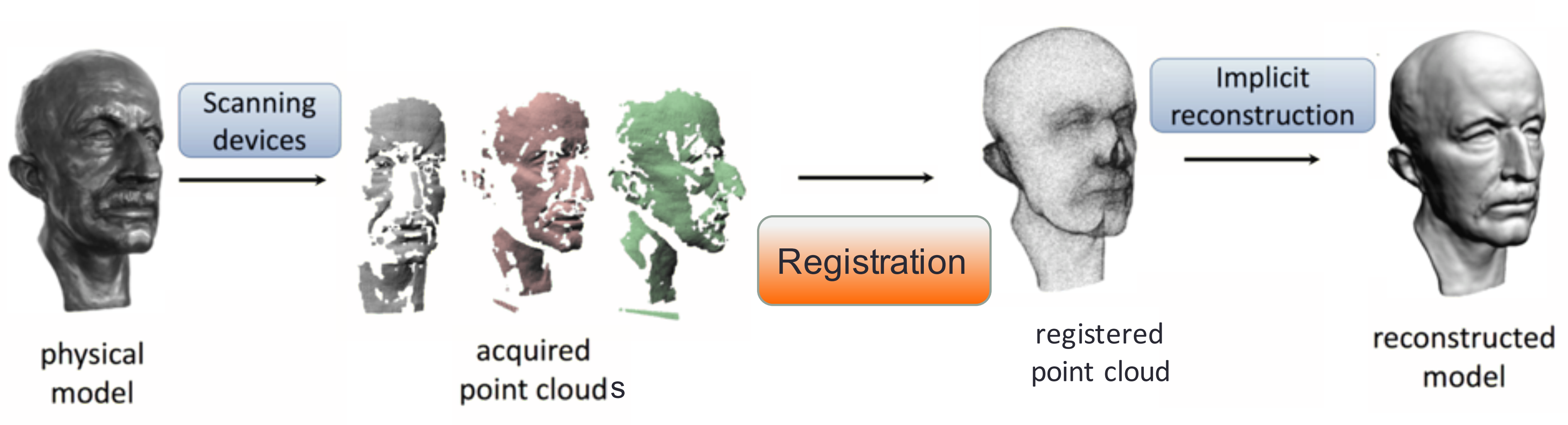
Point Cloud -> Triangle mesh

Two step process:

1) Given a point cloud, compute a signed distance function
   a. Simple projection
   b. Poisson-based

2) From the signed distance function, obtain a triangulation:
   a. Marching Cubes

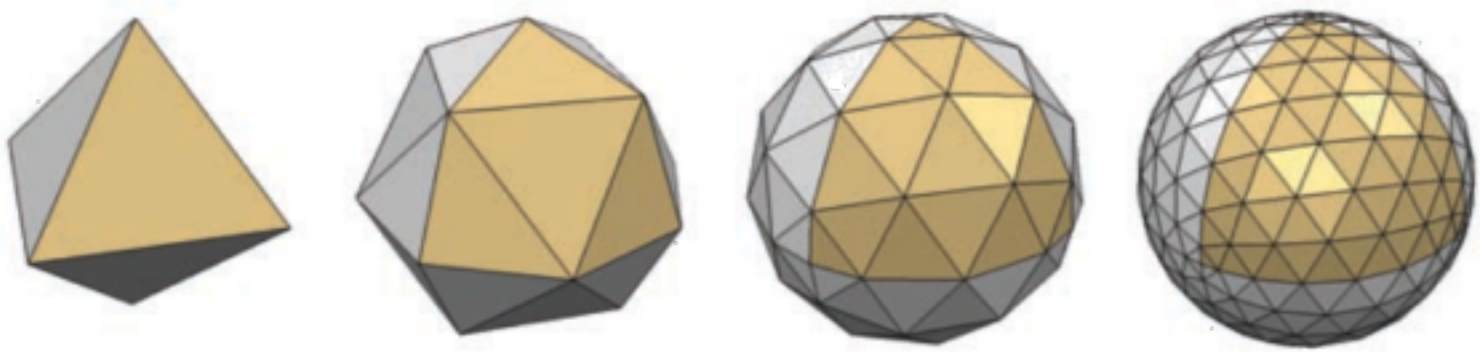
Last Time



Any Questions?

Today

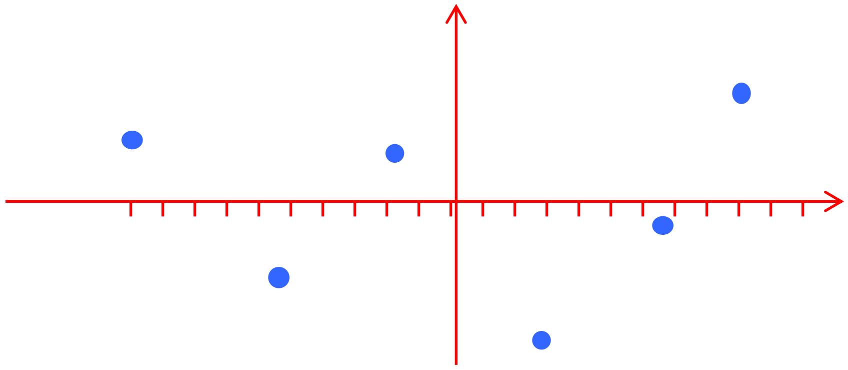
- Surface representation via triangle meshes
- Definitions and combinatorial properties
- Mesh simplification
- Mesh subdivision



Motivation: Curve Approximation

Setting (1D):

Given a set of pairs of points: $\{x_i, y_i\}$ approximate the function f s.t. $f(x_i) = y_i \; \forall \; i$

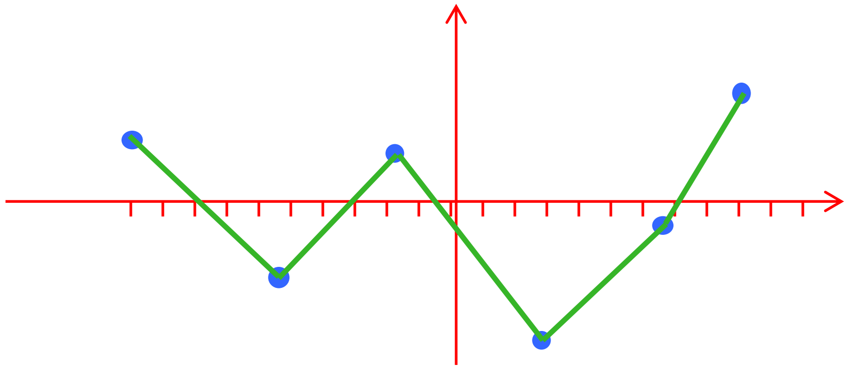


Point cloud

Motivation: Curve Approximation

Setting (1D):

Given a set of pairs of points: $\{x_i, y_i\}$ approximate the function f s.t. $f(x_i) = y_i \; \forall \; i$



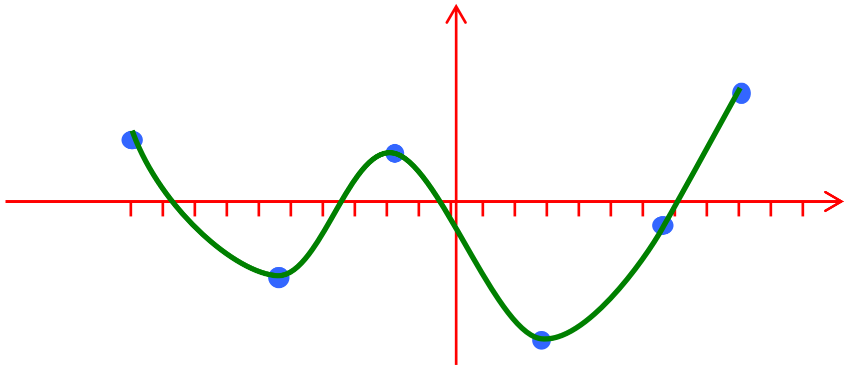
Simplest solution:

- Linear Interpolation:

Motivation: Curve Approximation

Setting (1D):

Given a set of pairs of points: $\{x_i, y_i\}$ approximate the function f s.t. $f(x_i) = y_i \ \forall \ i$



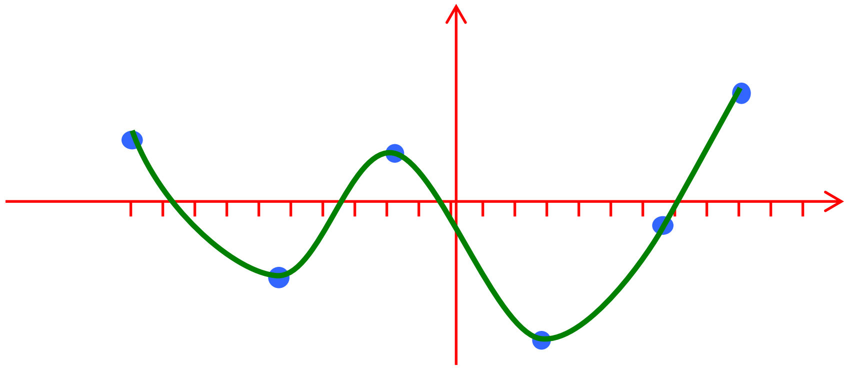
Smooth solution:

- Higher-order Interpolation: degree *p* polynomial

Motivation: Curve Approximation

Setting (1D):

Given a set of pairs of points: $\{x_i, y_i\}$ approximate the function f s.t. $f(x_i) = y_i \ \forall \ i$



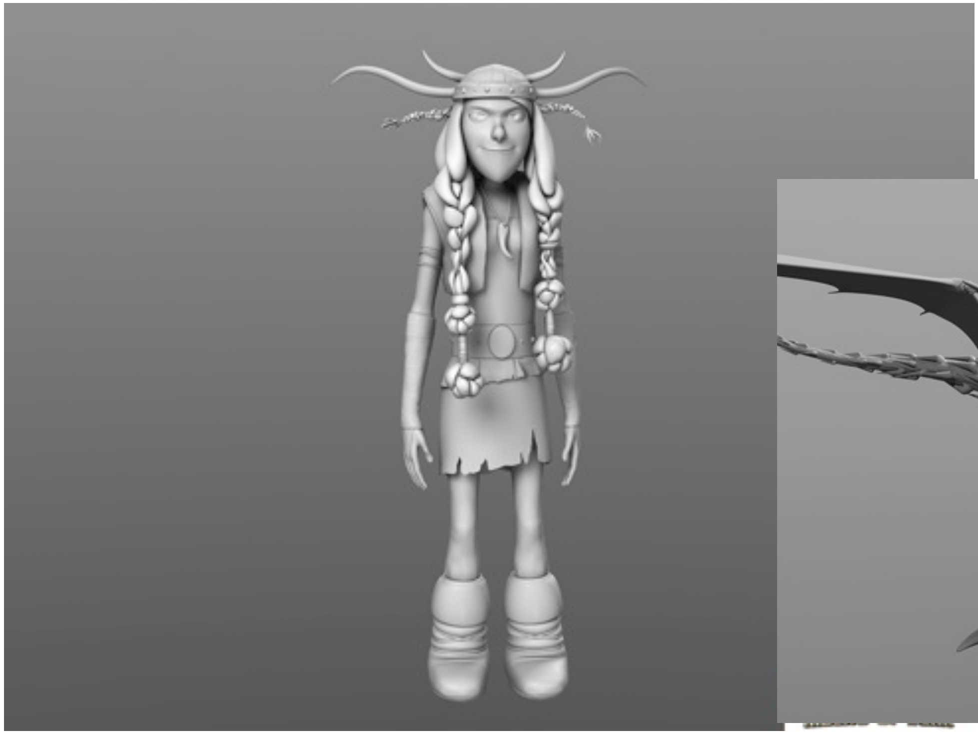
Generalized mean-value theorem: If *f* is a degree *p* polynomial, then the approximation error is:

$$|f(t) - g(t)| \leq \frac{1}{(p+1)!} \max f^{(p+1)} \prod_{i=0}^{p} (x_i - x) = O(h^{p+1})$$

Motivation: NURBS surfaces

Setting (surfaces in 3D):

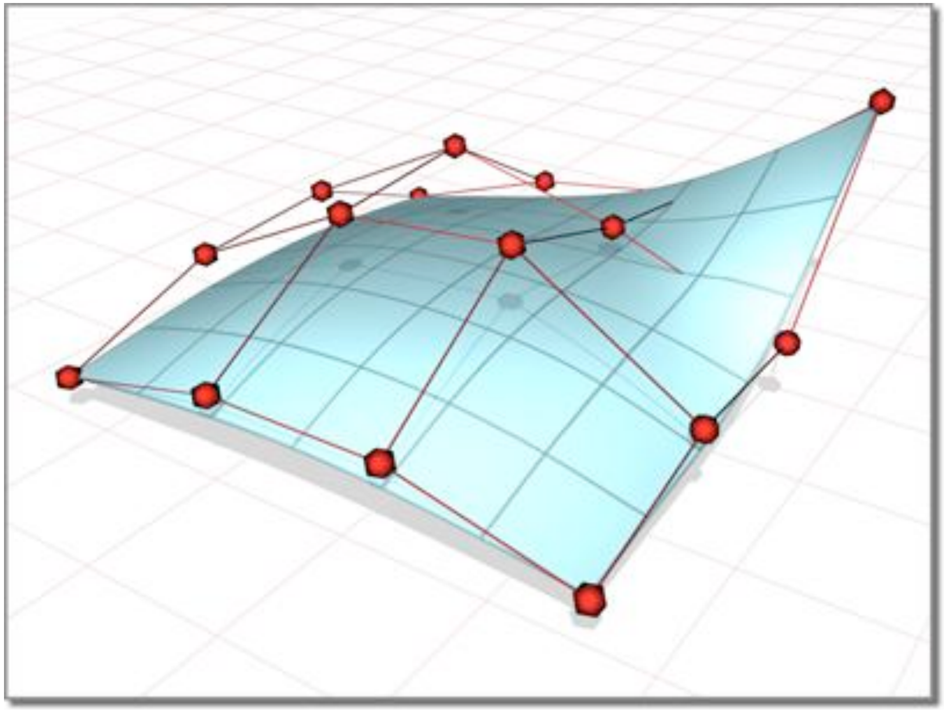
High-order approximations to surfaces (e.g. NURBS):



Motivation: NURBS surfaces

Setting (surfaces in 3D):

High-order approximations to surfaces (e.g. NURBS):



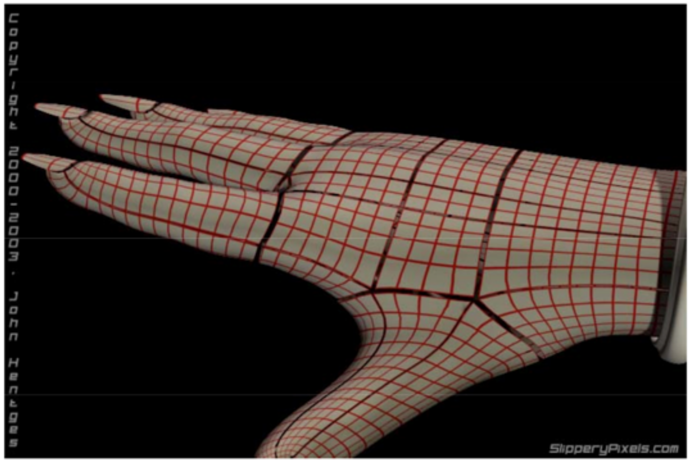
Defined via a *control lattice* with *control polygons*

Motivation: NURBS surfaces

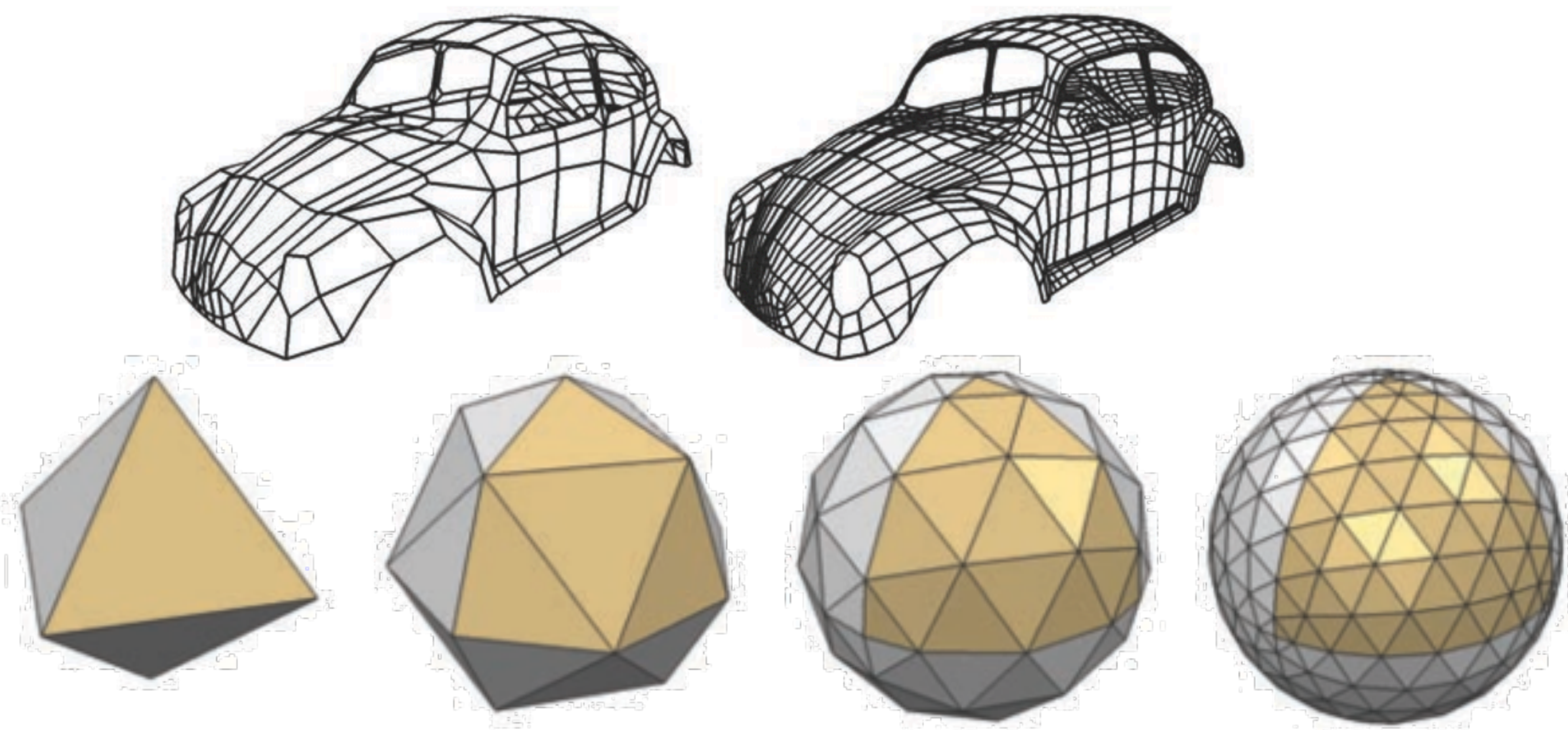
NURBS:

- Inherently **continuous**
- Intuitive controls (control mesh)
- Limited to grid domains
- A single NURBS patch has the topology of a sheet, cylinder or torus.

- Must use multiple patches to represent complex shapes
- Cracks occur after deformations.



[Triangle] Meshes

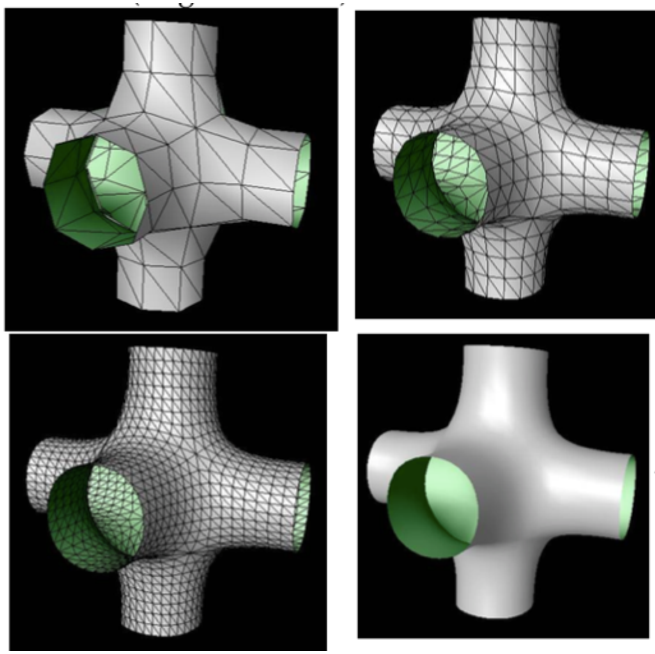


Surface represented simply as collections of:
Vertices, Edges, and **Faces**

NURBS vs. Triangle Meshes

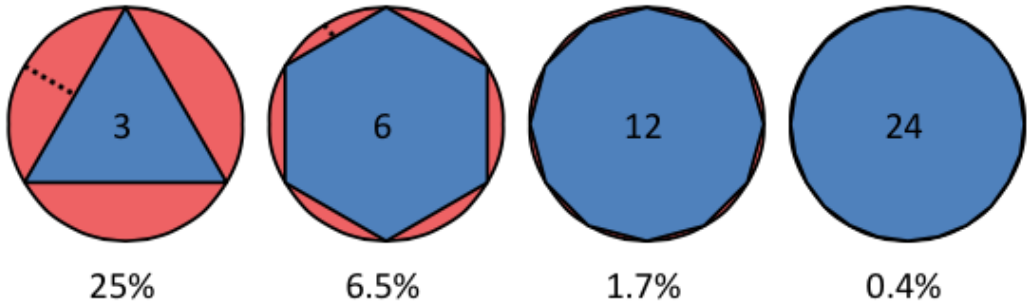
Triangle meshes

- Inherently **discrete**
- No need to have special rules for joining different patches.
- Can model shapes with arbitrary topology
- Can use adaptive sampling to add resolution where necessary
- Allow subdivision for smoothness (today)
- Easy to render

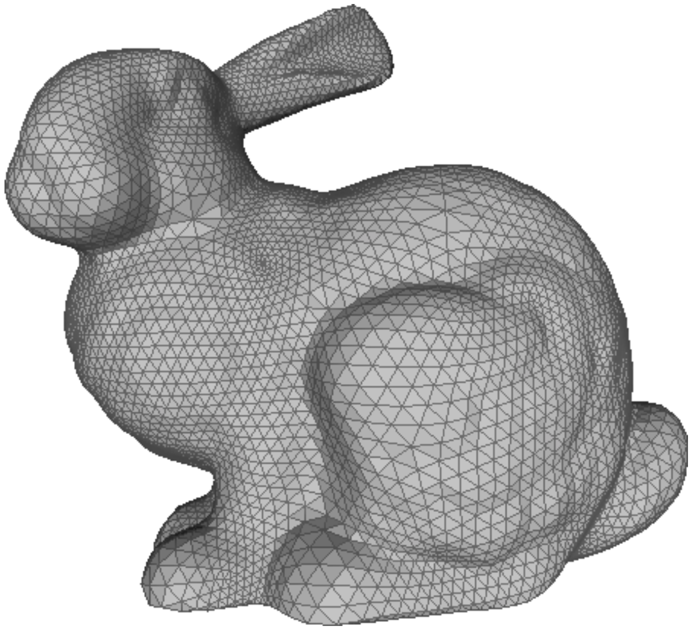


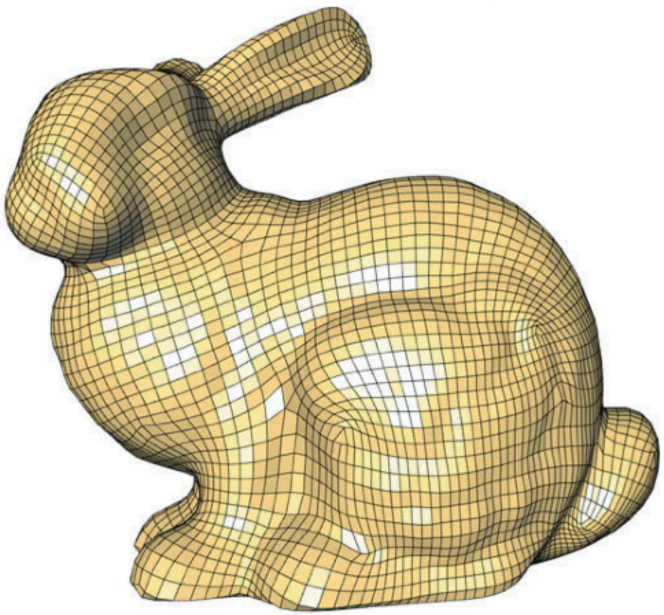
Why Triangle Meshes?

- Provide piece-wise linear approximation to the surface
  - Error is $O(h^2)$
- Doubling the number of vertices reduces the error by 4.



Why Triangles?

