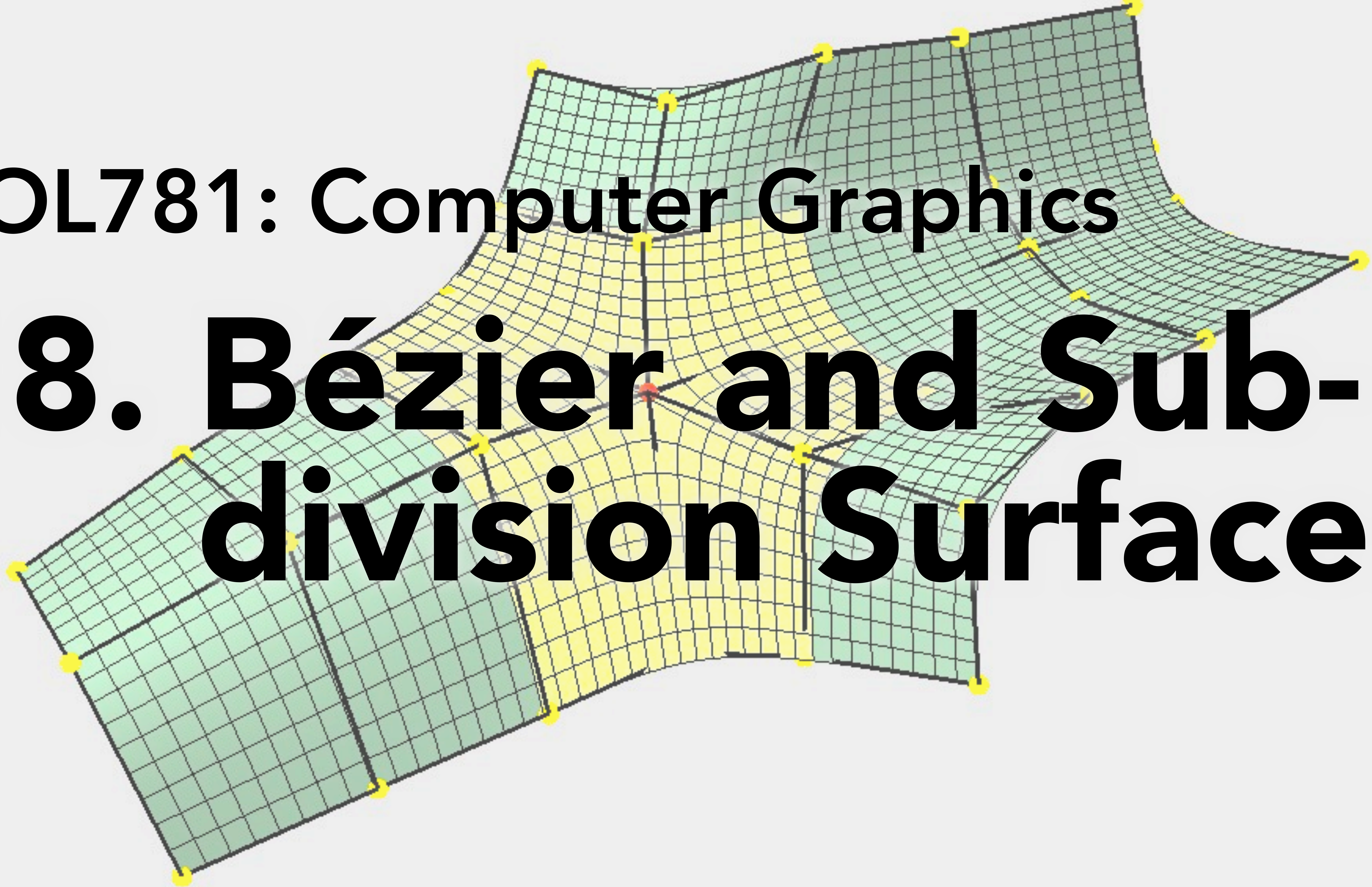
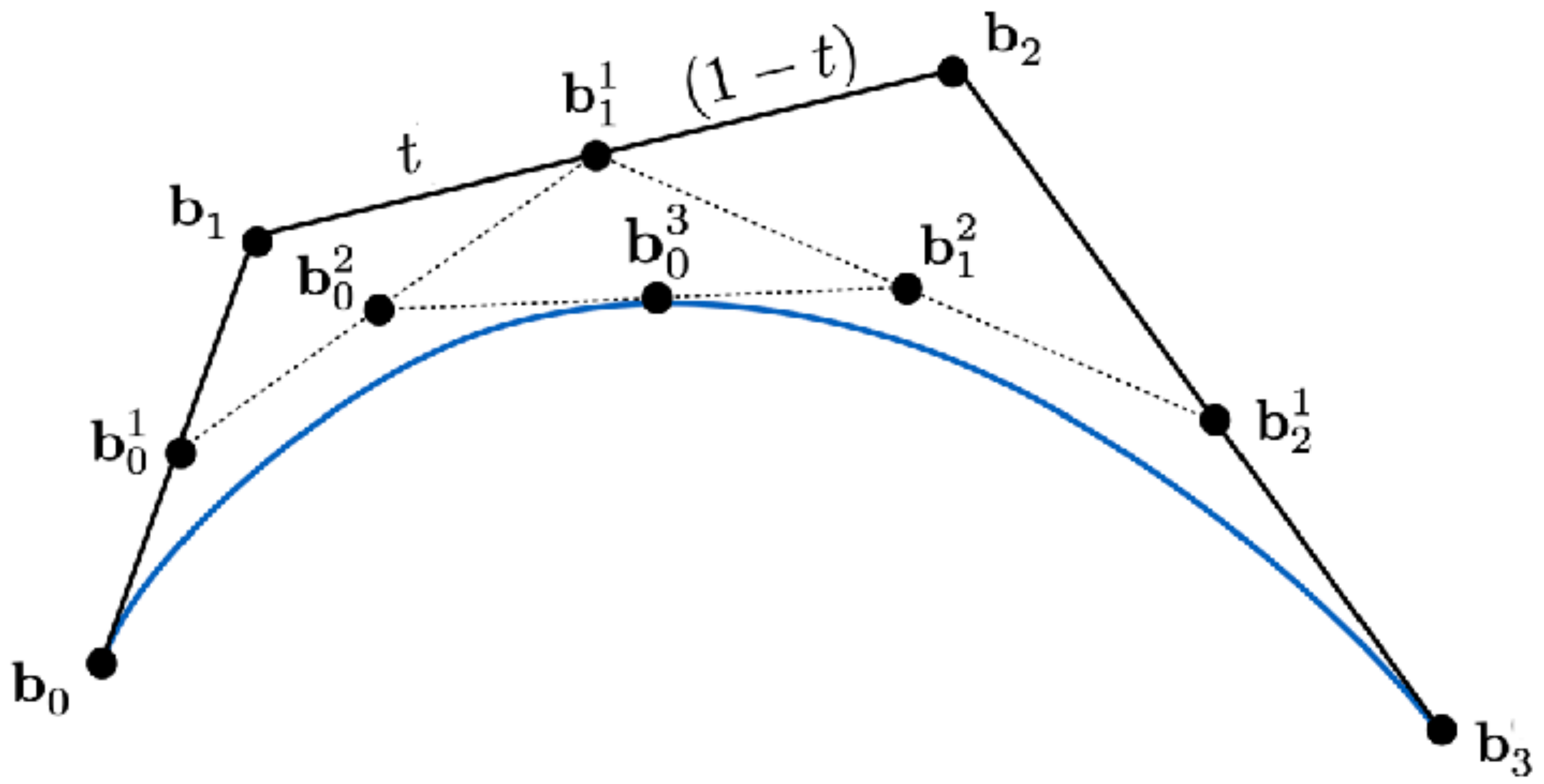