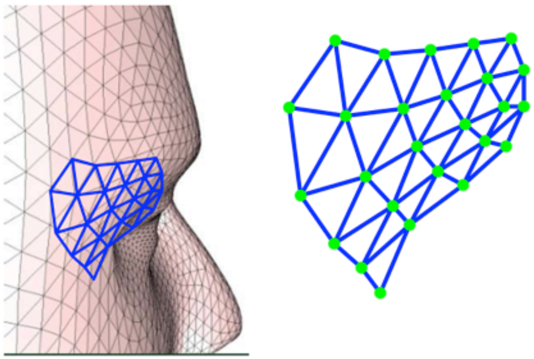




- Simplest piecewise linear element
- Easy to reconstruct from point clouds
- Easy to represent

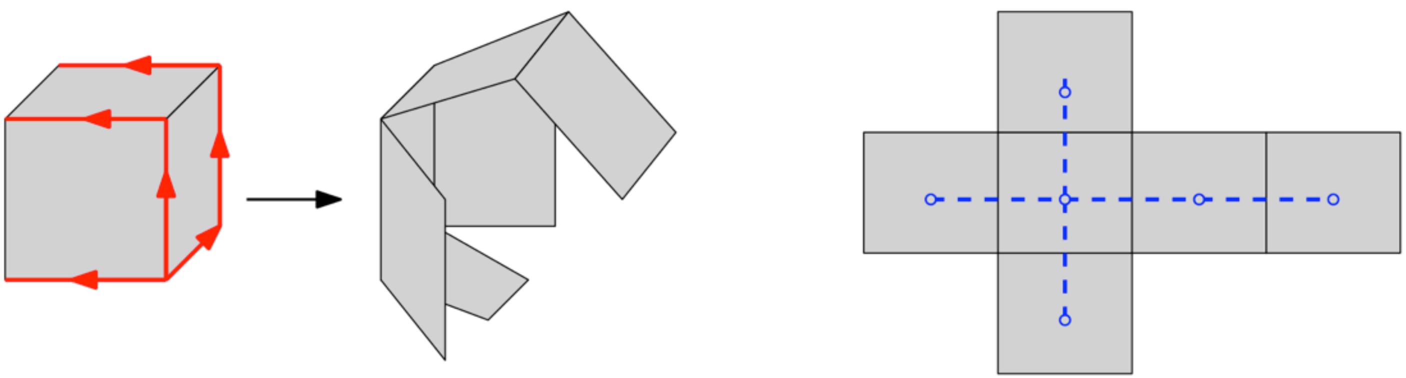
- Quad meshes often used in animation
- Typically require some hand-tuning in reconstruction
- Can provide more flexibility for deformation

[Triangle] Meshes



surface mesh: set of vertices, edges and faces (polygons) defining a polyhedral surface in embedded in 3D (discrete approximation of a shape)

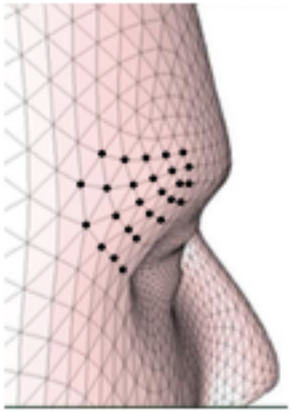
Combinatorial structure + geometric embedding



[Triangle] Meshes: 2 main parts

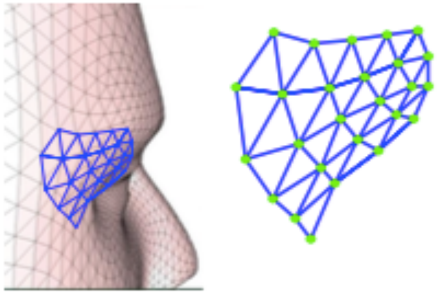
Geometric Structure vs. Combinatorial Structure

Geometry



vertex coordinates

"Connectivity": the underlying triangulation

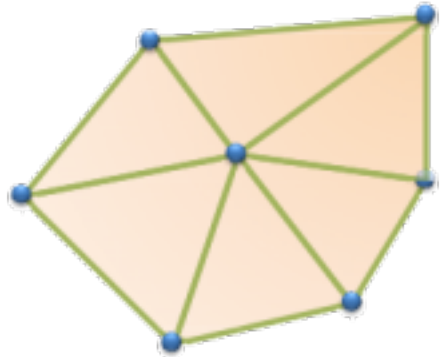


incidence relations between triangles, vertices and edges

[Triangle] Meshes: 2 main parts

- Geometry: vertex positions

$$\mathcal{P} = \{p_1, p_2, ..., p_n\}, \quad p_i \in \mathbb{R}^3$$



- Connectivity:
  - Vertices: $\mathcal{V} = \{v_1, v_2, ..., v_n\}$
  - Edges: $\mathcal{E} = \{e_1, e_2, ..., e_m\}, \quad e_i \in \mathcal{V} \times \mathcal{V}$
  - Faces: $\mathcal{F} = \{f_1, f_2, ..., f_k\}, \quad f_i \in \mathcal{V} \times \mathcal{V} \times \mathcal{V}$

What’s a **Valid** Triangle Mesh?

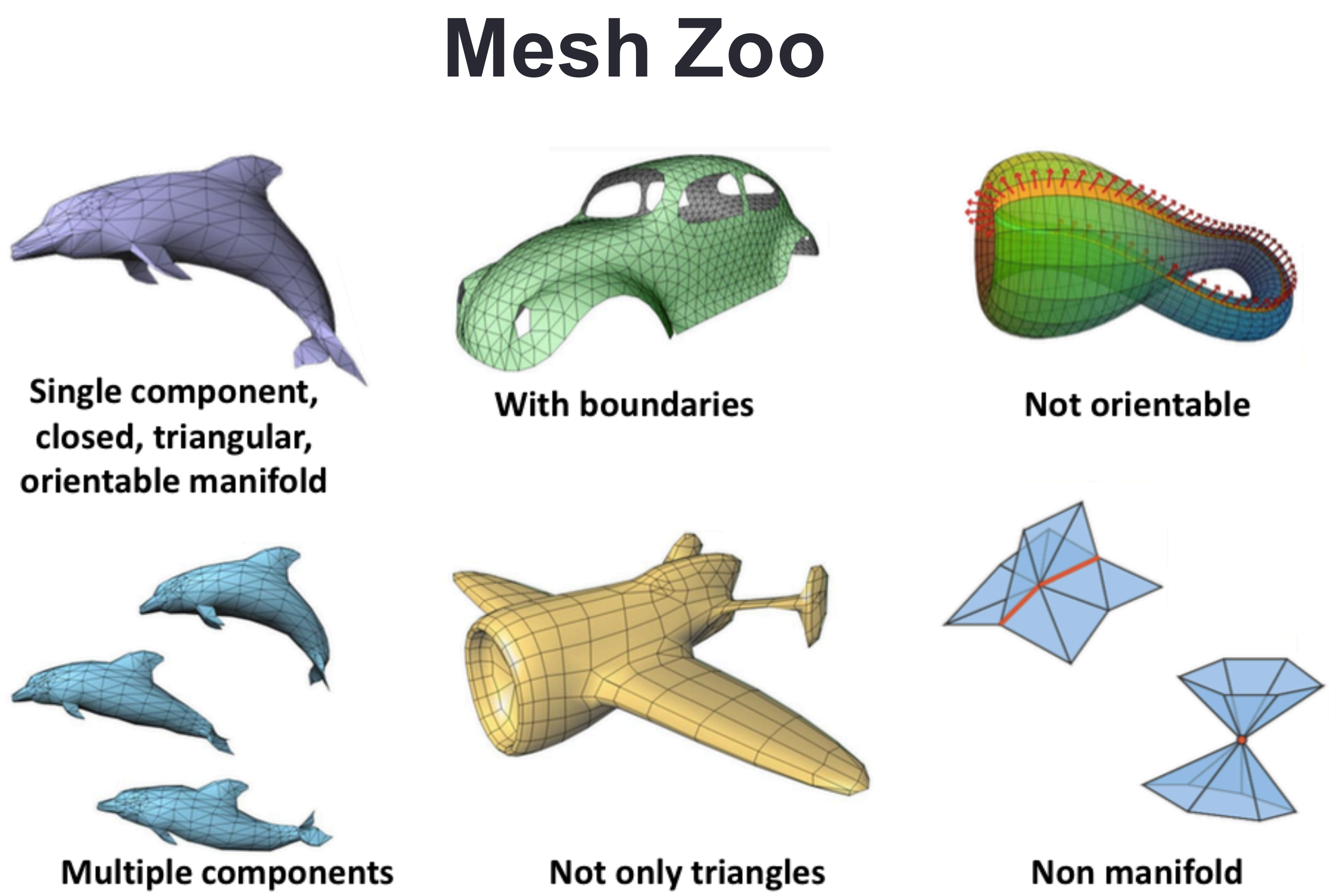


Image source: Mirela Ben-Chen

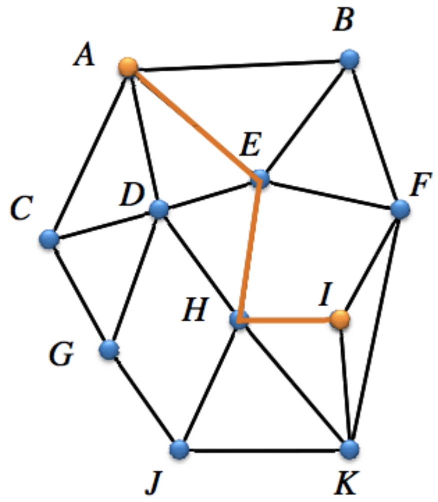
What's a **Valid** Triangle Mesh?

What is a valid connectivity?

Each face is a triangle

Single connected component

Manifold mesh



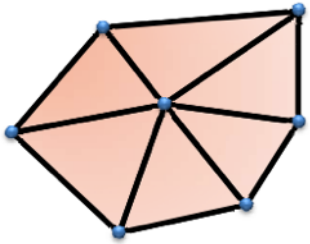
Connected =

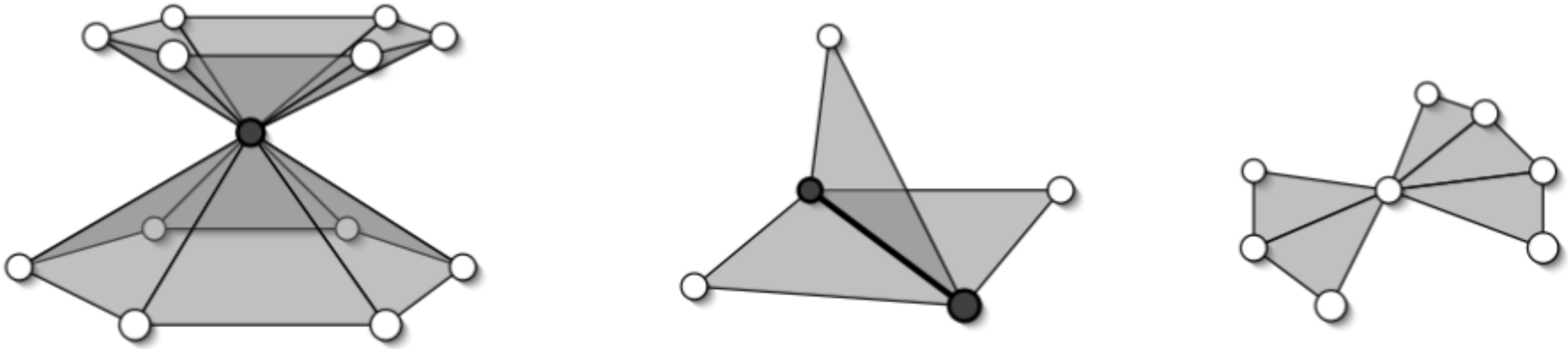
path of edges connecting every two vertices

What's a **Manifold Triangle** Mesh?

Manifold triangle mesh:

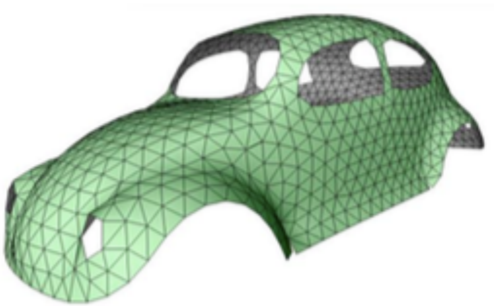
- Each Edge is adjacent to at most 2 faces:
- Each vertex has a disk-shaped neighborhood





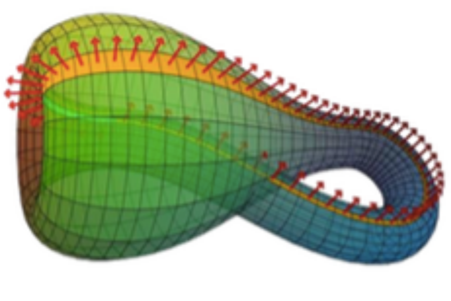
Non-manifold.

Some More Terminology



With boundaries

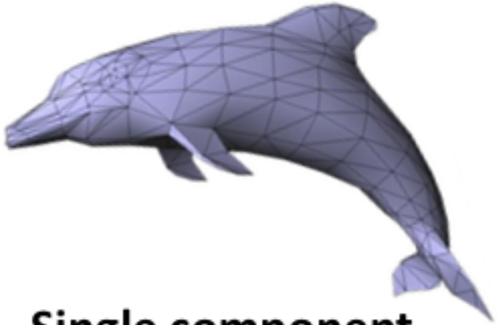
Boundary edge: adjacent to exactly one face



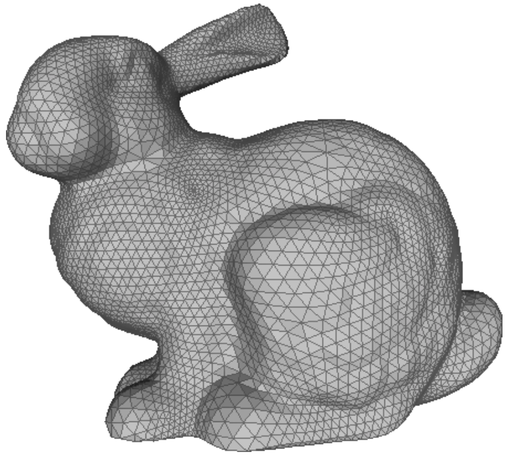
Not orientable

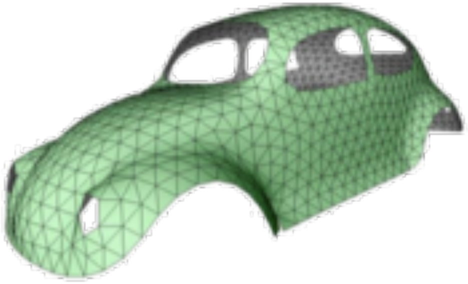
Orientable surface: possible to assign a consistent normal orientation (e.g. outward)

From now on, a triangle mesh:



Single component, closed, triangular, orientable manifold





Possibly with boundaries

Fundamental Combinatorial Relation

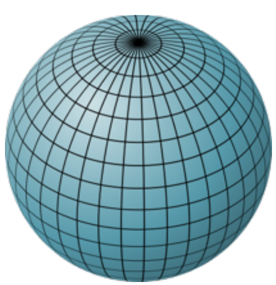
Euler-Poincaré identity for polyhedral surfaces

$$V - E + F = \chi = 2 - 2g - b$$

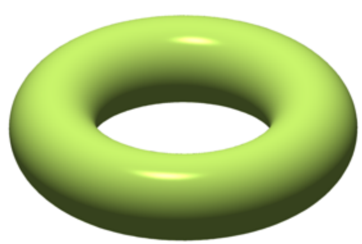
χ : Euler characteristic

g : genus (number of "handles")

b : number of boundary components

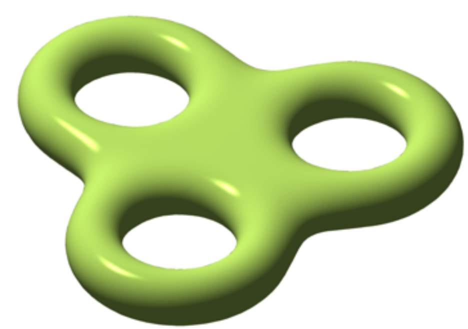


Genus 0



Genus 1

Genus 2

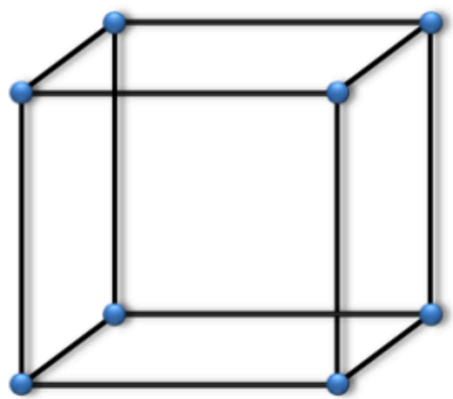


Genus 3

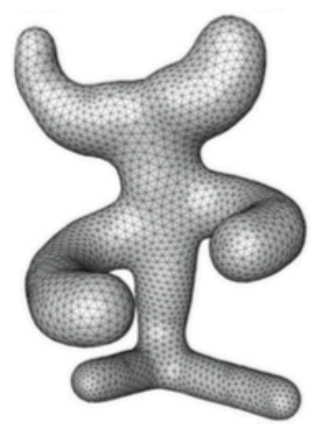
Fundamental Combinatorial Relation

Euler-Poincaré identity for polyhedral surfaces

$$V - E + F = \chi = 2 - 2g - b$$



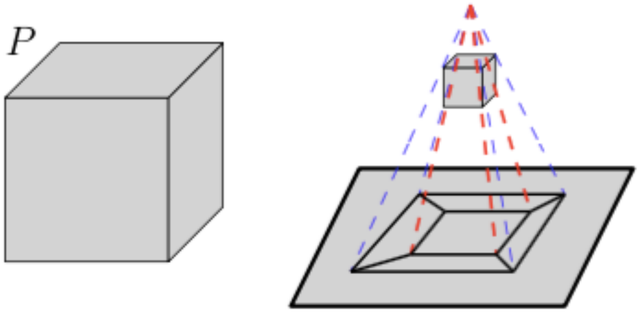
$V = 8$
$E = 12$
$F = 6$
$\chi = 8 + 6 - 12 = \mathbf{2}$



$V = 3890$
$E = 11664$
$F = 7776$
$\chi = \mathbf{2}$

Euler-Poincaré identity

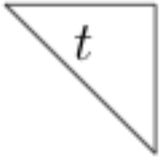
Proof in the case of planar graphs or convex surfaces:



Euler's relation for planar graphs:

$$V - E + F = 2$$

Base case:



$$\chi(t) = 3 - 3 + 2 = 2$$

(count the exterior face)

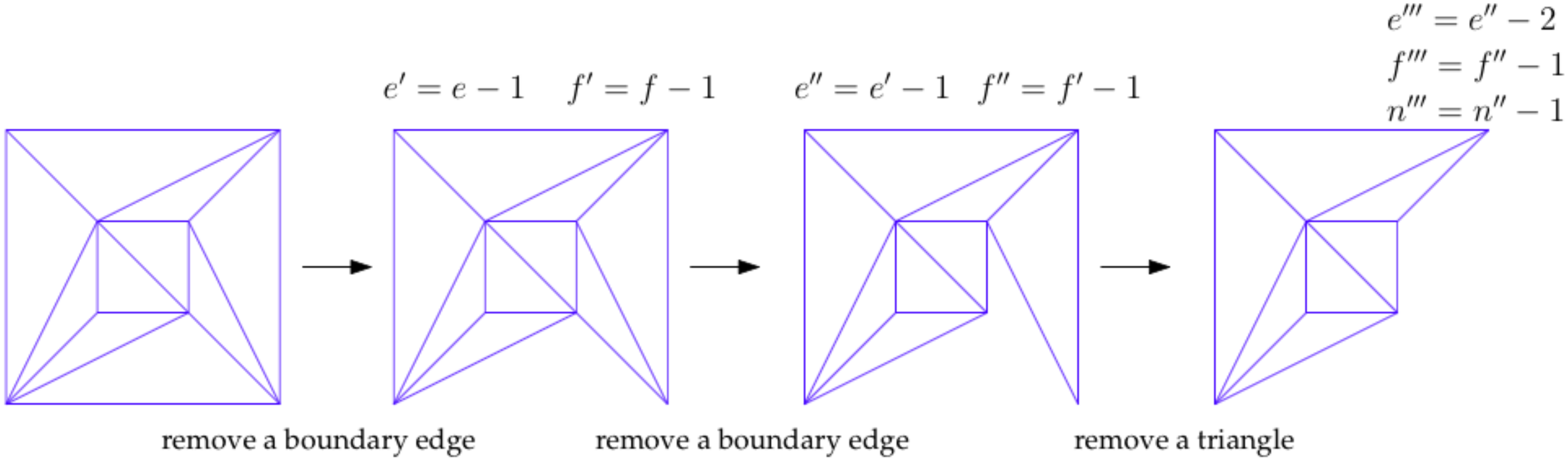
Euler-Poincaré identity

Proof in the case of planar graphs or convex surfaces:

$$V - E + F = 2$$

Proof by Induction:

invariant: the boundary (exterior) is a simple cycle
perform the removal according to a shelling order

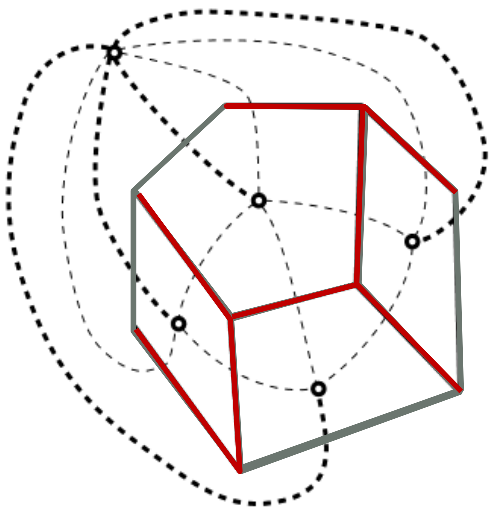


Euler-Poincaré identity

Proof in the case of planar graphs or convex surfaces:

$$V - E + F = 2$$

Von Staudt's proof:



Given a planar graph, construct any minimum spanning tree T.

The dual edges of its complement, *also form a spanning tree*.

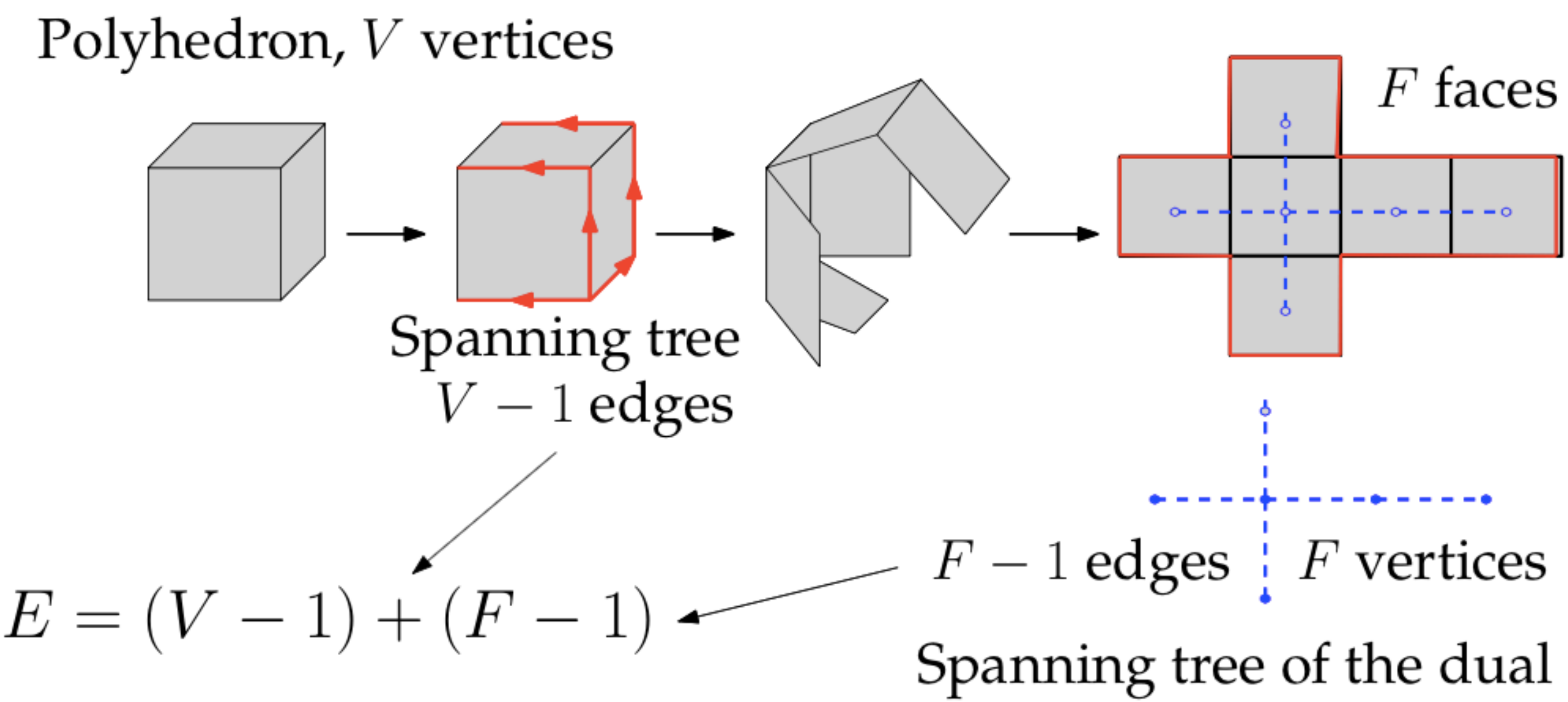
The two trees together have (V-1)+(F-1) edges.

$$E = (V - 1) + (F - 1) \quad \Rightarrow \quad V - E + F = 2$$

Euler-Poincaré identity

Proof in the case of planar graphs or convex surfaces:

$$V - E + F = 2$$



Some applications of Euler-Poincaré

For a manifold triangle mesh without boundary:

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

Since each triangle has 3 edges and each edge belongs to two triangles: $2E = 3F$

Combining with Euler:

$$2V - 3F + 2F = 2 - 2g \quad \Rightarrow$$

$$2V = F + (2 - 2g)$$

For small genus $F \approx 2V$, and $E \approx 3V$

Some applications of Euler-Poincaré

For a manifold triangle mesh without boundary:

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

For small genus $F \approx 2V$, and $E \approx 3V$

Since $$\sum_{i \in \mathcal{V}} \mathrm{degree}(v_i) = 2E$$

$$\mathrm{avg.\ \ degree} = \frac{2E}{V} \approx 6$$

Can distinguish torus, sphere and double torus by average degree.

Some applications of Euler-Poincaré

For a manifold triangle mesh without boundary:

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

Since each triangle has 3 edges and each edge belongs to two triangles: $2E = 3F$

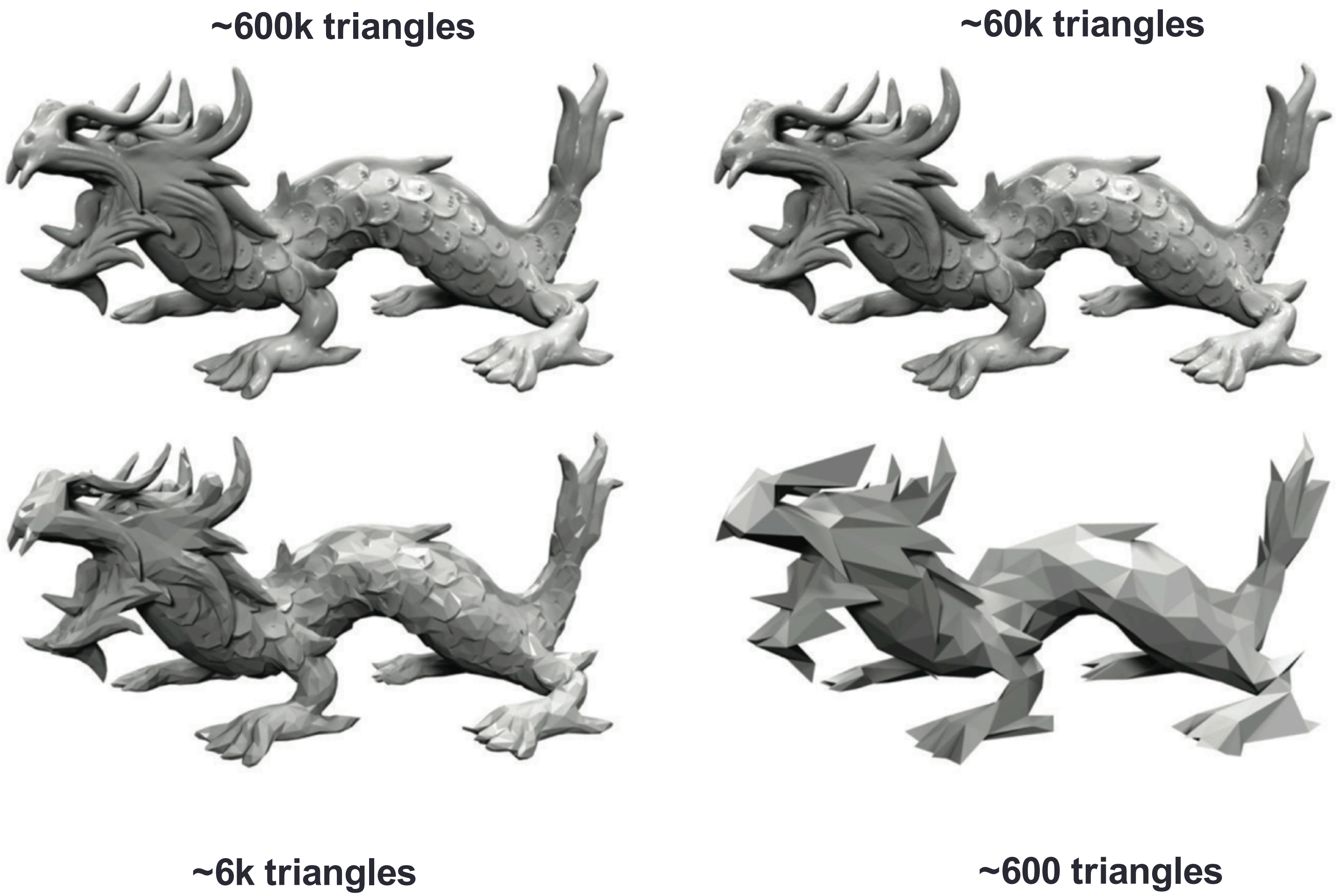
Combining with Euler:

Number of faces in terms of number of vertices

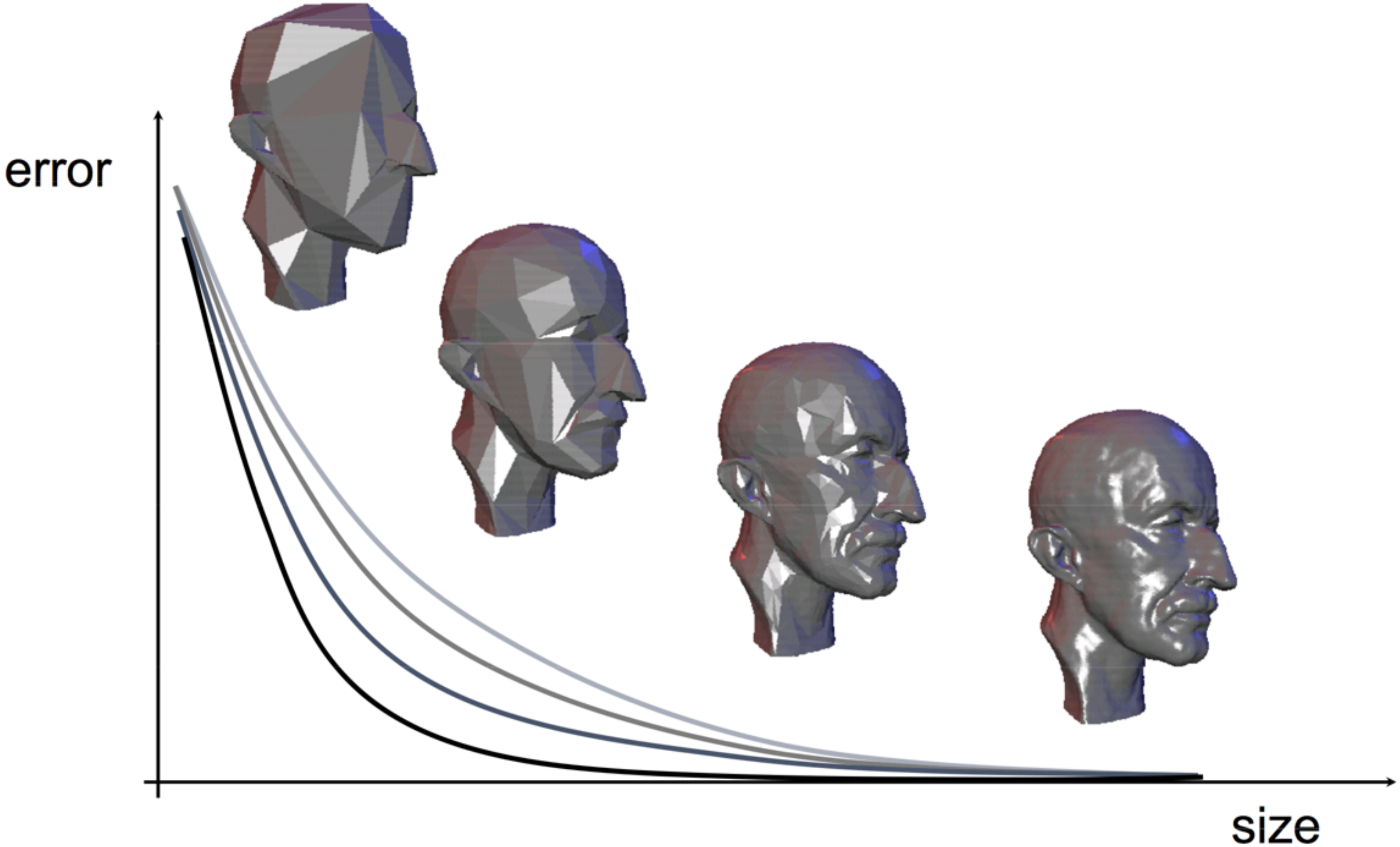
Average valence of the vertices

Triangulating a sphere with even-degree vertices

[Triangle] Meshes – simplification



[Triangle] Meshes – simplification



Incremental decimation – general framework

Evaluate local error

Select removable elements (vertices or edges)

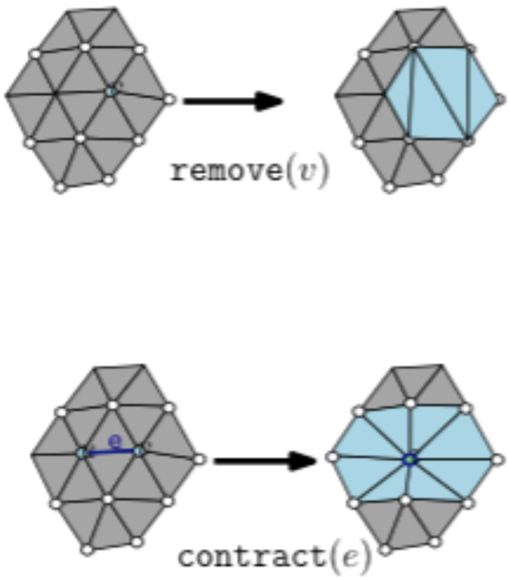
`while` $(error > \epsilon)$ { $/*$ or $(n' > C)*/$

select vertex (or edge to be contracted)

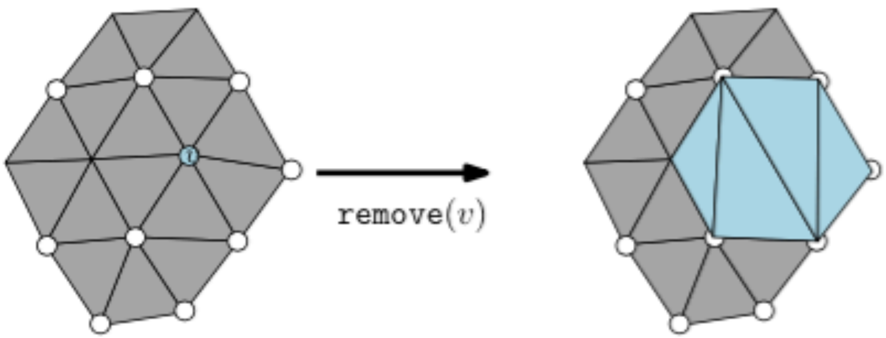
remove vertex (or contract edge)

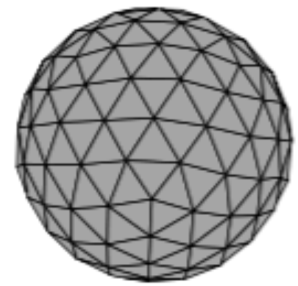
update local error for neighboring vertices (edges)

}

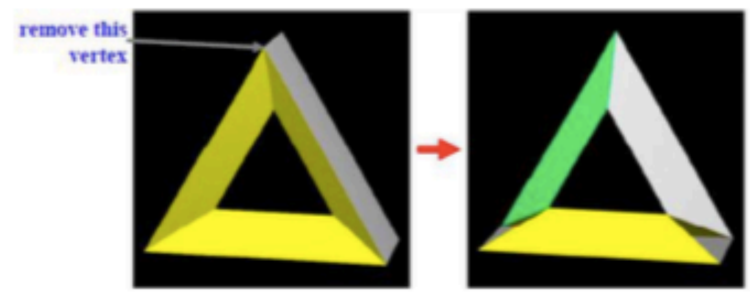


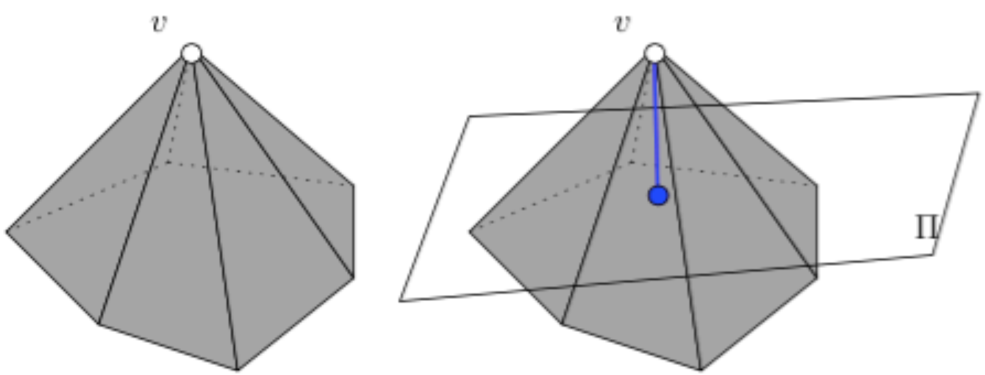
Incremental decimation – vertex removal

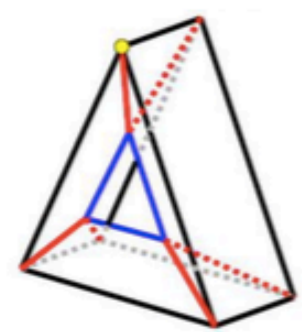




Iteratively perform vertex removals

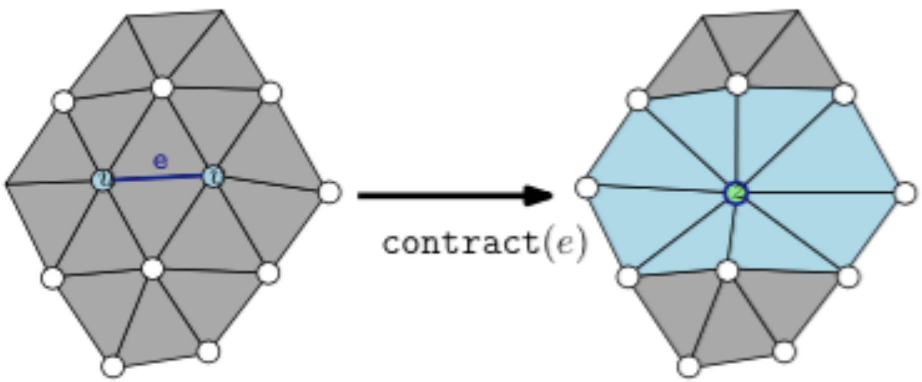


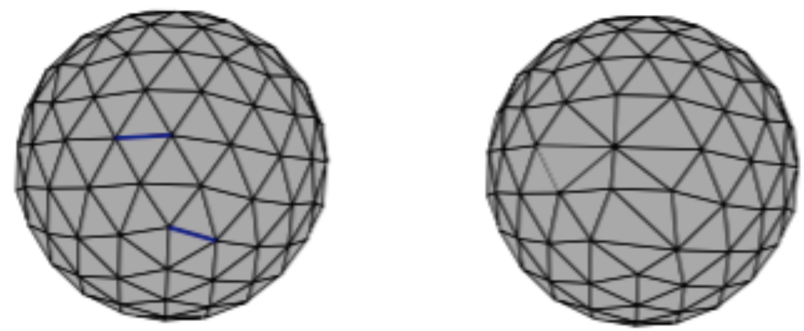




(images by C-K Shene, www.cs.mtu.edu/~shene/)

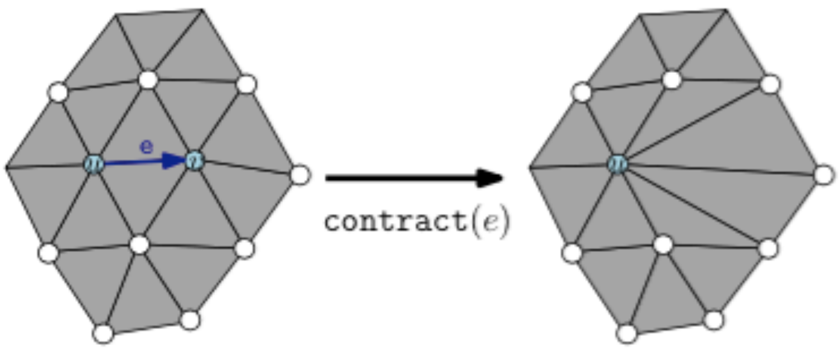
Incremental decimation – edge contraction



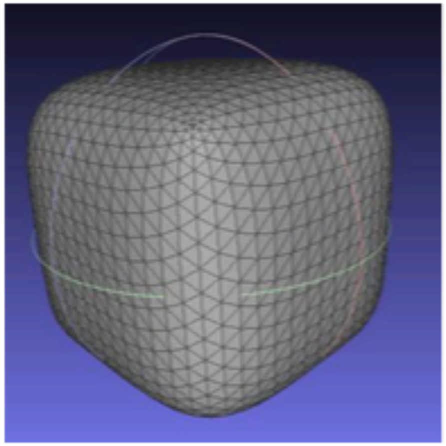


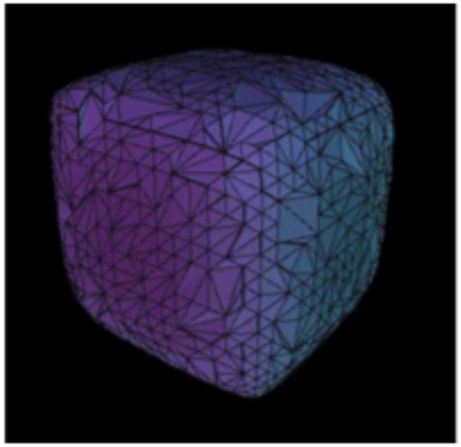
Iteratively perform edge contractions

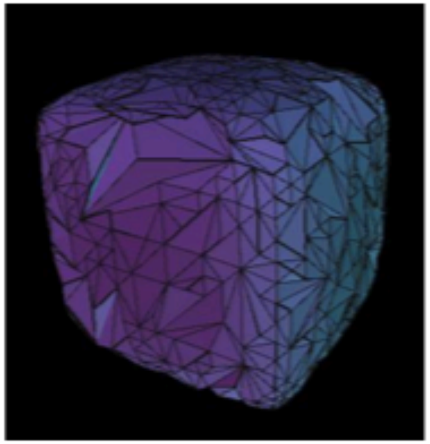
Incremental decimation – half-edge contraction



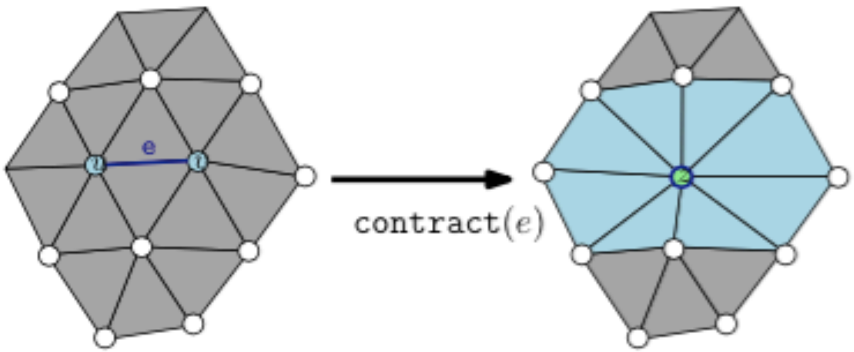
Half-edge contractions based on random selection (no geometric criteria)
Vertex locations correspond to original coordinates of input points



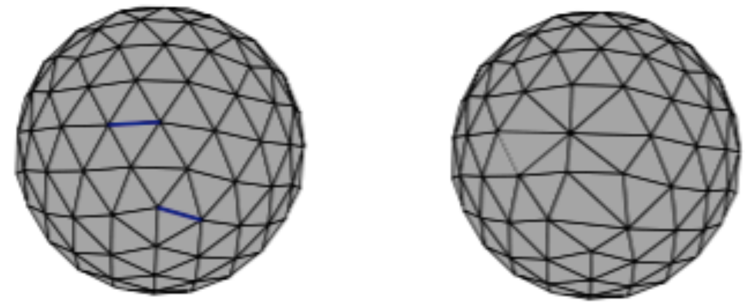




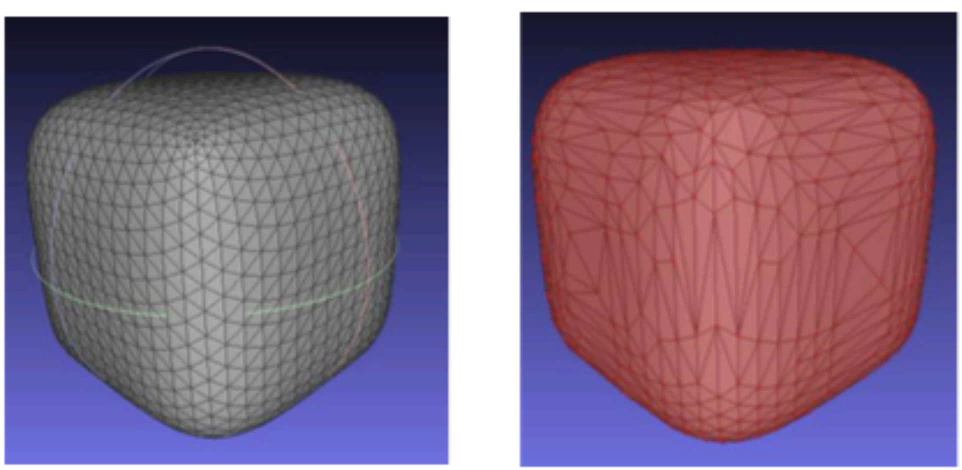
Incremental decimation – edge contraction



Select edge contractions based on local geometric criterion



Simplification based on *Quadric Error Metrics* (Garland and Heckbert, 1997)



Approximating error with quadrics

Let be $\mathbf{v} = [x \; y \; z \; 1]$ a vertex (in homogeneous coordinates)

Associate to each vertex v a 4×4 matrix $\mathbf{Q}_v$

Error at vertex v:

$$\Delta(\mathbf{v}) := \mathbf{v}^T \mathbf{Q} \mathbf{v} \qquad \text{(quadratic form)}$$

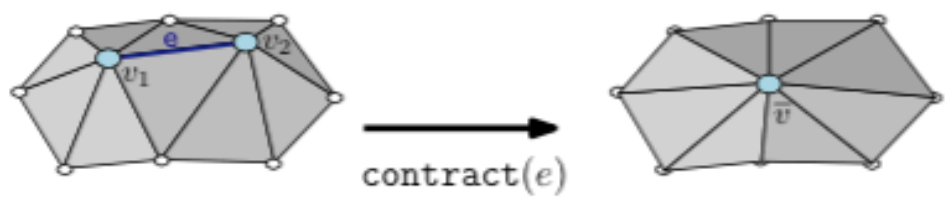
Level set surface:

$$\{\mathbf{v} \mid \Delta(\mathbf{v}) = \epsilon\} \qquad \text{(quadric surface)}$$

$$e = (v_1, v_2) \rightarrow \overline{v}$$

Associate an error to new vertex location $\overline{v}$ (use additivity)

$$\overline{\mathbf{Q}} := \mathbf{Q}_1 + \mathbf{Q}_2$$



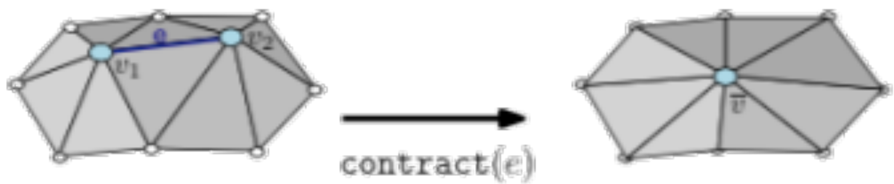
Simplification based on *Quadric Error Metrics* (Garland and Heckbert, 1997)

Approximating error with quadrics

$$\Delta(\mathbf{v}) := \mathbf{v}^T\mathbf{Q}\mathbf{v} \qquad \overline{\mathbf{Q}} := \mathbf{Q}_1 + \mathbf{Q}_2$$

$$e = (v_1, v_2) \to \overline{v}$$

Associate an error to new vertex location $\overline{v}$ (use additivity)



Find $\overline{\mathbf{v}}$ in order to minimize $\Delta(\overline{\mathbf{v}}) := \overline{\mathbf{v}}^T\mathbf{Q}\overline{\mathbf{v}}$

solve a linear system
$$\begin{cases} \dfrac{\partial\Delta(\overline{\mathbf{v}})}{\partial x} = 0 \\ \dfrac{\partial\Delta(\overline{\mathbf{v}})}{\partial y} = 0 \\ \dfrac{\partial\Delta(\overline{\mathbf{v}})}{\partial z} = 0 \end{cases}$$

$$\Delta(\overline{\mathbf{v}}) := q_{11}x^2 + 2q_{12}xy + 2q_{13}xz + 2q_{14}x + q_{22}y^2 + 2q23yz + 2q_{24}y + q_{33}z^2 + q_{44}$$

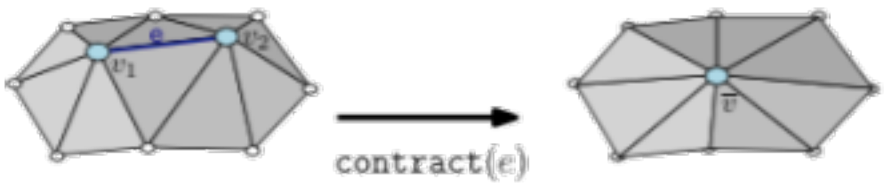
Simplification based on *Quadric Error Metrics* (Garland and Heckbert, 1997)

Approximating error with quadrics

$$\Delta(\mathbf{v}) := \mathbf{v}^T \mathbf{Q} \mathbf{v} \qquad \overline{\mathbf{Q}} := \mathbf{Q}_1 + \mathbf{Q}_2$$

$$e = (v_1, v_2) \rightarrow \overline{v}$$

Associate an error to new vertex location $\overline{v}$ (use additivity)



Find $\overline{\mathbf{v}}$ in order to minimize $\quad \Delta(\overline{\mathbf{v}}) := \overline{\mathbf{v}}^T \mathbf{Q} \overline{\mathbf{v}}$

solve a linear system

$$\begin{cases} \dfrac{\partial \Delta(\overline{\mathbf{v}})}{\partial x} = 0 \\ \dfrac{\partial \Delta(\overline{\mathbf{v}})}{\partial y} = 0 \\ \dfrac{\partial \Delta(\overline{\mathbf{v}})}{\partial z} = 0 \end{cases} \longleftrightarrow \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{24} \\ 0 & 0 & 0 & 1 \end{bmatrix} \overline{\mathbf{v}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$\Delta(\overline{\mathbf{v}}) := q_{11}x^2 + 2q_{12}xy + 2q_{13}xz + 2q_{14}x + q_{22}y^2 + 2q_{23}yz + 2q_{24}y + q_{33}z^2 + q_{44}$$

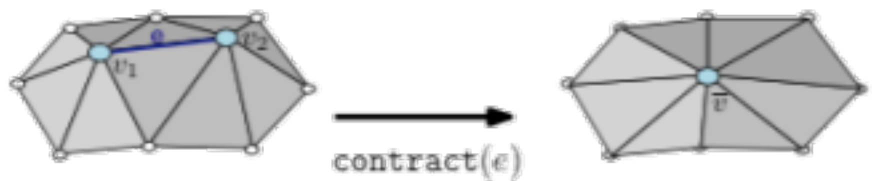
Simplification based on *Quadric Error Metrics*
(Garland and Heckbert, 1997)

Approximating error with quadrics

$\Delta(\mathbf{v}) := \mathbf{v}^T \mathbf{Q} \mathbf{v}$ $\qquad$ $\overline{\mathbf{Q}} := \mathbf{Q}_1 + \mathbf{Q}_2$

$e = (v_1, v_2) \rightarrow \overline{v}$

Associate an error to new vertex location $\overline{v}$ (use additivity)



Find $\overline{\mathbf{v}}$ in order to minimize $\qquad$ $\Delta(\overline{\mathbf{v}}) := \overline{\mathbf{v}}^T \mathbf{Q} \overline{\mathbf{v}}$

$$\text{if} \quad \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{24} \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ is invertible} \longrightarrow \quad \overline{\mathbf{v}} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{24} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

otherwise, select $\qquad$ $\overline{\mathbf{v}} \in \{v_1, v_2, v_1 + v_2\}$

Simplification based on *Quadric Error Metrics* (Garland and Heckbert, 1997)

Approximating error with quadrics

$$\Delta(\mathbf{v}) = \Delta([v_x\ v_y\ v_z\ 1]^\mathsf{T}) = \sum_{\mathbf{p}\in\text{planes}(\mathbf{v})} (\mathbf{p}^\mathsf{T}\mathbf{v})^2$$

equation of a plane

$$ax + by + cz + d = 0$$

$$a^2 + b^2 + c^2 = 1$$

normalized vector

$$\begin{aligned}
\Delta(\mathbf{v}) &= \sum_{\mathbf{p}\in\text{planes}(\mathbf{v})} (\mathbf{v}^\mathsf{T}\mathbf{p})(\mathbf{p}^\mathsf{T}\mathbf{v}) \\
&= \sum_{\mathbf{p}\in\text{planes}(\mathbf{v})} \mathbf{v}^\mathsf{T}(\mathbf{p}\mathbf{p}^\mathsf{T})\mathbf{v} \\
&= \mathbf{v}^\mathsf{T}\left(\sum_{\mathbf{p}\in\text{planes}(\mathbf{v})} \mathbf{K}_\mathbf{p}\right)\mathbf{v}
\end{aligned}$$

$$\mathbf{p} = [a\ b\ c\ 1]$$

$$\mathbf{K}_\mathbf{p} = \mathbf{p}\mathbf{p}^\mathsf{T} = \begin{bmatrix} a^2 & ab & ac & ad \\ ab & b^2 & bc & bd \\ ac & bc & c^2 & cd \\ ad & bd & cd & d^2 \end{bmatrix}$$

error metric in quadratic form

Simplification based on *Quadric Error Metrics* (Garland and Heckbert, 1997)

Quadric error simplification: algorithm

Compute error matrix for $\mathbf{Q}_p$ each point p.

For each candidate edge *pq* do

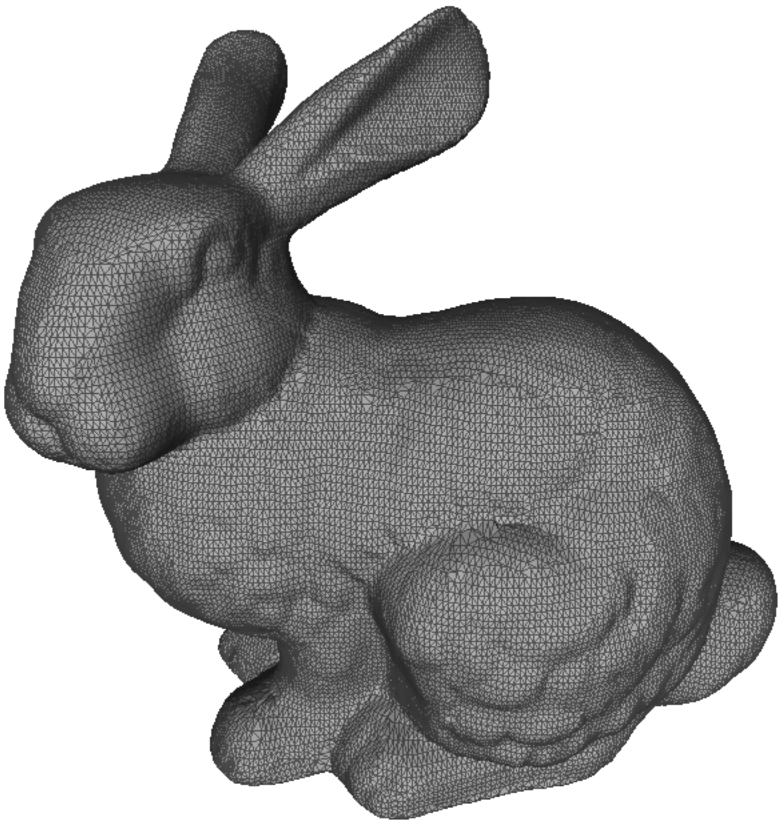
- Minimize: $\Delta(r) = r^T \left(\mathbf{Q}_p + \mathbf{Q}_q\right) r/2$ to find the optimal location of the intermediate vertex.
- Store the error $\Delta(r)$ in a priority queue (heap)

Iterate:

- Pick the edge with the smallest error from the queue
- Collapse the edge and place the new collapsed vertex
- Update the error metrics of adjacent edges

Quadric error simplification: algorithm

- Implemented in Meshlab[1]

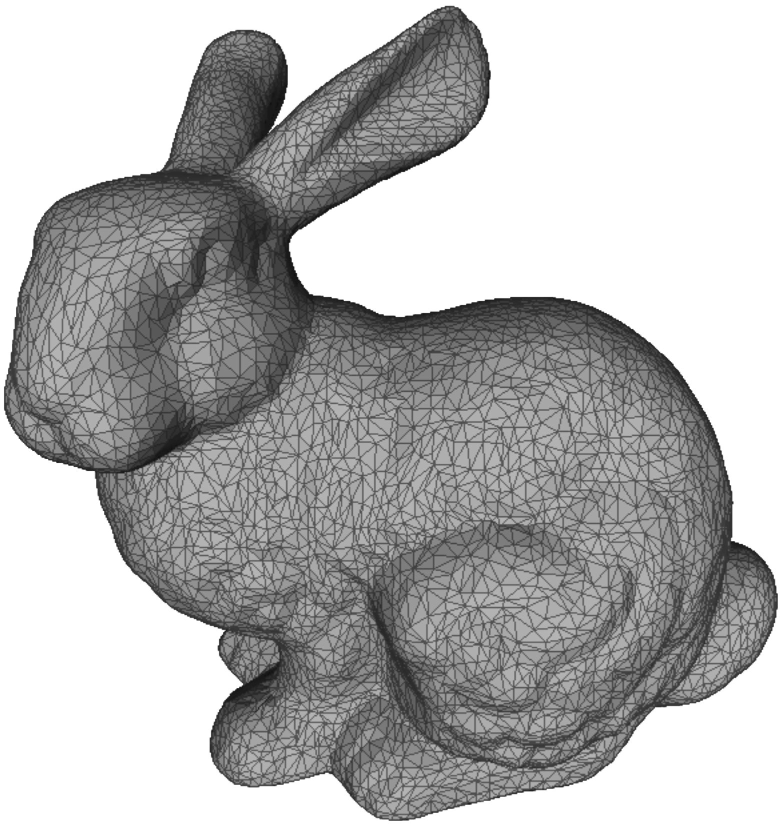


35k vertices

[1]http://www.meshlab.net/
ISTI, CNR, Pisa

Quadric error simplification: algorithm

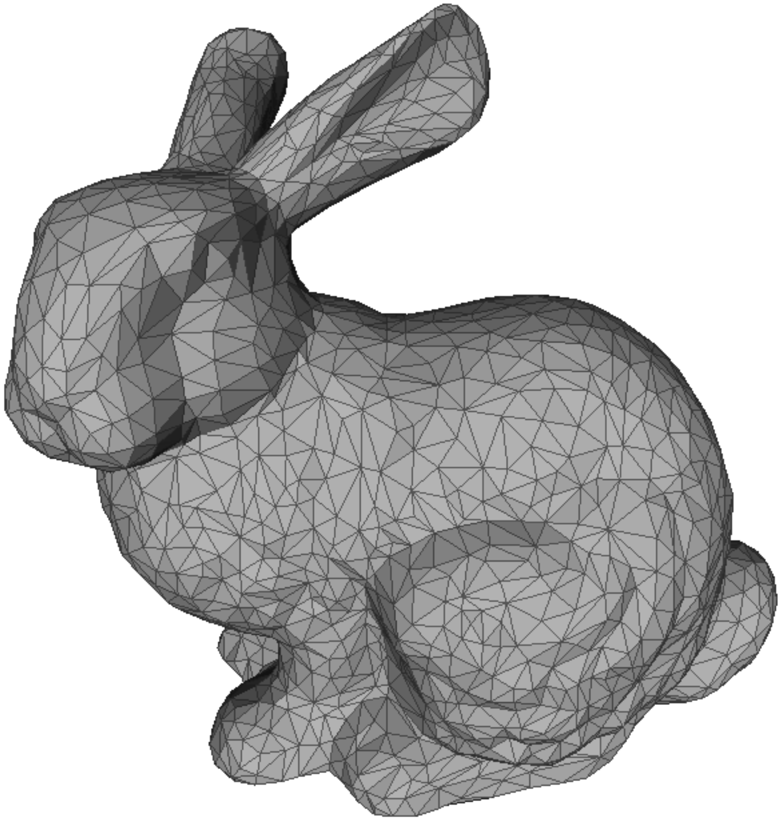
- Implemented in Meshlab



8.7k vertices

Quadric error simplification: algorithm

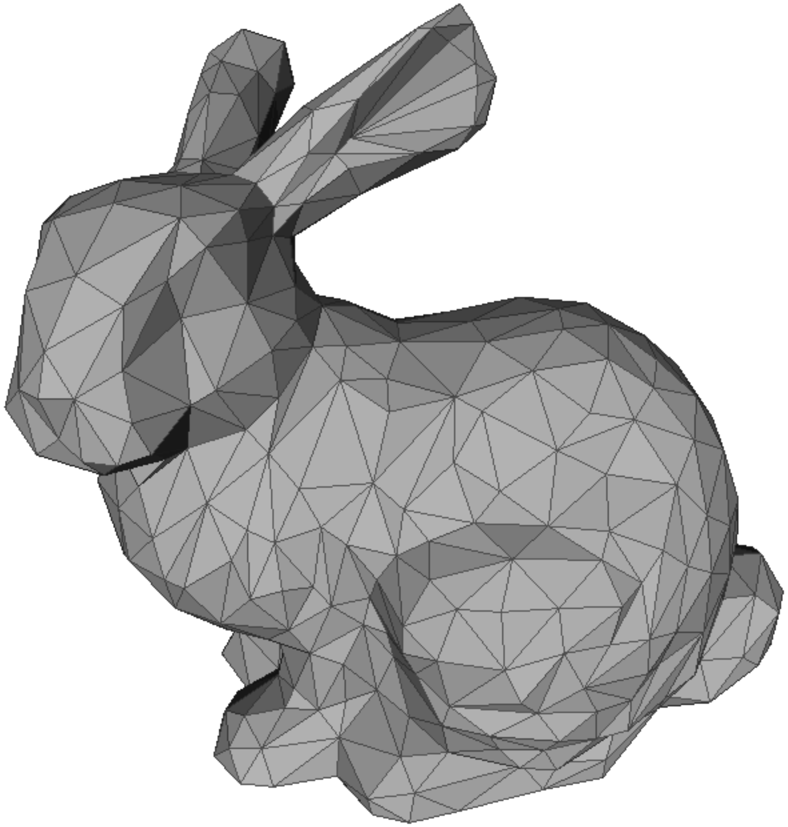
- Implemented in Meshlab



2,1k vertices

Quadric error simplification: algorithm

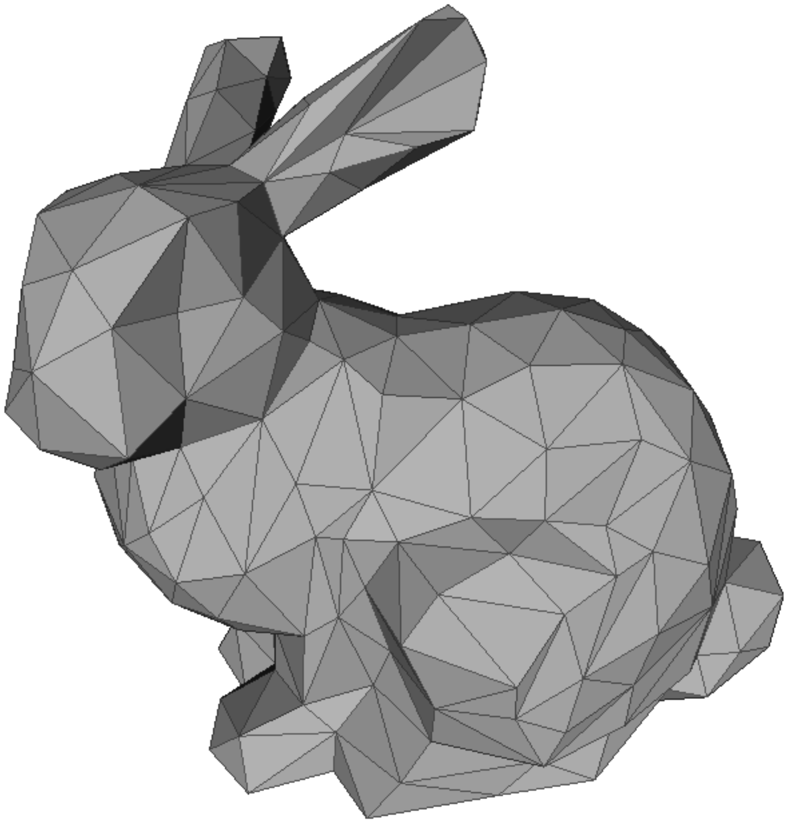
- Implemented in Meshlab



560 vertices

Quadric error simplification: algorithm

- Implemented in Meshlab

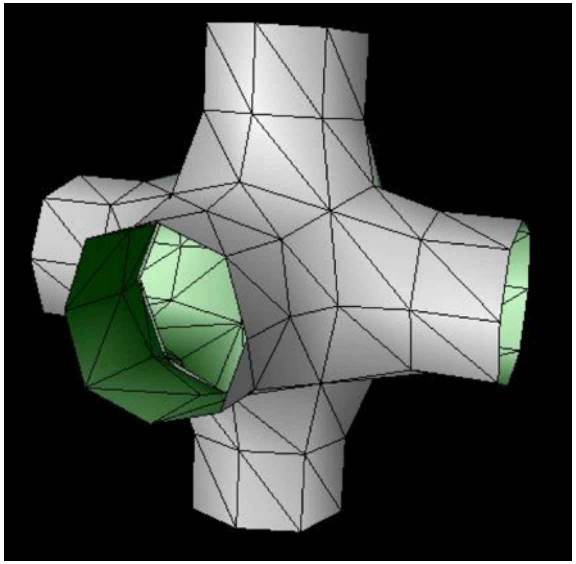


280 vertices

Subdivision Surfaces

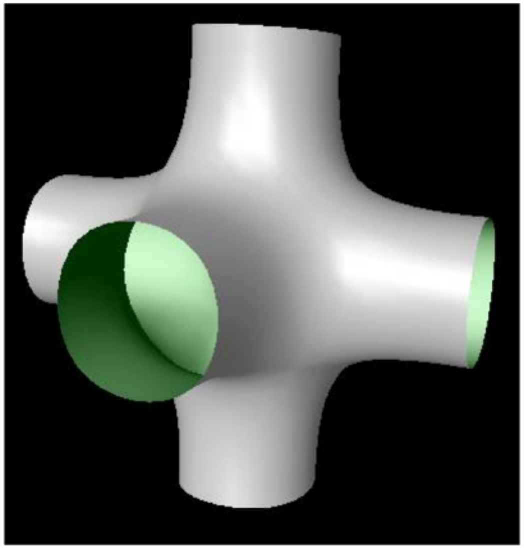
Provide a trade-off between Smooth and Mesh techniques:

- Inherently **continuous**
- Intuitive controls (control mesh)
- Can model shapes with arbitrary topology



Modeling

Subdivision surfaces

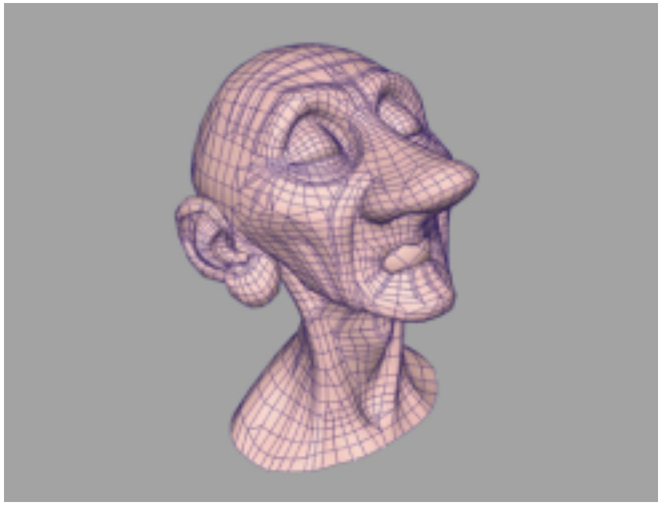


Rendering

Subdivision Surfaces

Provide a trade-off between Smooth and Mesh techniques:

- Inherently **continuous**
- Intuitive controls (control mesh)
- Can model shapes with arbitrary topology



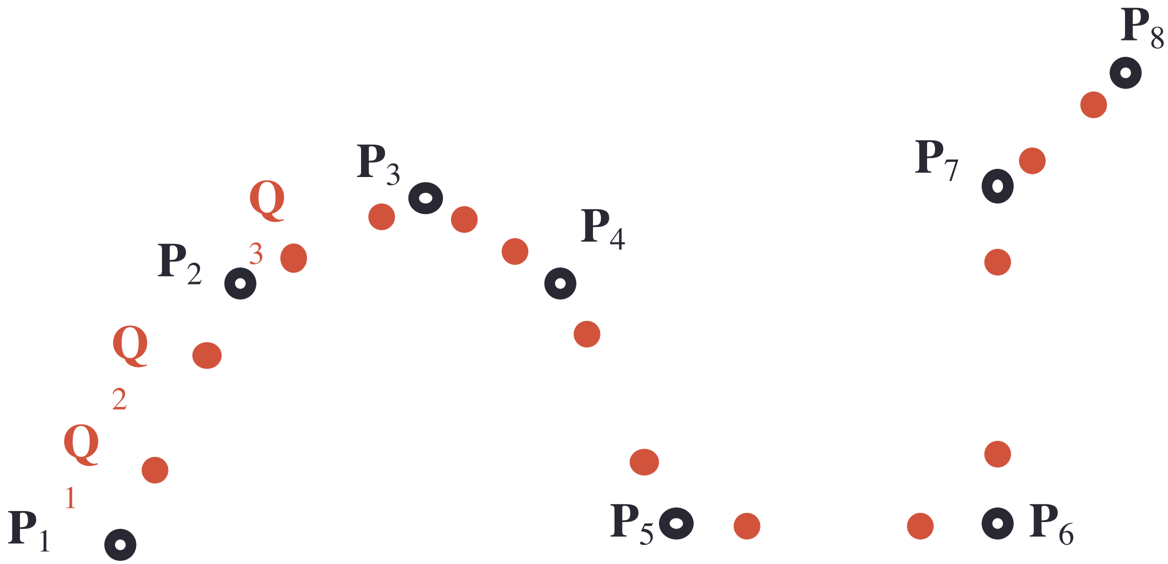
Subdivision surfaces

Modeling

Rendering

Subdivision Curves

Uniform B-spline of order 2:



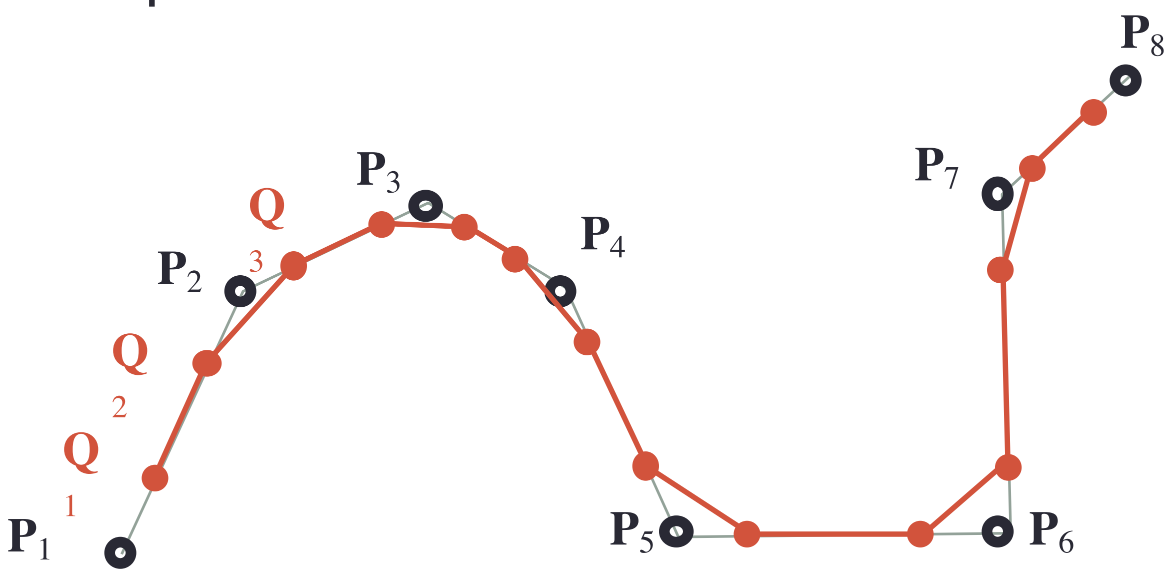
Chaikin's algorithm for Quadratic Uniform B-splines:

$$j \text{ odd: } \mathbf{Q}_j = \frac{3}{4}\mathbf{P}_{(j+1)/2} + \frac{1}{4}\mathbf{P}_{(j+3)/2}$$

$$j \text{ even: } \mathbf{Q}_j = \frac{1}{4}\mathbf{P}_{j/2} + \frac{3}{4}\mathbf{P}_{(j+2)/2}$$

Subdivision Curves

Uniform B-spline of order 2:



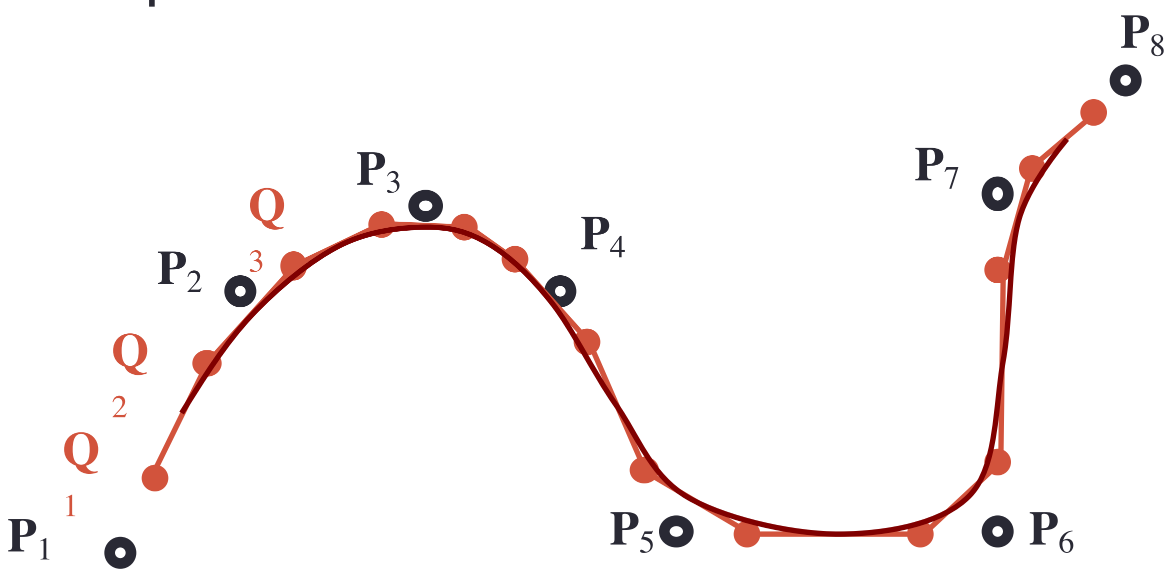
Chaikin's algorithm for Quadratic Uniform B-splines:

Given n points: $\mathbf{P}_i, i \in (1, 2, \ldots, n)$

Produce 2(n-1) points: $\mathbf{Q}_j, j \in (1, 2, \ldots, 2n-2)$

Subdivision Curves

Uniform B-spline of order 2:



Chaikin's algorithm for Quadratic Uniform B-splines:

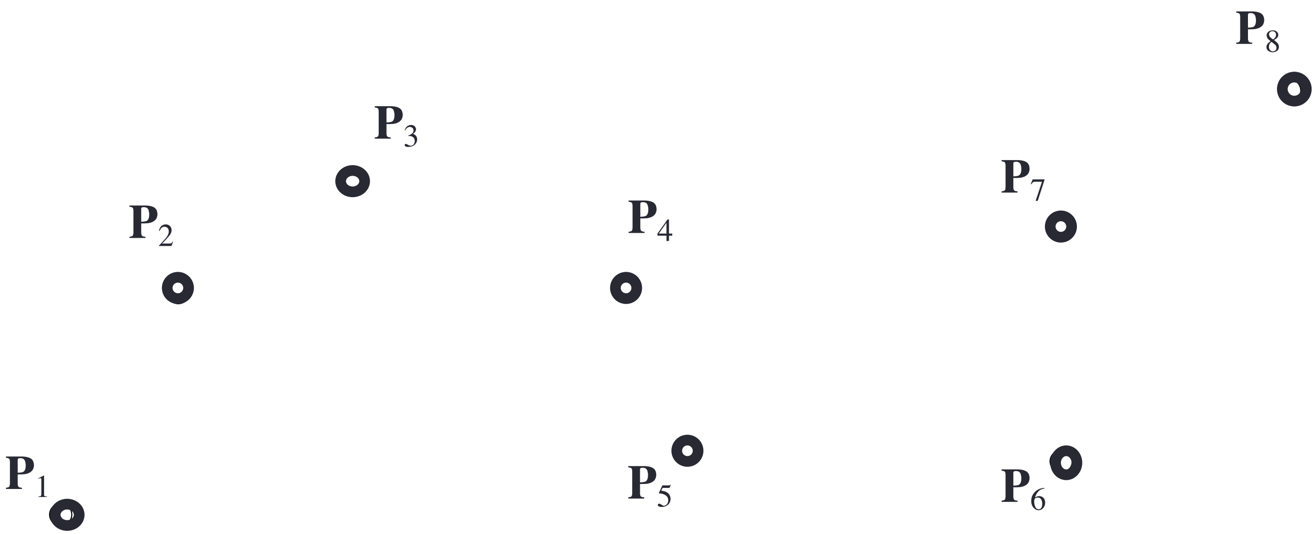
Given n points: $\mathbf{P}_i, i \in (1, 2, \ldots, n)$

Produce 2(n-1) points: $\mathbf{Q}_j, j \in (1, 2, \ldots, 2n-2)$

Let $\mathbf{P} = \mathbf{Q}$ and iterate until number of points reaches desired accuracy.

Subdivision Curves

Uniform B-spline of order 3:

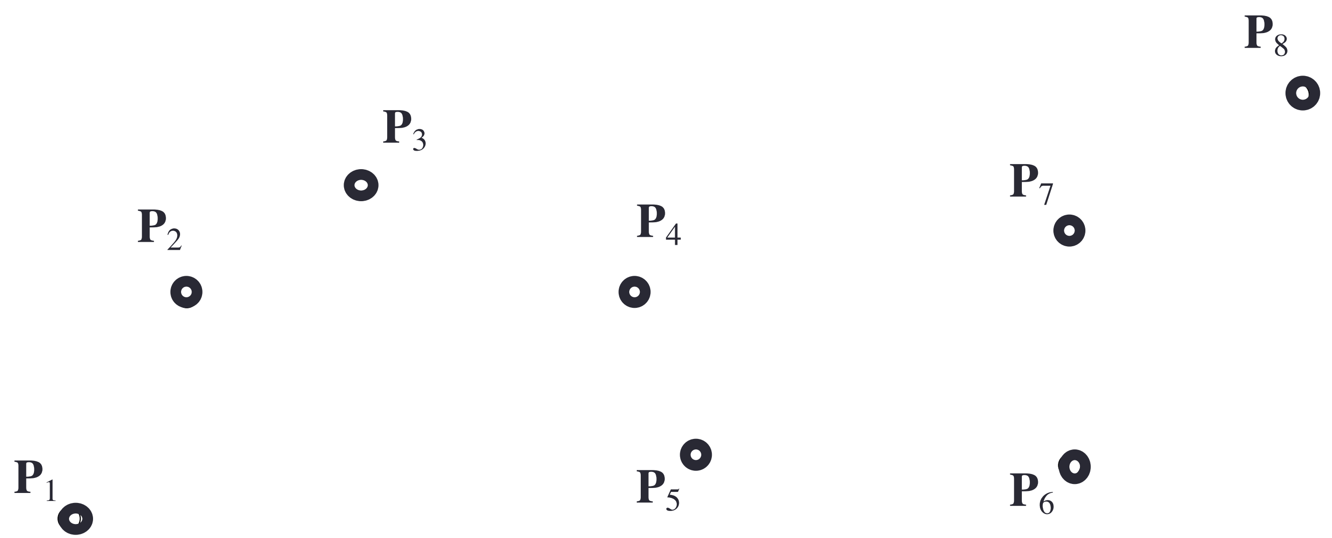


Given n points: $\mathbf{P}_i, i \in (1, 2, \ldots, n)$

Produce 2(n-1)-1 points: $\mathbf{Q}_j, j \in (1, 2, \ldots, 2n-3)$

Subdivision Curves

Uniform B-spline of order 3:



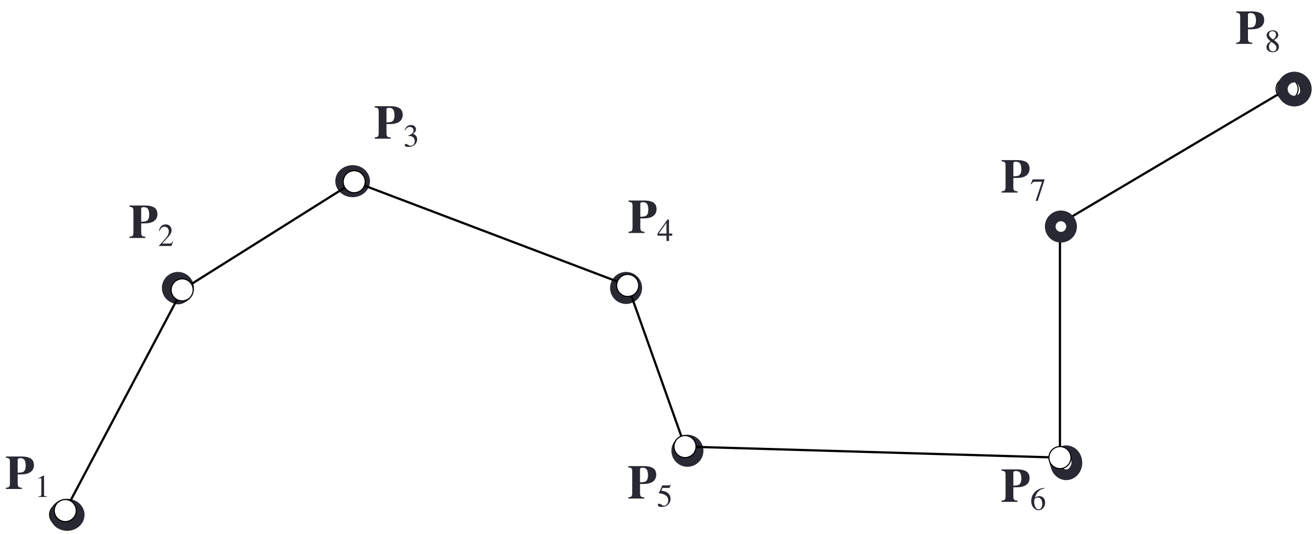
At each iteration produce 2(n-1)-1 points:

$$\mathbf{Q}_{2i-1} = \frac{1}{2}\mathbf{P}_i + \frac{1}{2}\mathbf{P}_{i+1}$$

$$\mathbf{Q}_{2i} = \frac{1}{8}\mathbf{P}_{i-1} + \frac{3}{4}\mathbf{P}_i + \frac{1}{8}\mathbf{P}_{i+1}$$

Subdivision Curves

Uniform B-spline of order 3:



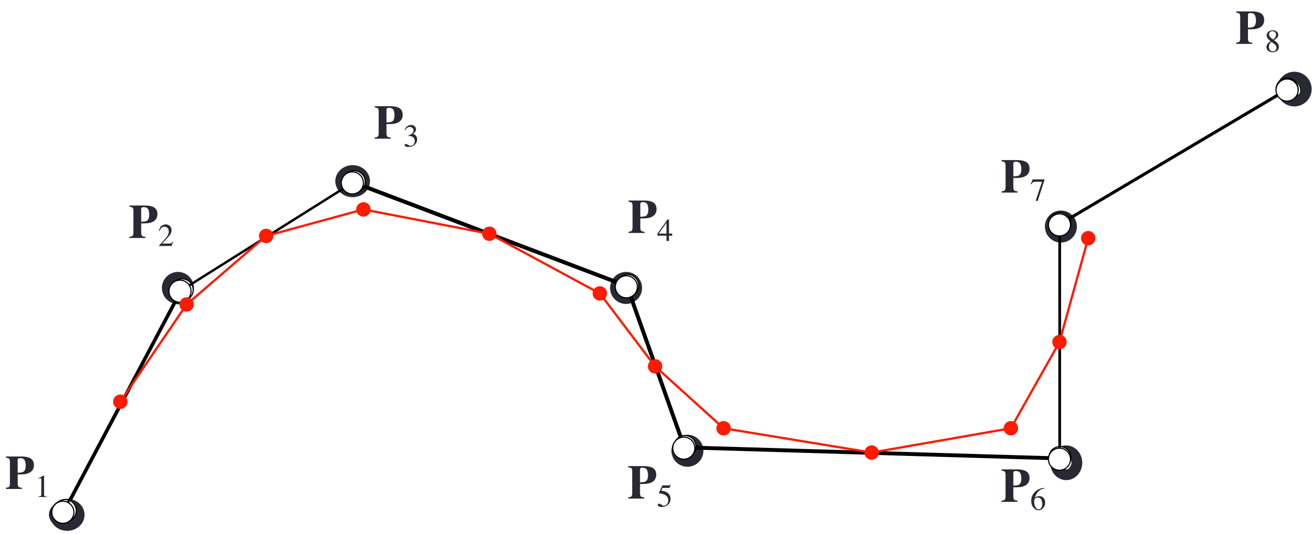
At each iteration produce 2(n-1)-1 points:

$$\mathbf{Q}_{2i-1} = \frac{1}{2}\mathbf{P}_i + \frac{1}{2}\mathbf{P}_{i+1}$$

$$\mathbf{Q}_{2i} = \frac{1}{8}\mathbf{P}_{i-1} + \frac{3}{4}\mathbf{P}_i + \frac{1}{8}\mathbf{P}_{i+1}$$

Subdivision Curves

Uniform B-spline of order 3:



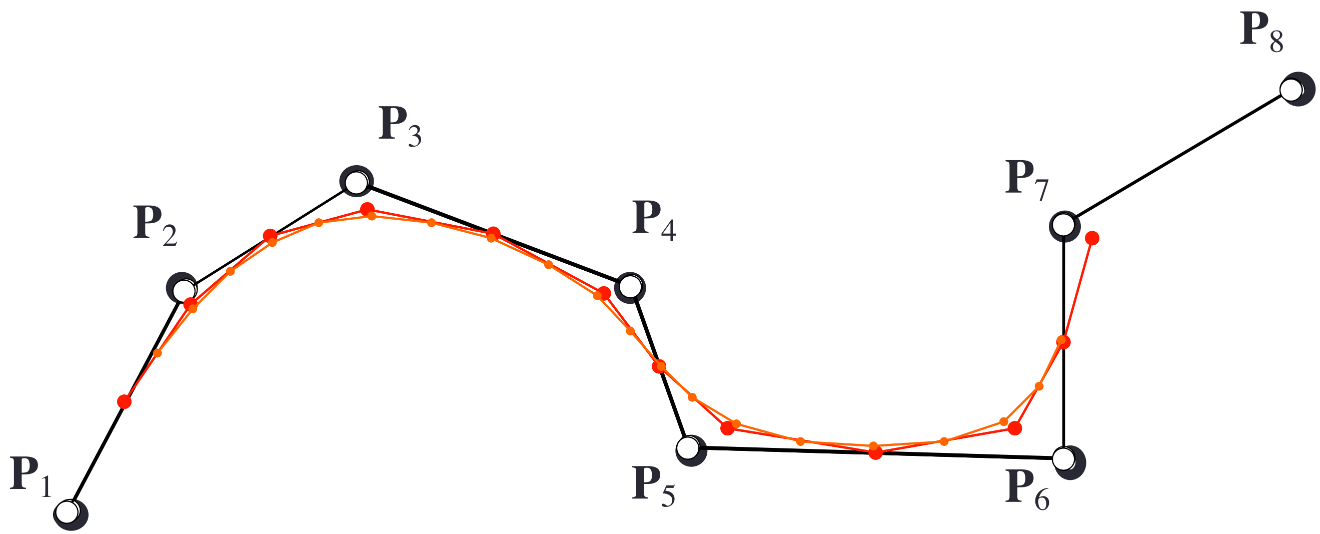
At each iteration produce 2(n-1)-1 points:

$$\mathbf{Q}_{2i-1} = \frac{1}{2}\mathbf{P}_i + \frac{1}{2}\mathbf{P}_{i+1}$$

$$\mathbf{Q}_{2i} = \frac{1}{8}\mathbf{P}_{i-1} + \frac{3}{4}\mathbf{P}_i + \frac{1}{8}\mathbf{P}_{i+1}$$

Subdivision Curves

Uniform B-spline of order 3:



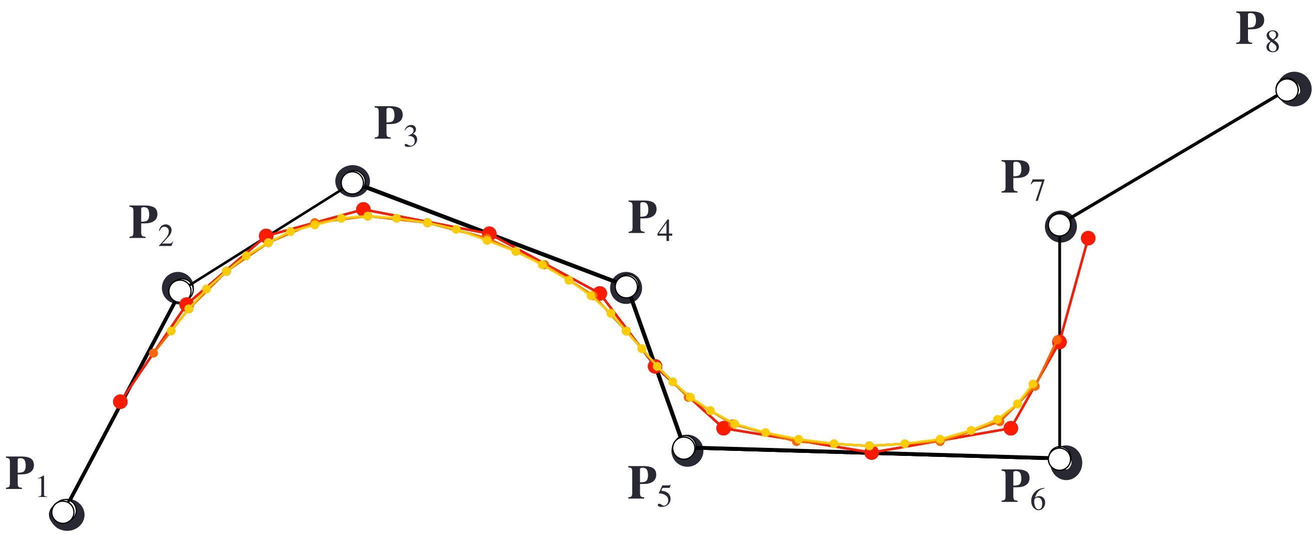
At each iteration produce 2(n-1)-1 points:

$$\mathbf{Q}_{2i-1} = \frac{1}{2}\mathbf{P}_i + \frac{1}{2}\mathbf{P}_{i+1}$$

$$\mathbf{Q}_{2i} = \frac{1}{8}\mathbf{P}_{i-1} + \frac{3}{4}\mathbf{P}_i + \frac{1}{8}\mathbf{P}_{i+1}$$

Subdivision Curves

Uniform B-spline of order 3:



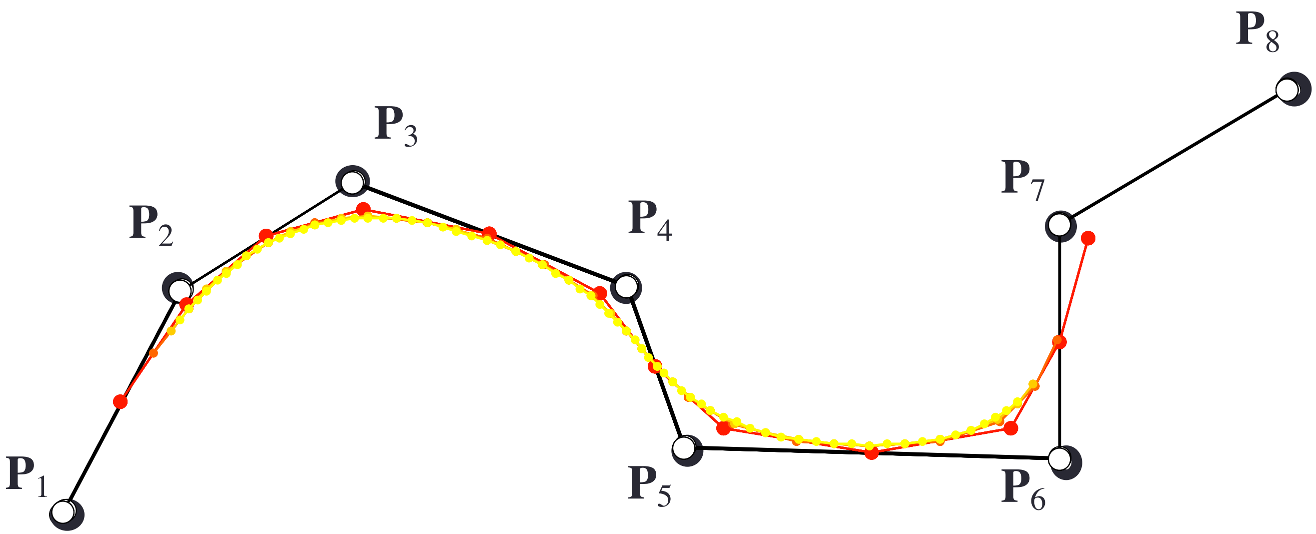
At each iteration produce 2(n-1)-1 points:

$$\mathbf{Q}_{2i-1} = \frac{1}{2}\mathbf{P}_i + \frac{1}{2}\mathbf{P}_{i+1}$$

$$\mathbf{Q}_{2i} = \frac{1}{8}\mathbf{P}_{i-1} + \frac{3}{4}\mathbf{P}_i + \frac{1}{8}\mathbf{P}_{i+1}$$

Subdivision Curves

Uniform B-spline of order 3:



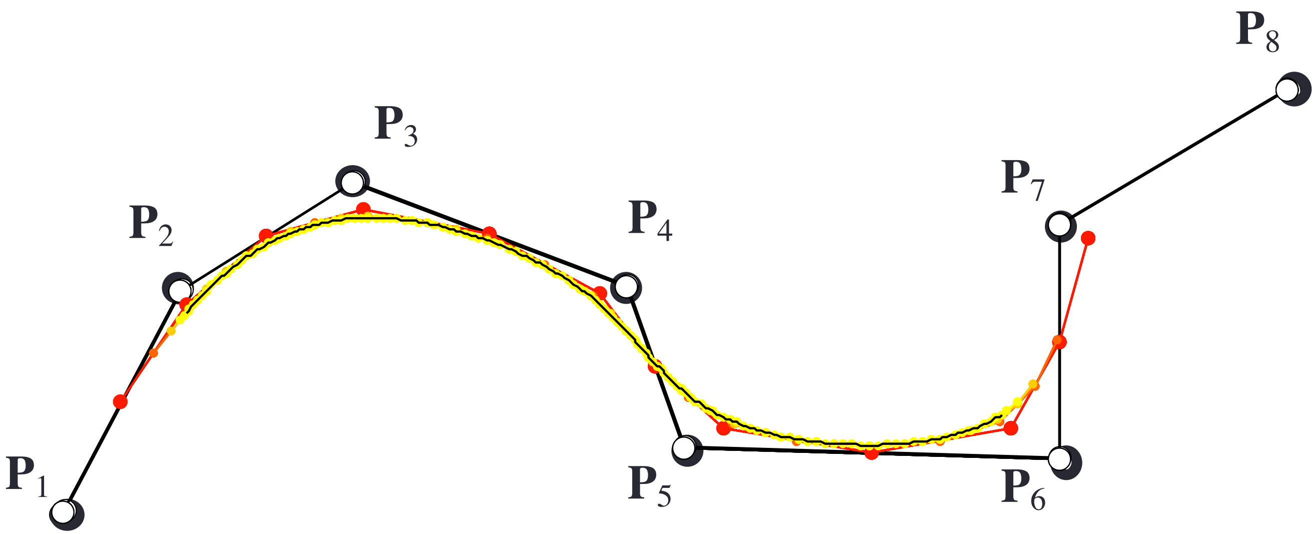
At each iteration produce 2(n-1)-1 points:

$$\mathbf{Q}_{2i-1} = \frac{1}{2}\mathbf{P}_i + \frac{1}{2}\mathbf{P}_{i+1}$$

$$\mathbf{Q}_{2i} = \frac{1}{8}\mathbf{P}_{i-1} + \frac{3}{4}\mathbf{P}_i + \frac{1}{8}\mathbf{P}_{i+1}$$

Subdivision Curves

Uniform B-spline of order 3:



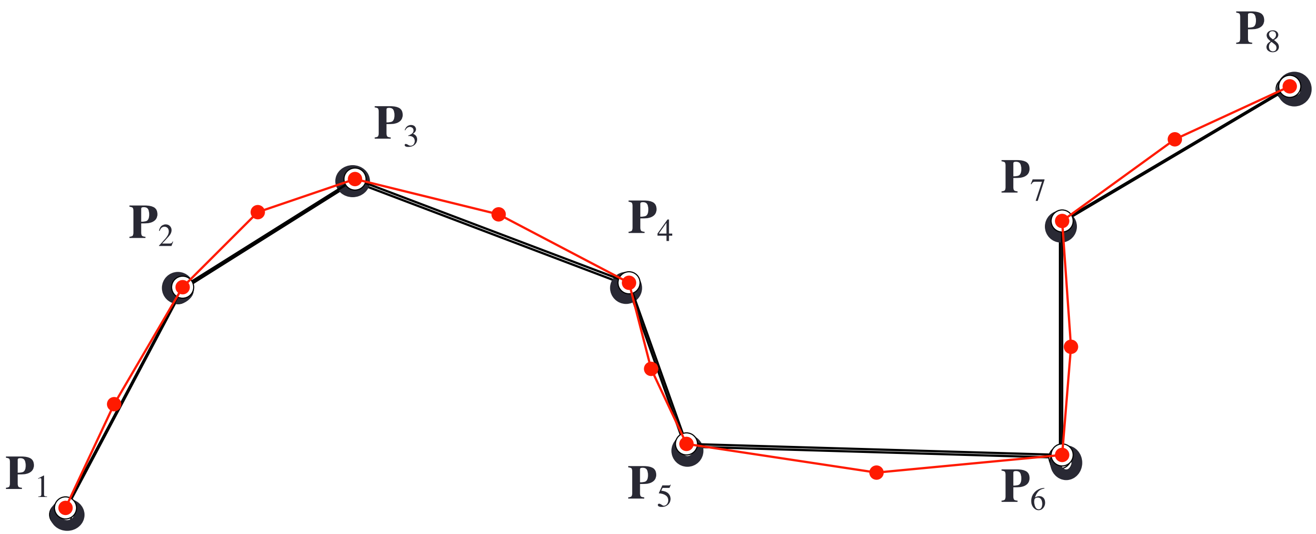
At each iteration produce 2(n-1)-1 points:

$$\mathbf{Q}_{2i-1} = \frac{1}{2}\mathbf{P}_i + \frac{1}{2}\mathbf{P}_{i+1}$$

$$\mathbf{Q}_{2i} = \frac{1}{8}\mathbf{P}_{i-1} + \frac{3}{4}\mathbf{P}_i + \frac{1}{8}\mathbf{P}_{i+1}$$

Subdivision Curves

Interpolating curves:

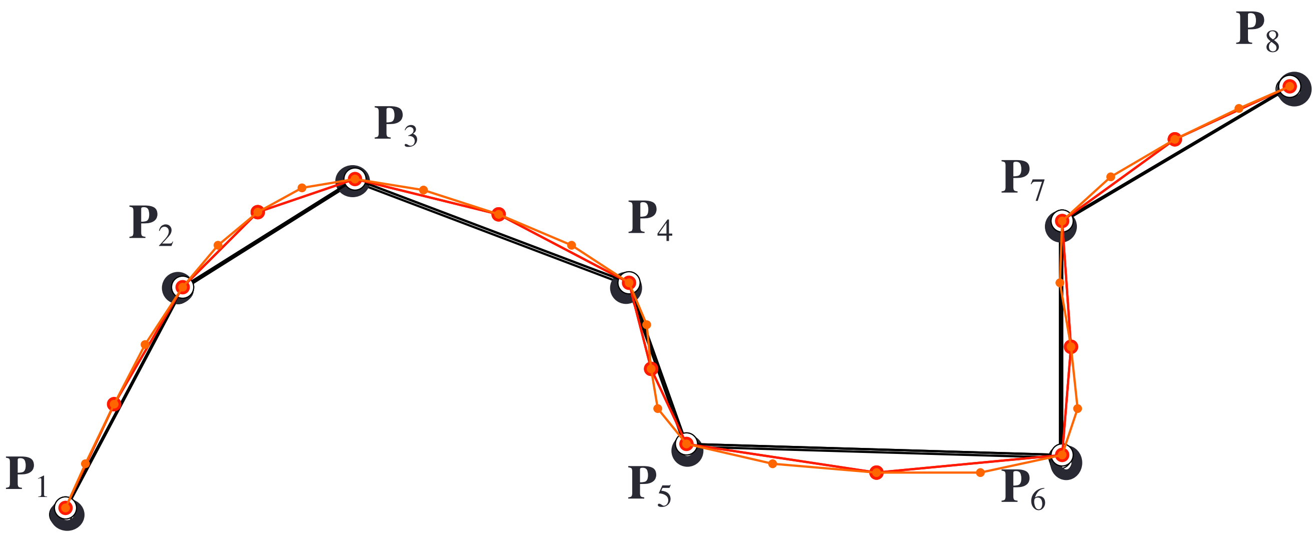


In matrix form: for every 4 consecutive old points, produce 2 new points:

$$\begin{pmatrix} \mathbf{Q}_1 \\ \mathbf{Q}_2 \end{pmatrix} = \frac{1}{16} \begin{pmatrix} 0 & 16 & 0 & 0 \\ -1 & 9 & 9 & -1 \end{pmatrix} \begin{pmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \\ \mathbf{P}_4 \end{pmatrix}$$

Subdivision Curves

Interpolating curves:

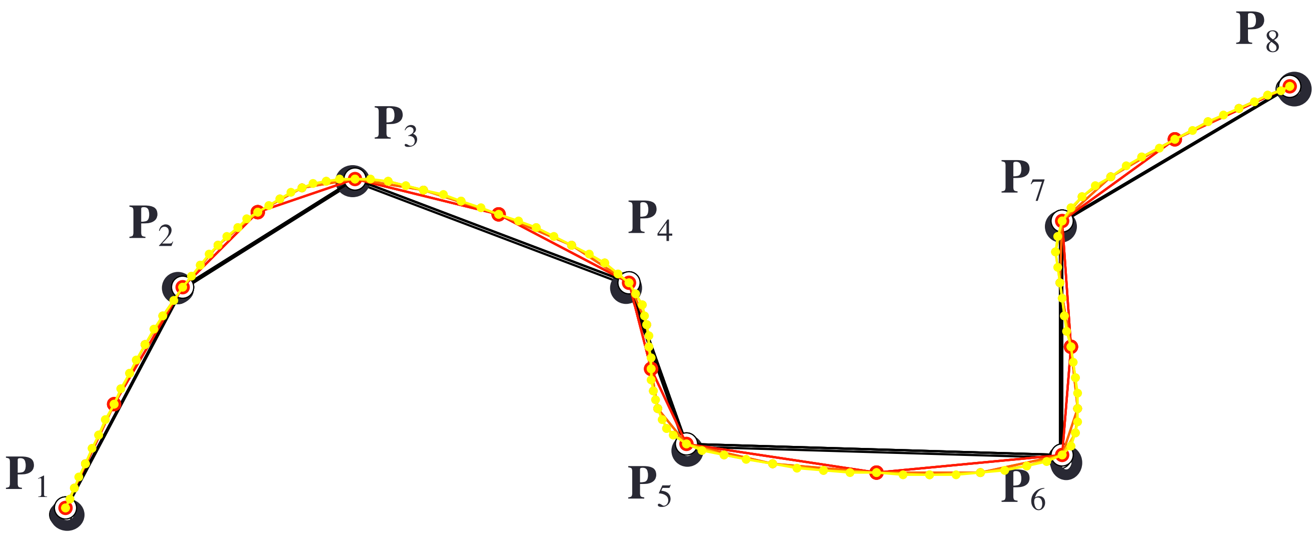


In matrix form: for every 4 consecutive old points, produce 2 new points:

$$\begin{pmatrix} \mathbf{Q}_1 \\ \mathbf{Q}_2 \end{pmatrix} = \frac{1}{16} \begin{pmatrix} 0 & 16 & 0 & 0 \\ -1 & 9 & 9 & -1 \end{pmatrix} \begin{pmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \\ \mathbf{P}_4 \end{pmatrix}$$

Subdivision Curves

Interpolating curves:

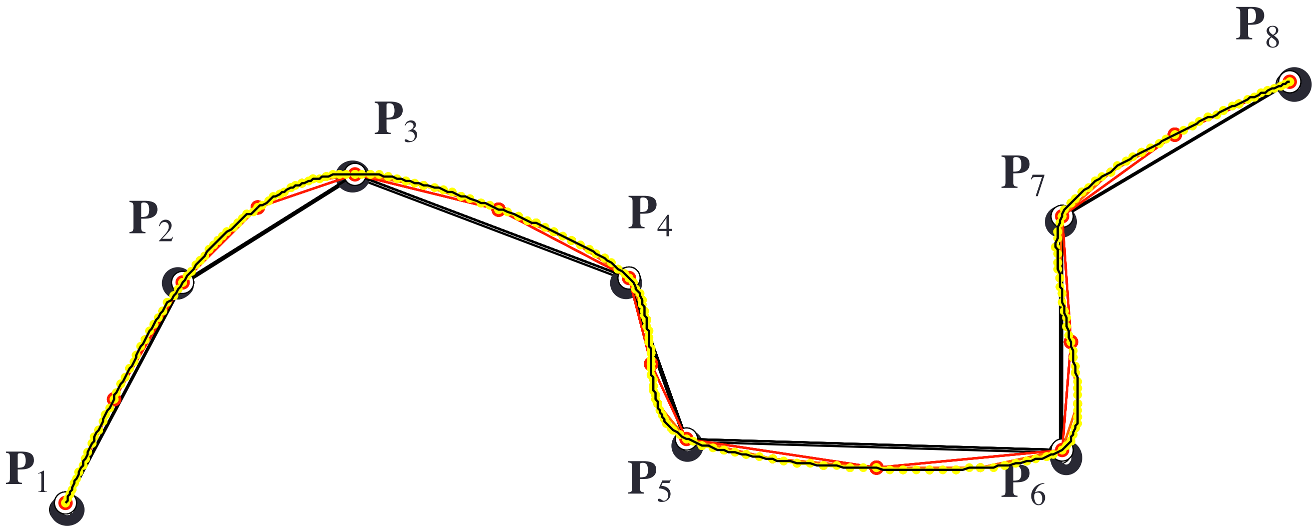


In matrix form: for every 4 consecutive old points, produce 2 new points:

$$\begin{pmatrix} \mathbf{Q}_1 \\ \mathbf{Q}_2 \end{pmatrix} = \frac{1}{16} \begin{pmatrix} 0 & 16 & 0 & 0 \\ -1 & 9 & 9 & -1 \end{pmatrix} \begin{pmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \\ \mathbf{P}_4 \end{pmatrix}$$

Subdivision Curves

Interpolating curves:

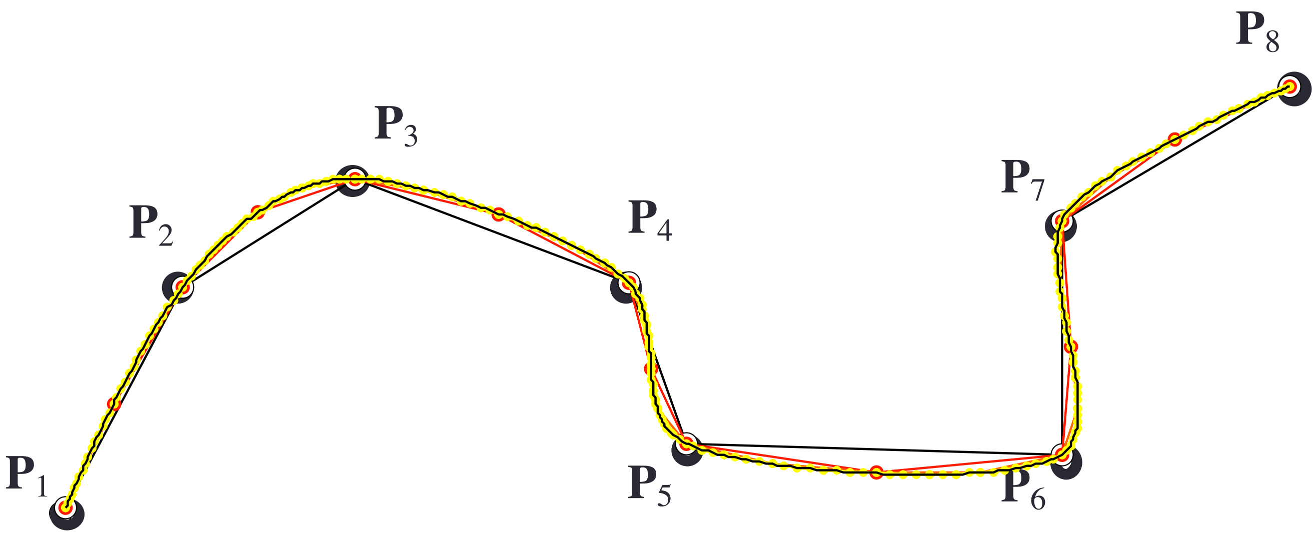


In matrix form: for every 4 consecutive old points, produce 2 new points:

$$\begin{pmatrix} \mathbf{Q}_1 \\ \mathbf{Q}_2 \end{pmatrix} = \frac{1}{16} \begin{pmatrix} 0 & 16 & 0 & 0 \\ -1 & 9 & 9 & -1 \end{pmatrix} \begin{pmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \\ \mathbf{P}_4 \end{pmatrix}$$

Subdivision Curves

Interpolating curves:

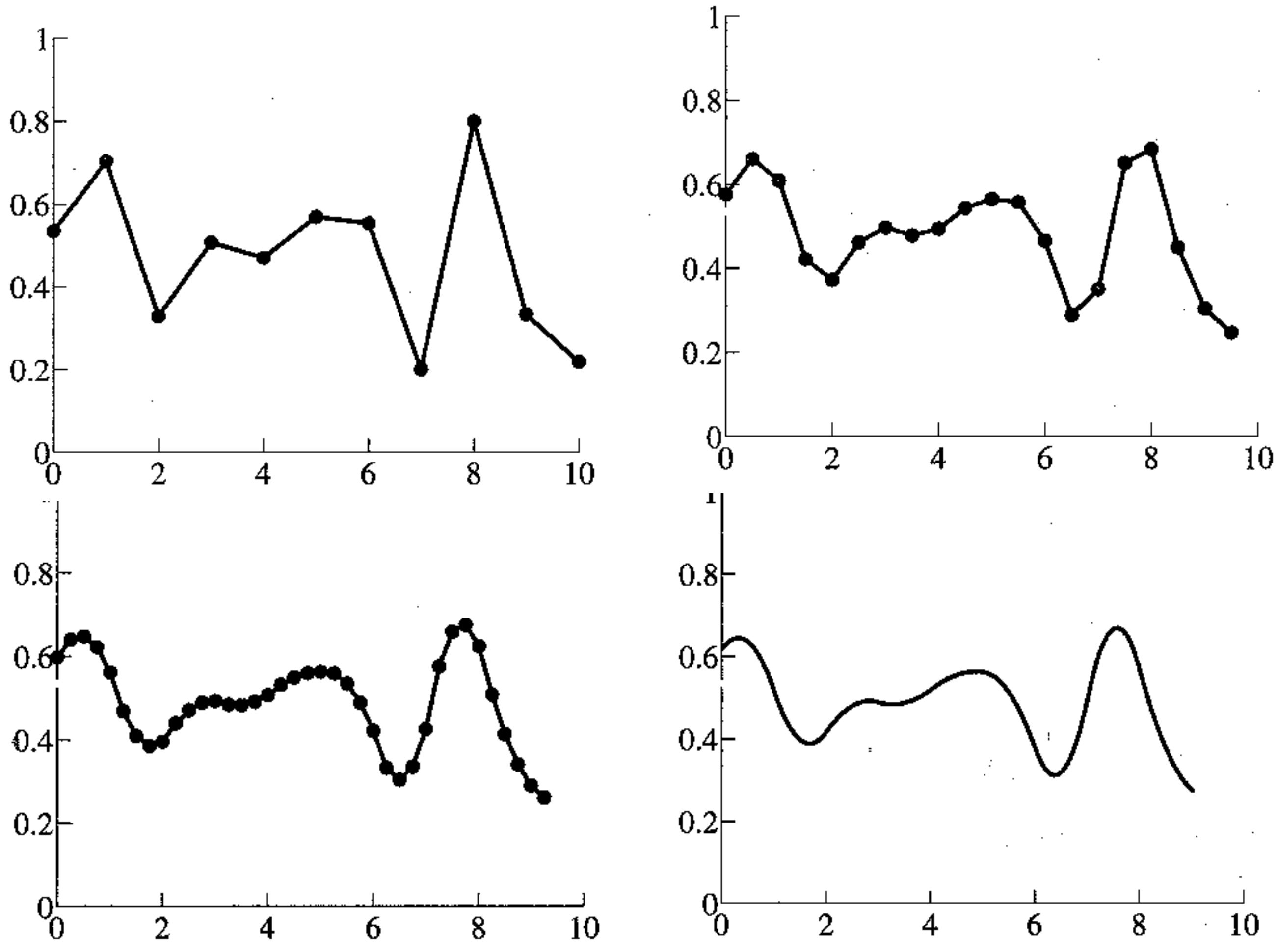


Note:

Before starting, make a copy of first and last points.
At each iteration, copy the first and last points.

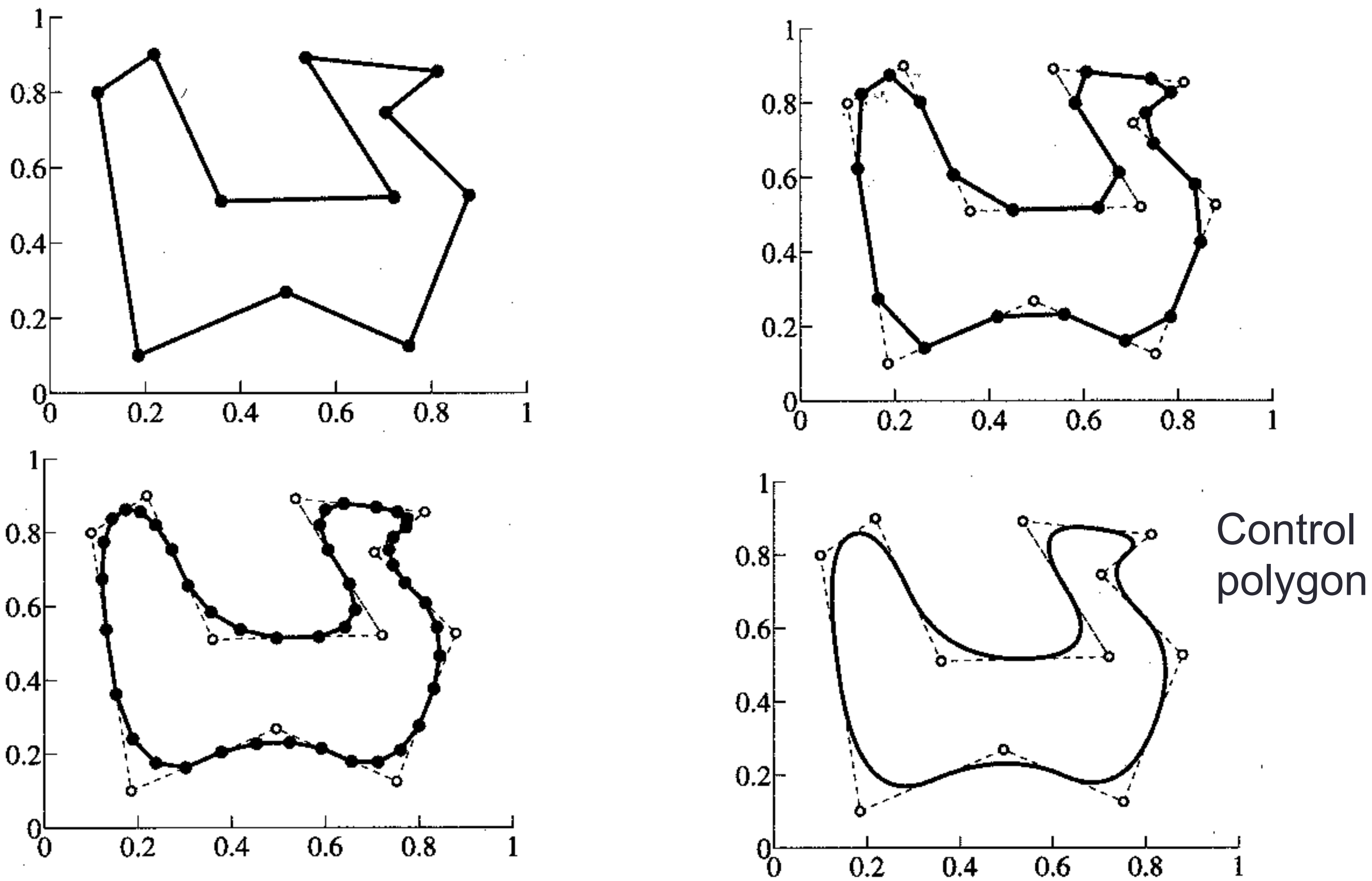
Examples

Chaikin's scheme



Examples

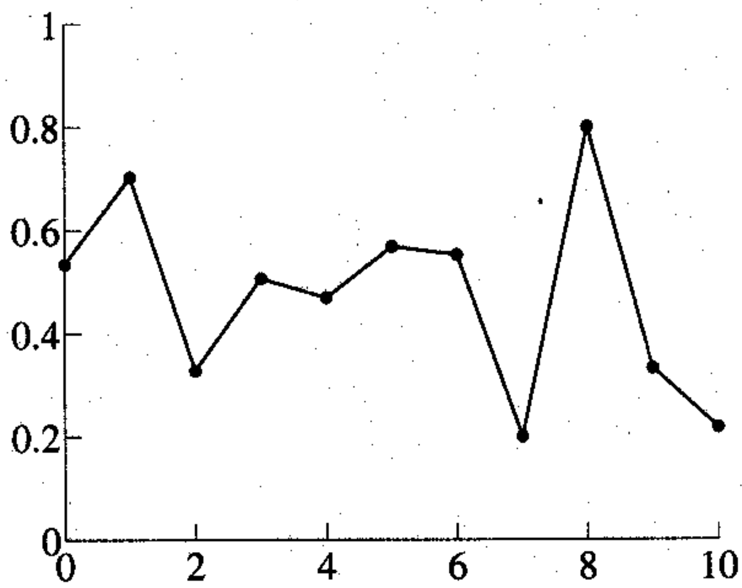
Chaikin’s scheme

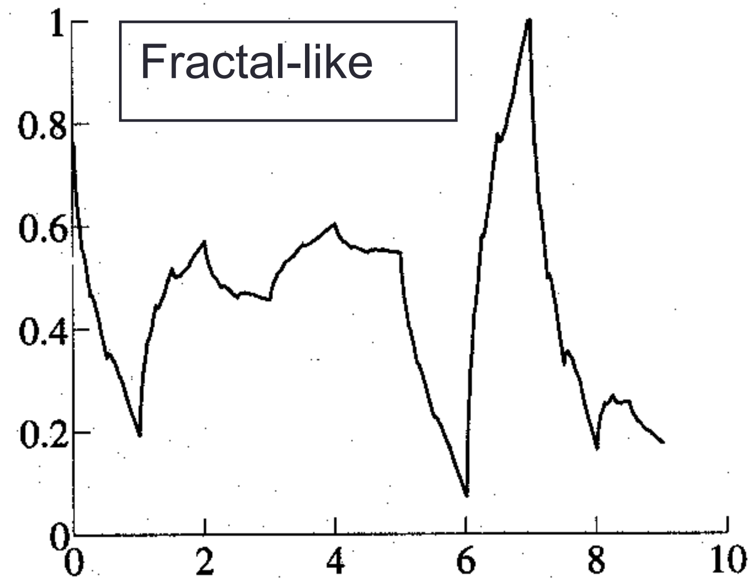


Examples

Daubechies scheme

$$(r_0, r_1) = \frac{1}{2}(1+\sqrt{3}, 1-\sqrt{3})$$



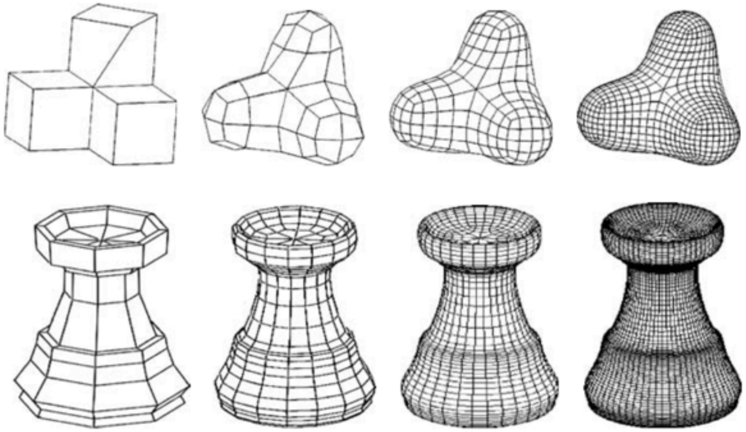


Subdivision Surfaces

Apply the same ideas to generating smooth surfaces.

General approach:

1. Start with a control **Polytope.**
2. At each iteration refine the polytope according to some rules.
3. Stop when resolution is high enough.

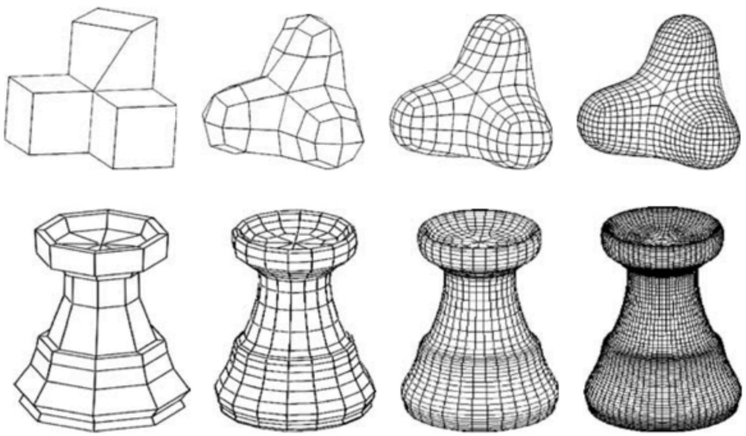


Subdivision Surfaces

Apply the same ideas to generating smooth surfaces.

General approach:

1. Start with a control **Polytope.**
2. At each iteration refine the polytope according to some rules.
3. Stop when resolution is high enough.



Subdivision Rules

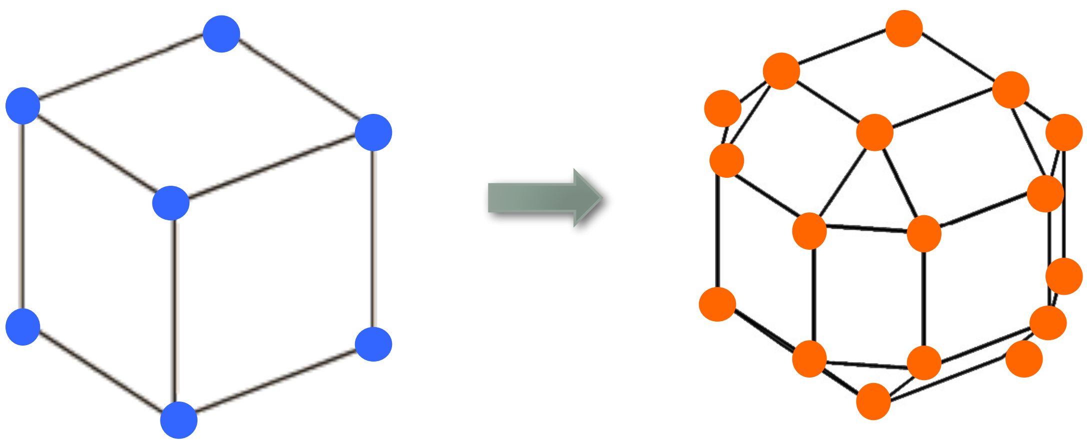
There are **topological** and **geometric** changes.

Geometric:

- How the positions of the vertices change

Topological:

- How the connectivity changes

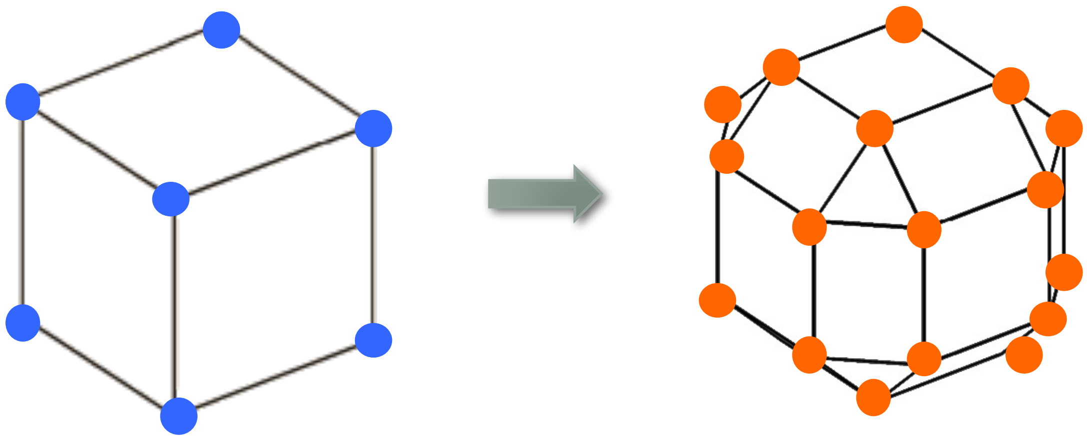


Subdivision Rules

There are **topological** and **geometric** changes.

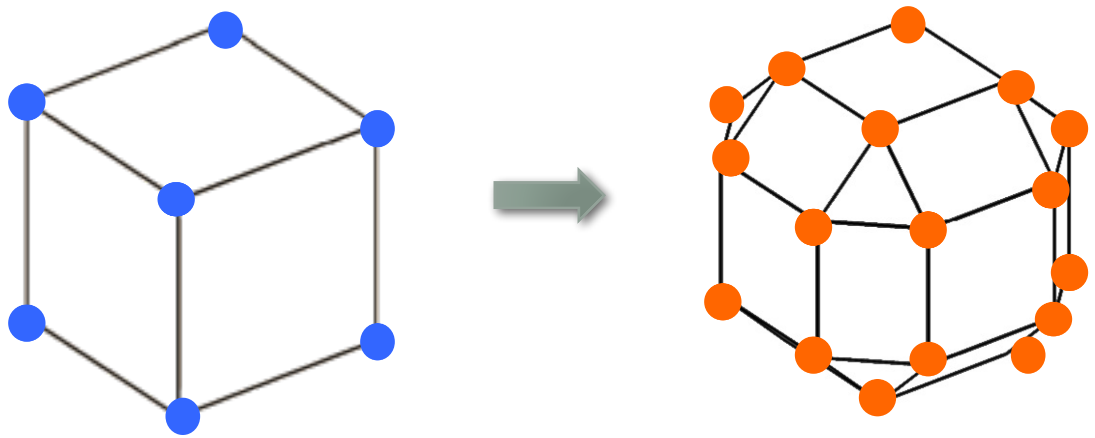
Typically, both geometric and topological changes are local:

New vertices, edges and faces depend on a small neighborhood of old ones.



Doo-Sabin subdivision surfaces

Generalization of Chaikin's corner cutting to surfaces.

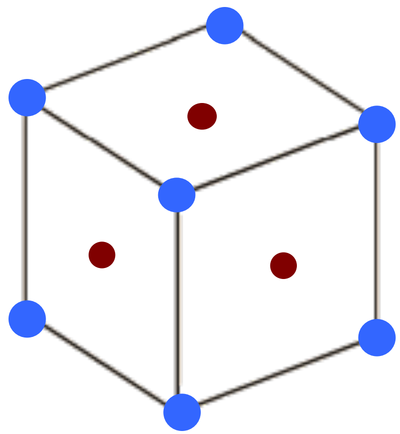


At each iteration:

1. Consider the barycenter of every (old) face
2. Construct centroids between the center and old vertices.
3. Connect them in a natural way.
4. Restart.

Doo-Sabin subdivision surfaces

Generalization of Chaikin’s corner cutting to surfaces.

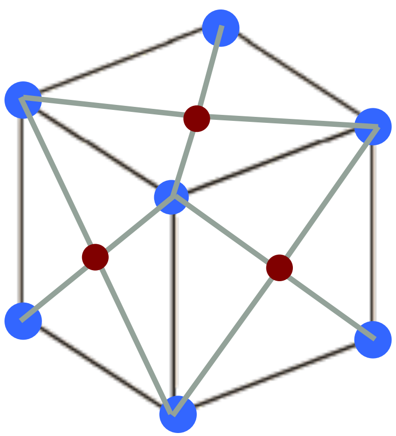


At each iteration:

1. Consider the barycenter of every (old) face
2. Construct centroids between the center and old vertices.
3. Connect them in a natural way.
4. Restart.

Doo-Sabin subdivision surfaces

Generalization of Chaikin's corner cutting to surfaces.

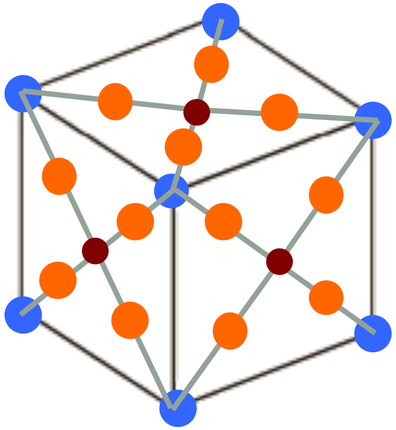


At each iteration:

1. Consider the barycenter of every (old) face
2. Construct centroids between the center and old vertices.
3. Connect them in a natural way.
4. Restart.

Doo-Sabin subdivision surfaces

Generalization of Chaikin's corner cutting to surfaces.

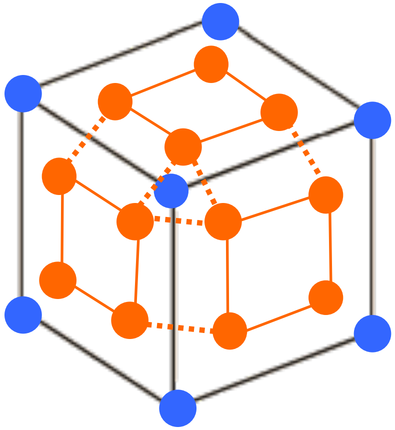


At each iteration:

1. Consider the barycenter of every (old) face
2. Construct centroids between the center and old vertices.
3. Connect them in a natural way.
4. Restart.

Doo-Sabin subdivision surfaces

Generalization of Chaikin's corner cutting to surfaces.

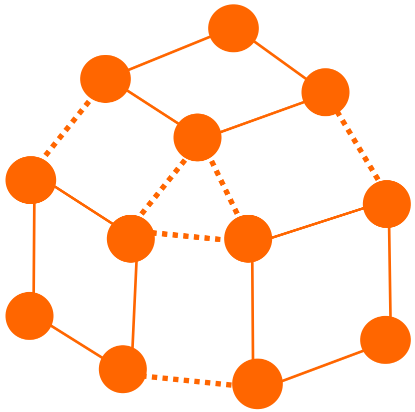


At each iteration:

1. Consider the barycenter of every (old) face
2. Construct centroids between the center and old vertices.
3. Connect them in a natural way.
4. Restart.

Doo-Sabin subdivision surfaces

Generalization of Chaikin’s corner cutting to surfaces.

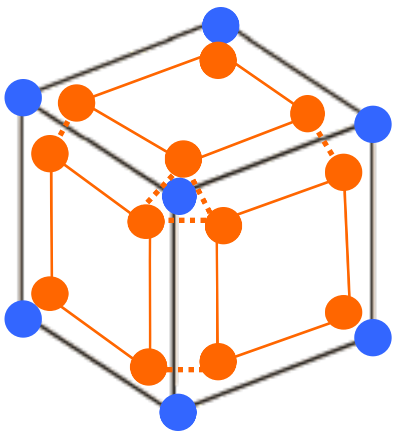


At each iteration:

1. Consider the barycenter of every (old) face
2. Construct centroids between the center and old vertices.
3. Connect them in a natural way.
4. Restart.

Doo-Sabin subdivision surfaces

Generalization of Chaikin’s corner cutting to surfaces.

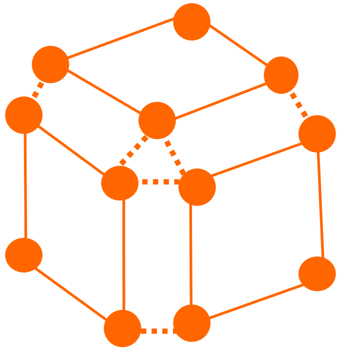


At each iteration:

1. Consider the barycenter of every (old) face
2. Construct ~~centroids~~ between the center and old vertices.
3. Connect them in a natural way.
4. Restart.

Doo-Sabin subdivision surfaces

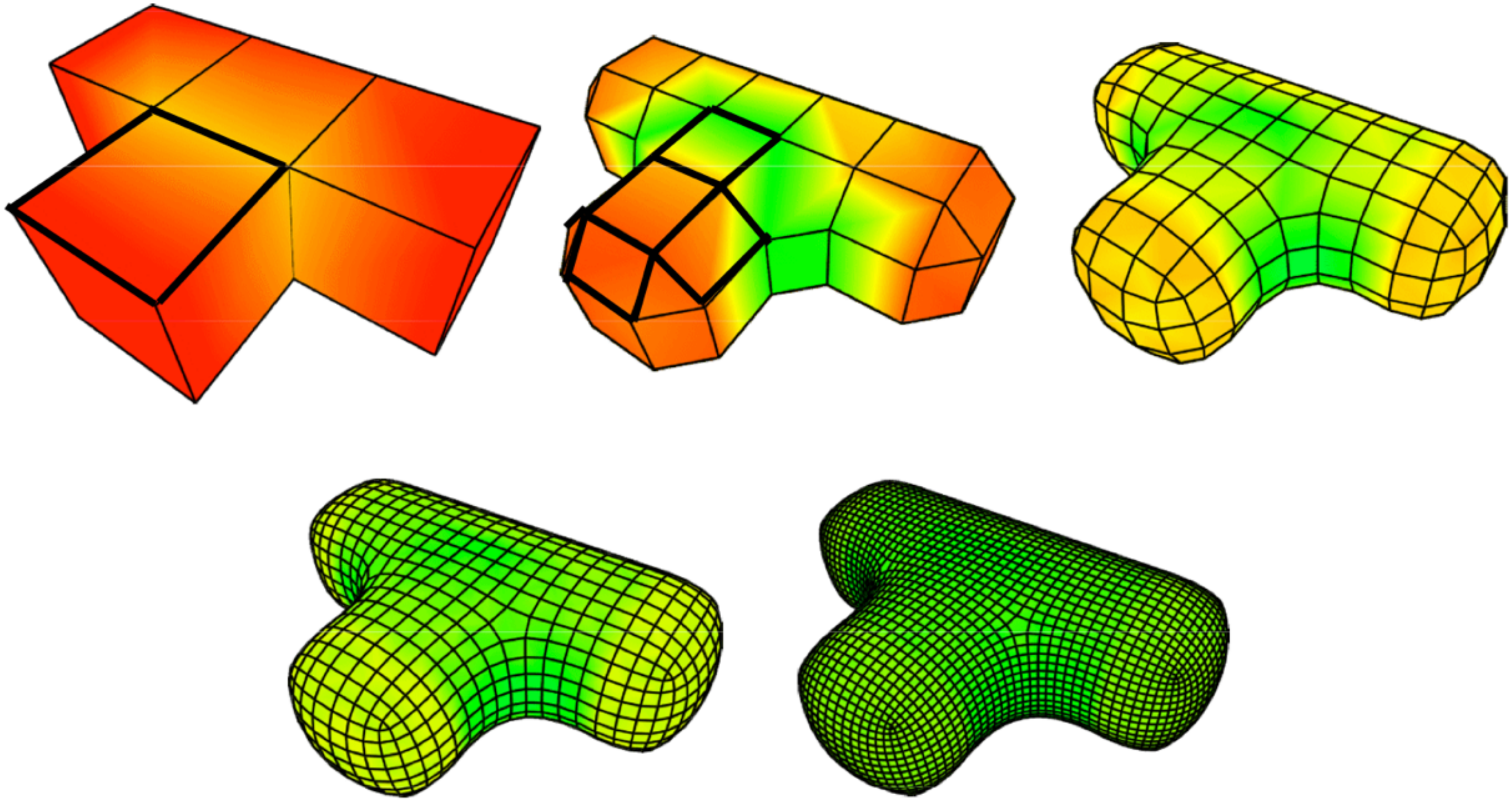
Generalization of Chaikin's corner cutting to surfaces.



At each iteration:

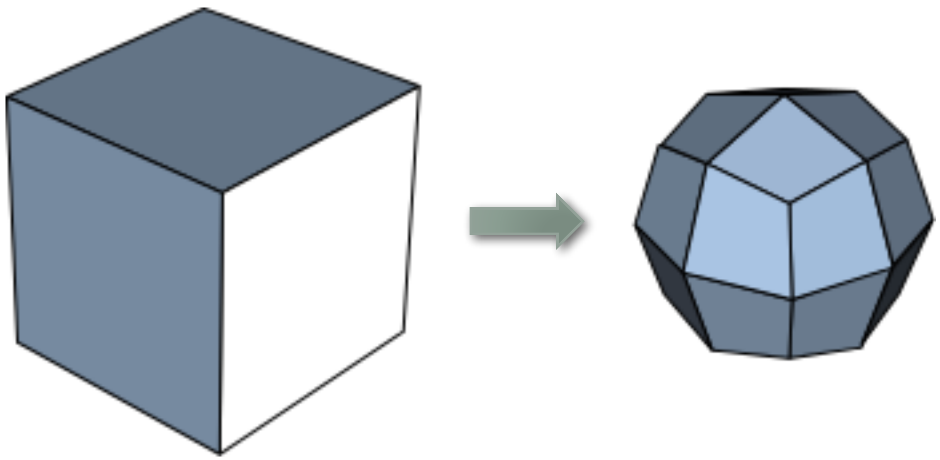
1. Consider the barycenter of every (old) face
2. Construct ~~centroids~~ between the center and old vertices.
3. Connect them in a natural way.
4. Restart.

Doo-Sabin subdivision surfaces



Catmull-Clark subdivision surfaces

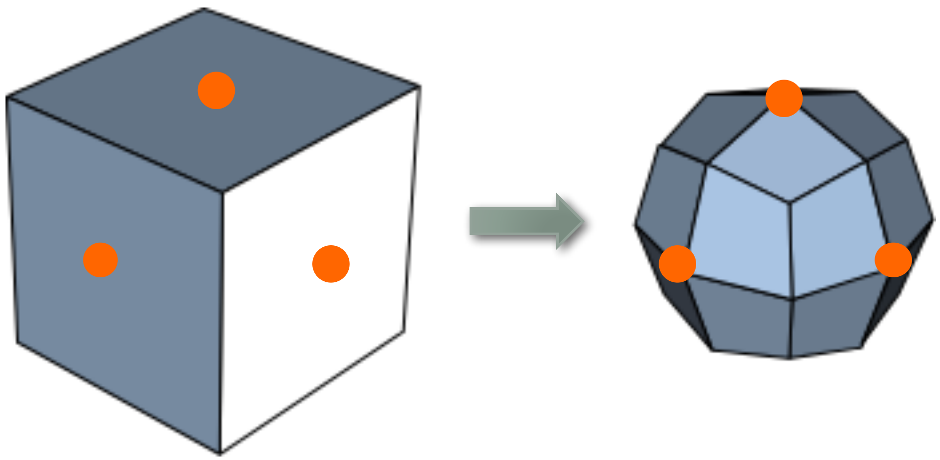
Generalization of cubic spline subdivision to surfaces.



- Approximating Scheme
- Small support stencil (just immediate neighbors)
- Limit surface is 2nd-order continuous except at extraordinary vertices
- Subdivision scheme used in all modern Pixar films

Catmull-Clark subdivision surfaces

Generalization of cubic spline subdivision to surfaces.

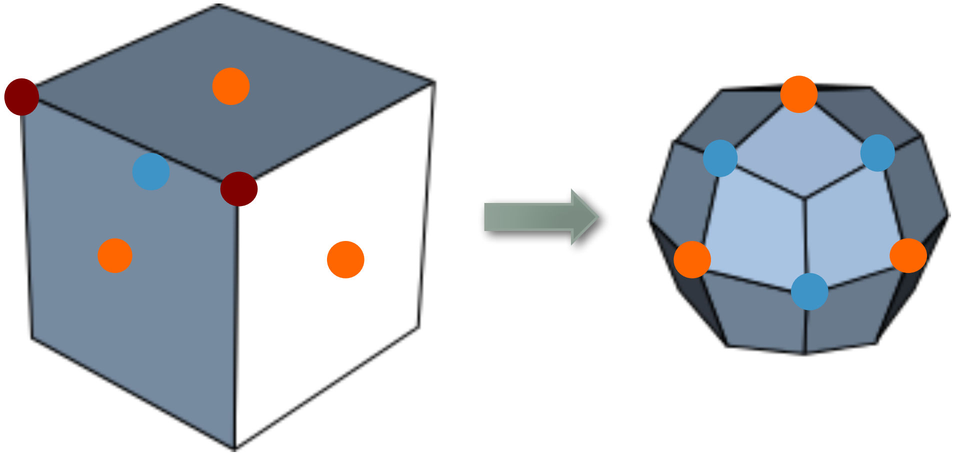


At each iteration:

1. Construct Face vertices: barycenters of old faces.
2. Construct Edge vertices.
3. Update existing vertices.
4. Connect them in a natural way.
4. Restart.

Catmull-Clark subdivision surfaces

Generalization of cubic spline subdivision to surfaces.

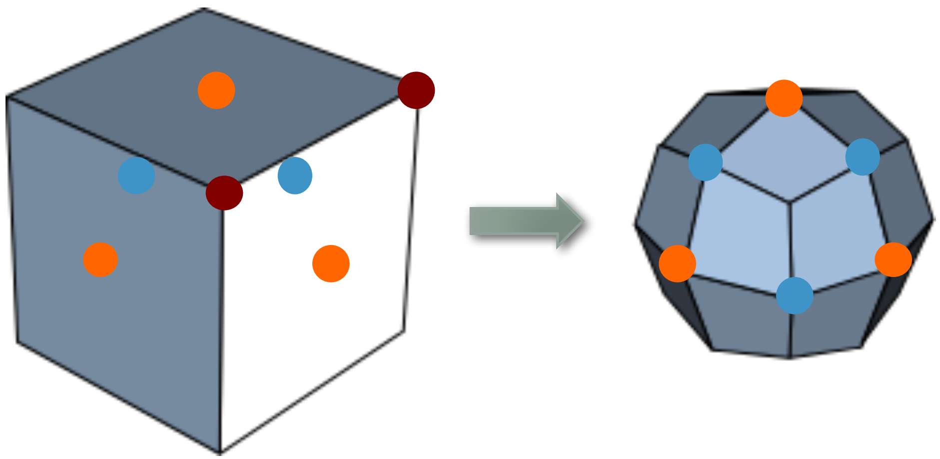


At each iteration:

1. Construct Face vertices: barycenters of old faces.
2. Construct Edge vertices: average of the old edge vertices and the associated face vertices

Catmull-Clark subdivision surfaces

Generalization of cubic spline subdivision to surfaces.

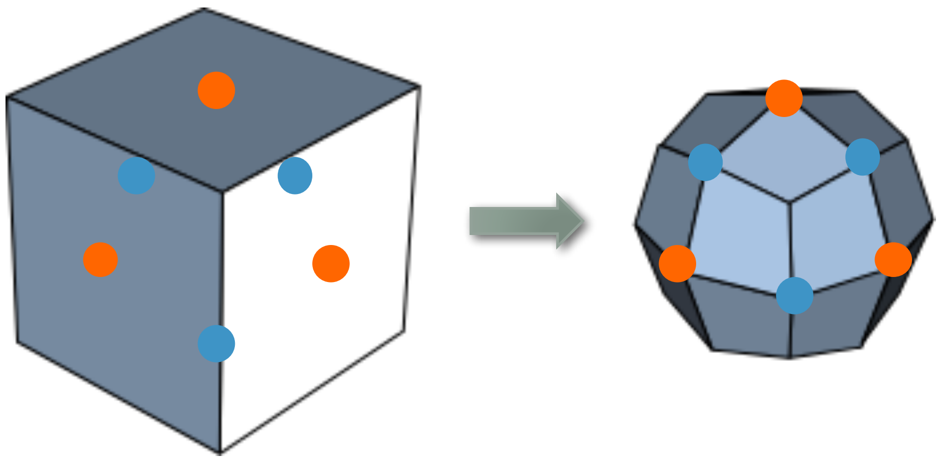


At each iteration:

1. Construct Face vertices: barycenters of old faces.
2. Construct Edge vertices: average of the old edge vertices and the associated face vertices

Catmull-Clark subdivision surfaces

Generalization of cubic spline subdivision to surfaces.

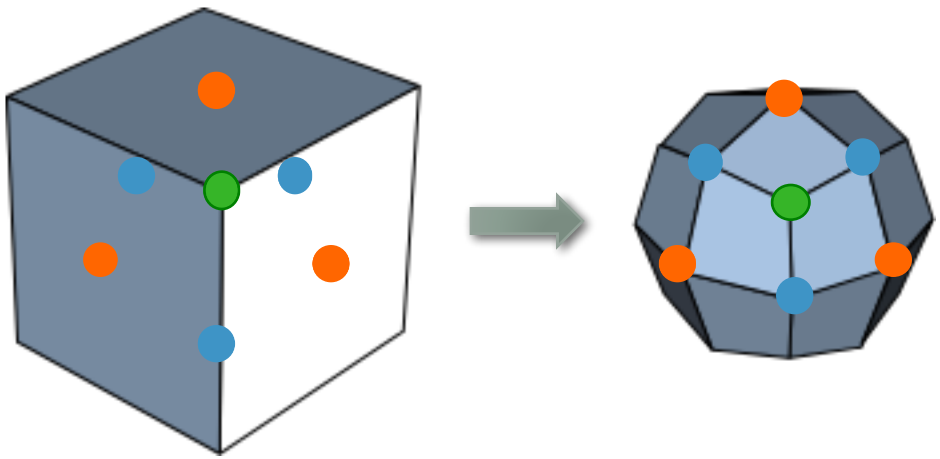


At each iteration:

1. Construct Face vertices: barycenters of old faces.
2. Construct Edge vertices: average of the old edge vertices and the associated face vertices

Catmull-Clark subdivision surfaces

Generalization of cubic spline subdivision to surfaces.

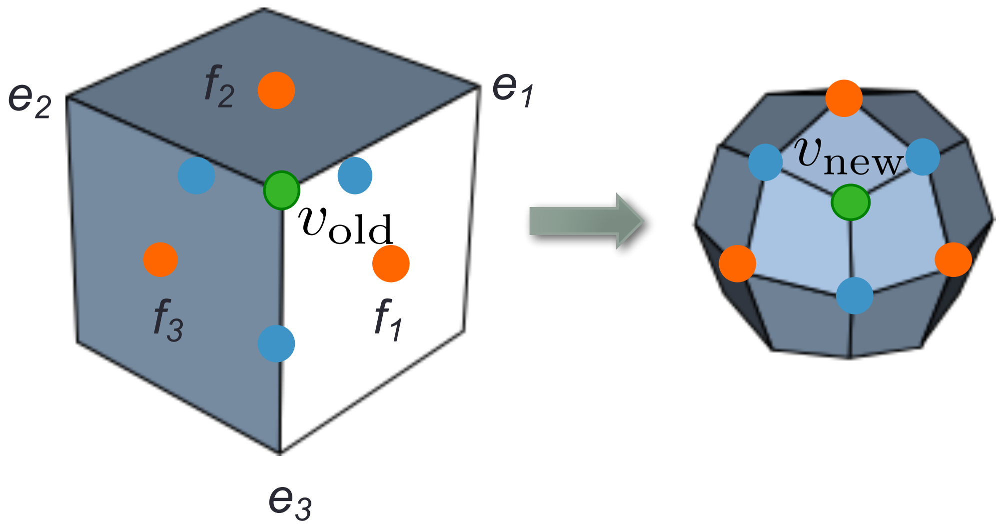


At each iteration:

1. Construct Face vertices: barycenters of old faces.
2. Construct Edge vertices.
3. Update existing vertices.
4. Connect them in a natural way.
4. Restart.

Catmull-Clark subdivision surfaces

Generalization of cubic spline subdivision to surfaces.



At each iteration:

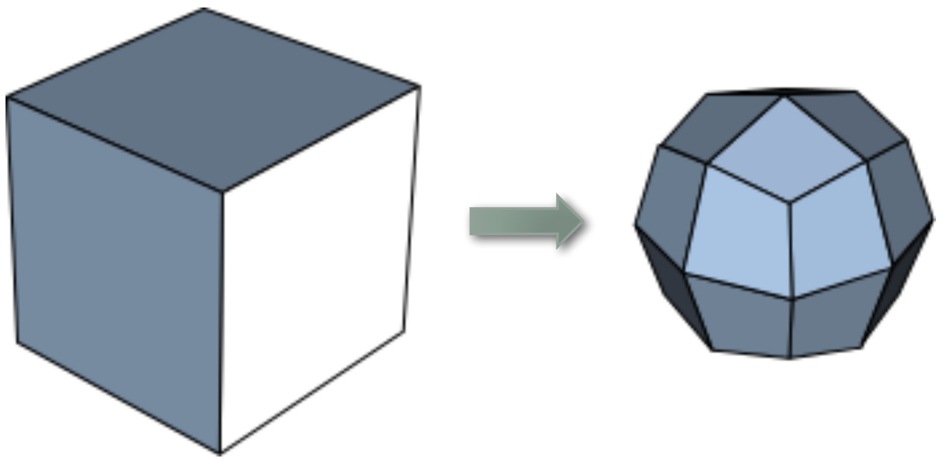
1. Construct Face vertices: barycenters of old faces.
2. Construct Edge vertices.
3. Update existing vertices.

$$v_{\text{new}} = v_{\text{old}} + \frac{1}{n^2}\sum_{j=1}^{n}(e_j - v_{\text{old}}) + \frac{1}{n^2}\sum_{j=1}^{n}(f_j - v_{\text{old}})$$

e_j: **old** vertex incident along edge j
f_i: **new** (orange) vertex on face j
n: number of incident edges.

Catmull-Clark subdivision surfaces

Generalization of cubic spline subdivision to surfaces.

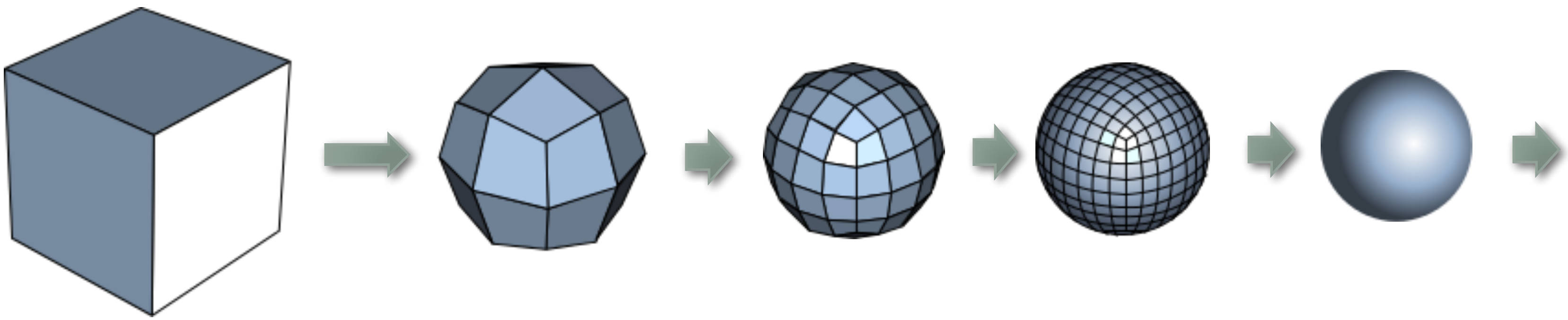


At each iteration:

1. Construct Face vertices.
2. Construct Edge vertices.
3. Update existing vertices.
4. Connect them in a natural way.
4. Restart.

Catmull-Clark subdivision surfaces

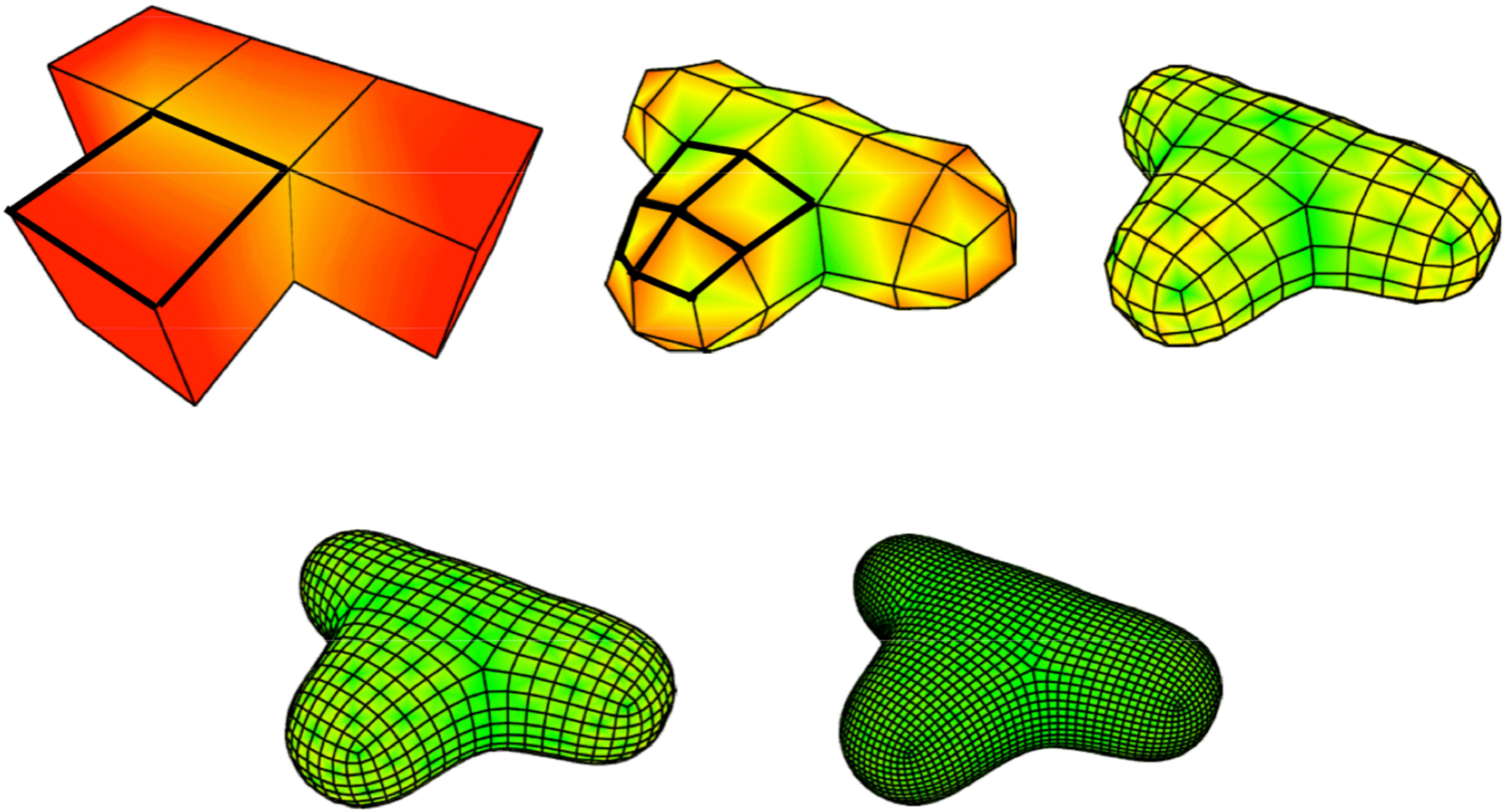
Generalization of cubic spline subdivision to surfaces.



At each iteration:

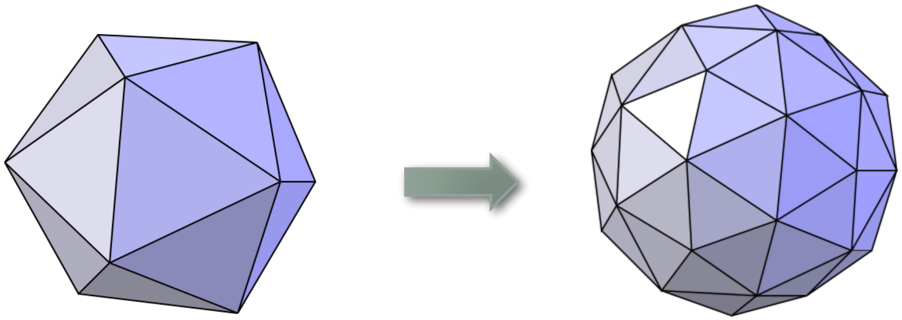
1. Construct Face vertices.
2. Construct Edge vertices.
3. Update existing vertices.
4. Connect them in a natural way.
4. Restart.

Catmull-Clark subdivision surfaces



Loop subdivision surfaces

Triangle-based subdivision:

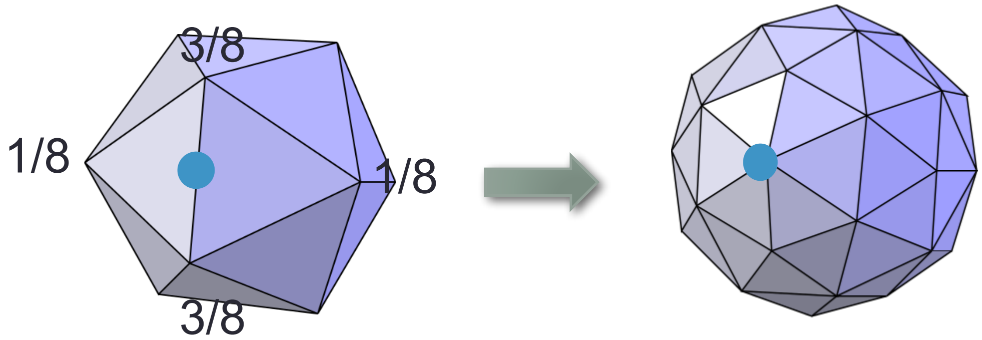


At each iteration:

1. Construct Edge vertices.
2. Update existing vertices.
3. Connect them in a natural way.
4. Restart.

Loop subdivision surfaces

Triangle-based subdivision:



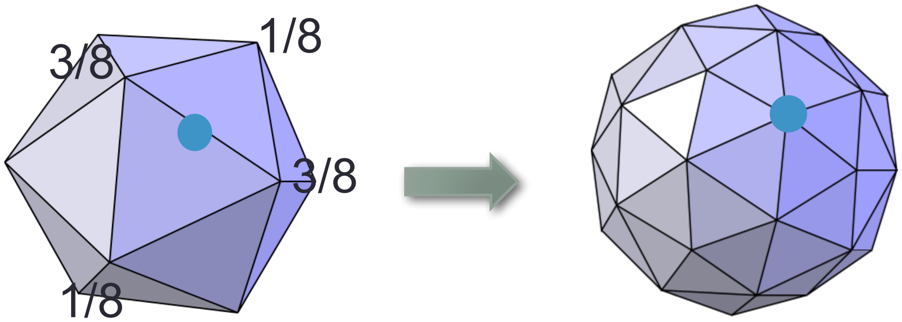
At each iteration:

1. Construct Edge vertices.

$$e_i = \frac{3}{8}v_{e1} + \frac{3}{8}v_{e2} + \frac{1}{8}v_{t1} + \frac{1}{8}v_{t2}$$

Loop subdivision surfaces

Triangle-based subdivision:



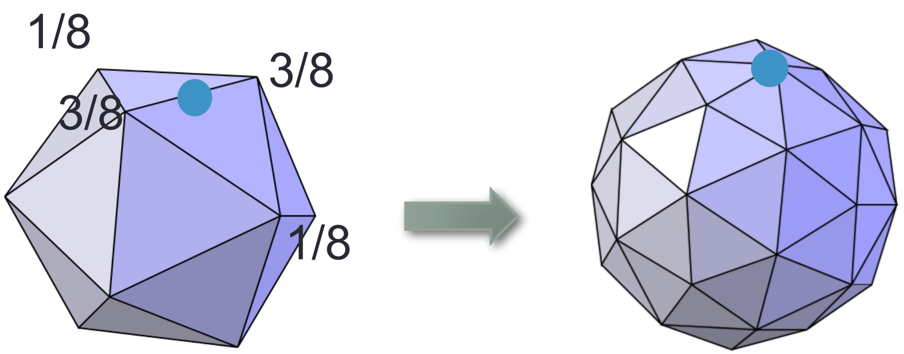
At each iteration:

1. Construct Edge vertices.

$$e_i = \frac{3}{8}v_{e1} + \frac{3}{8}v_{e2} + \frac{1}{8}v_{t1} + \frac{1}{8}v_{t2}$$

Loop subdivision surfaces

Triangle-based subdivision:



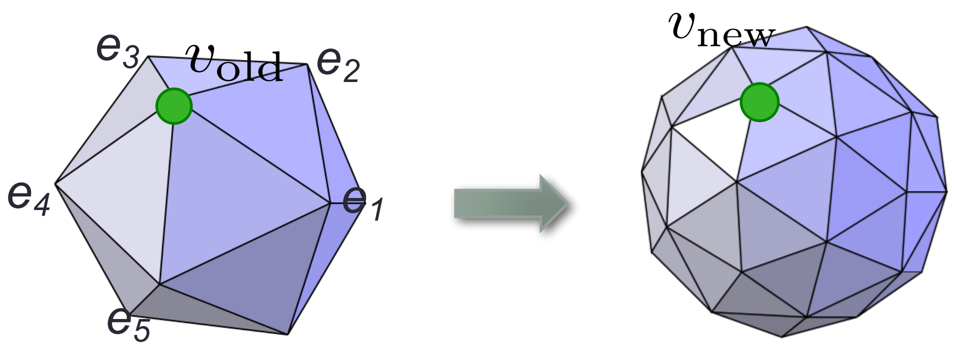
At each iteration:

1. Construct Edge vertices.

$$e_i = \frac{3}{8}v_{e1} + \frac{3}{8}v_{e2} + \frac{1}{8}v_{t1} + \frac{1}{8}v_{t2}$$

Loop subdivision surfaces

Triangle-based subdivision:



At each iteration:

1. Construct Edge vertices.
2. Update existing vertices:

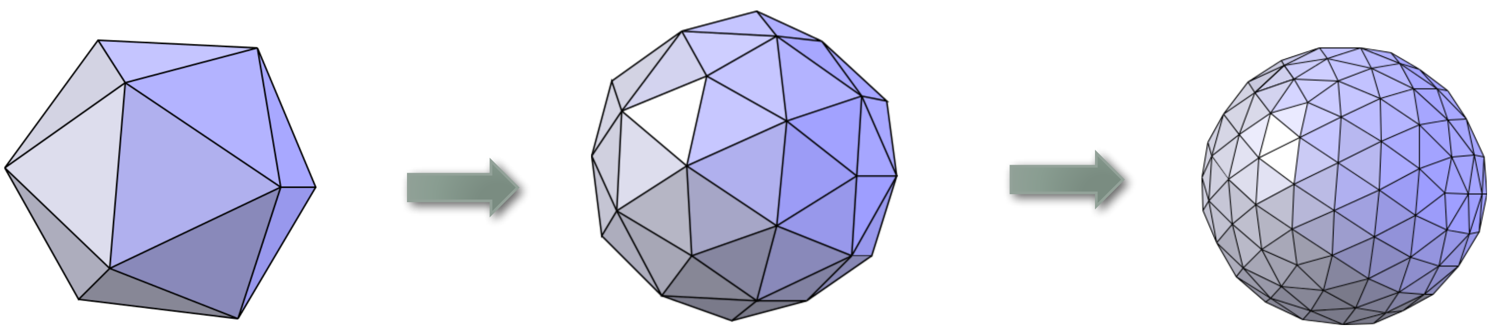
$$v_{\text{new}} = (1 - \alpha n) v_{\text{old}} + \alpha \sum_{j=1}^{n} e_j$$

$$\alpha = \begin{cases} \frac{3}{16} & \text{if } n = 3 \\ \frac{3}{8n} & \text{if } n > 3 \end{cases}$$

e_j: old vertex incident along edge j
n: number of incident edges.

Loop subdivision surfaces

Triangle-based subdivision:

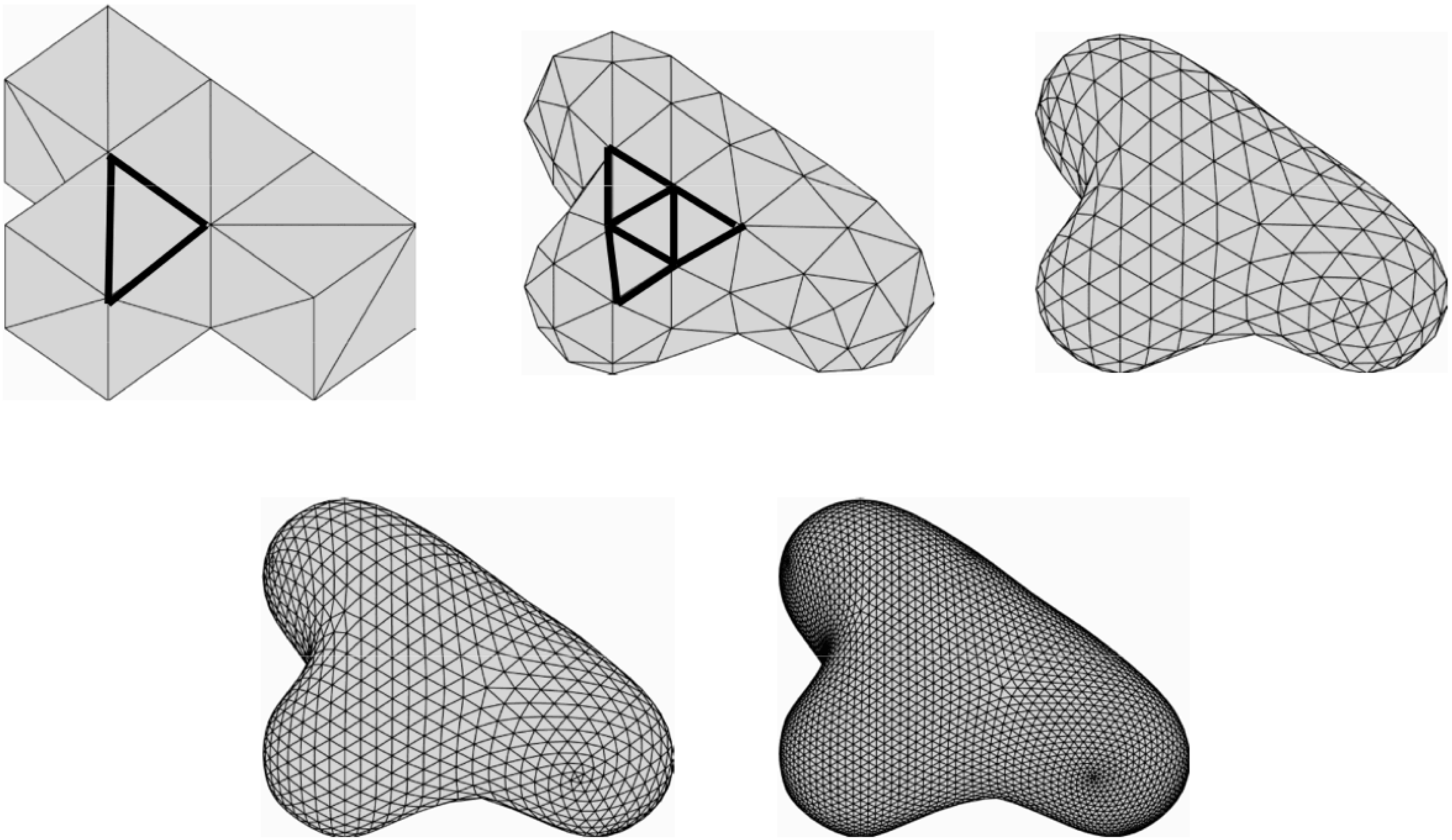


At each iteration:

1. Construct Edge vertices.
2. Update existing vertices.
3. Connect them in a natural way.
4. Restart.

Attention: different update rules on the boundary.

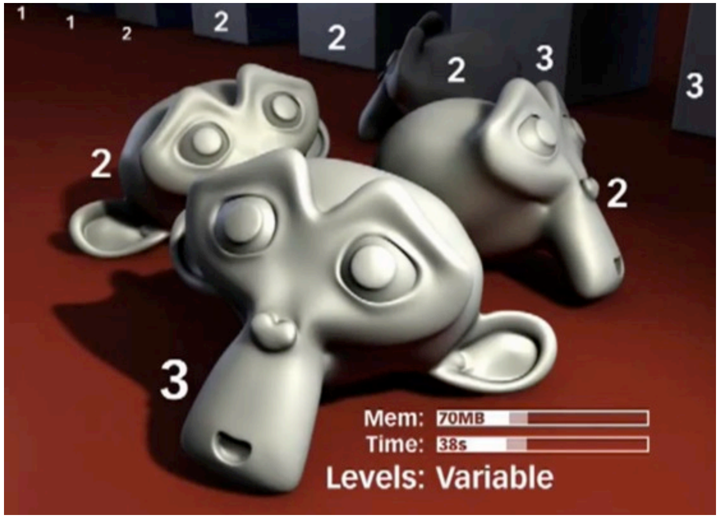
Loop subdivision surfaces



Conclusions

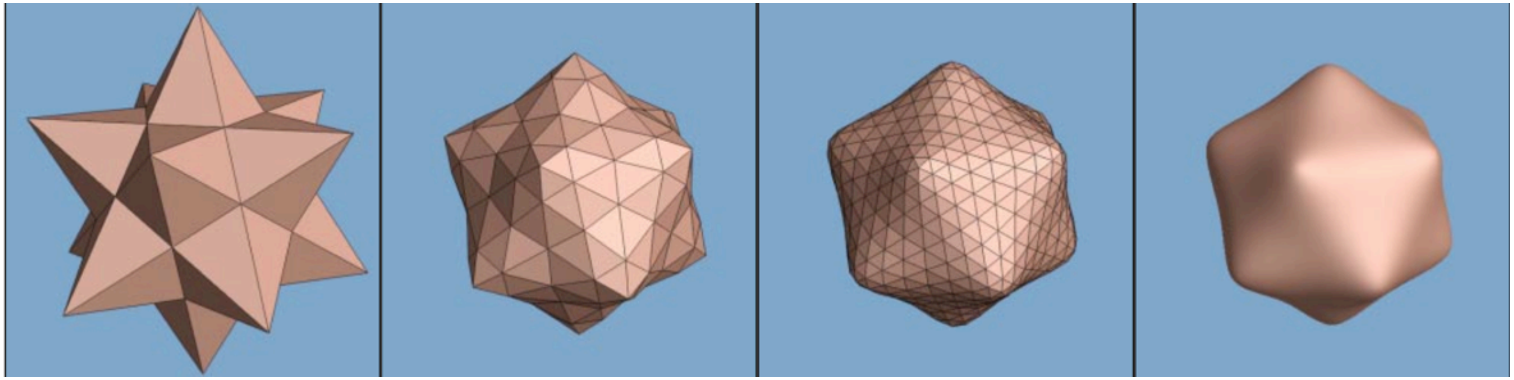
Subdivision surfaces:

- Allow simpler modeling
  - Major strength: surfaces of arbitrary topology
  - Limit surfaces are smooth
  - Control mesh is typically simple and intuitive
- Adapt to user's needs
  - Render only at required level-of-detail
- Usability
  - Compact representation
  - Simple and efficient code

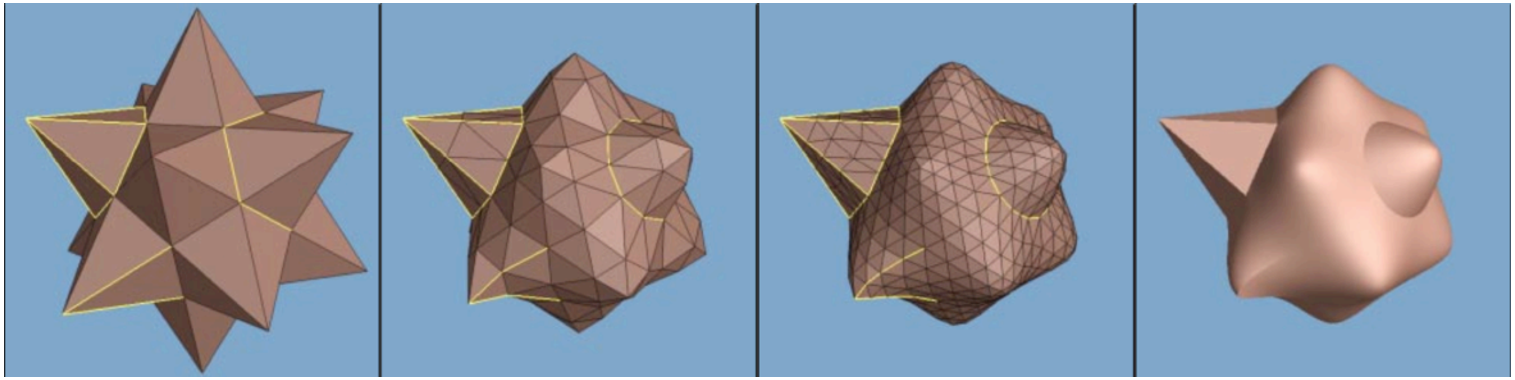


Extensions

Piecewise-smooth subdivision surfaces:



Allow some sharp edges to remain



Hoppe et al. *Piecewise Smooth Surface Reconstruction*, SIGGRAPH '94