COL781: Computer Graphics

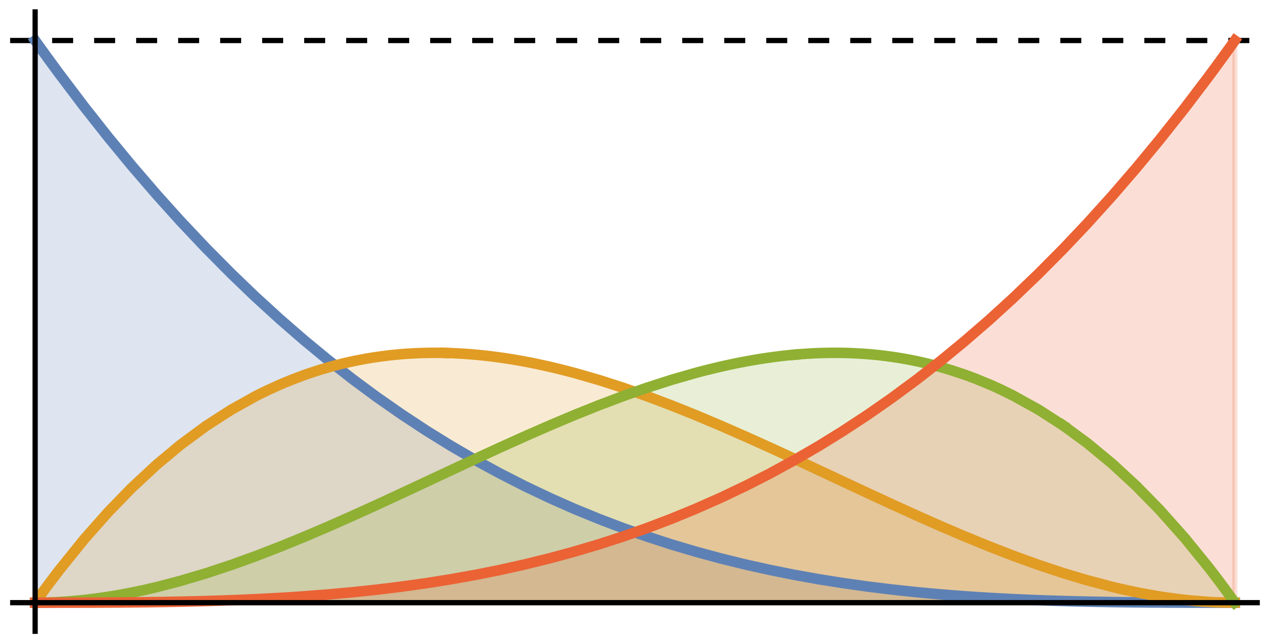
18. Bézier and Sub-division Surfaces

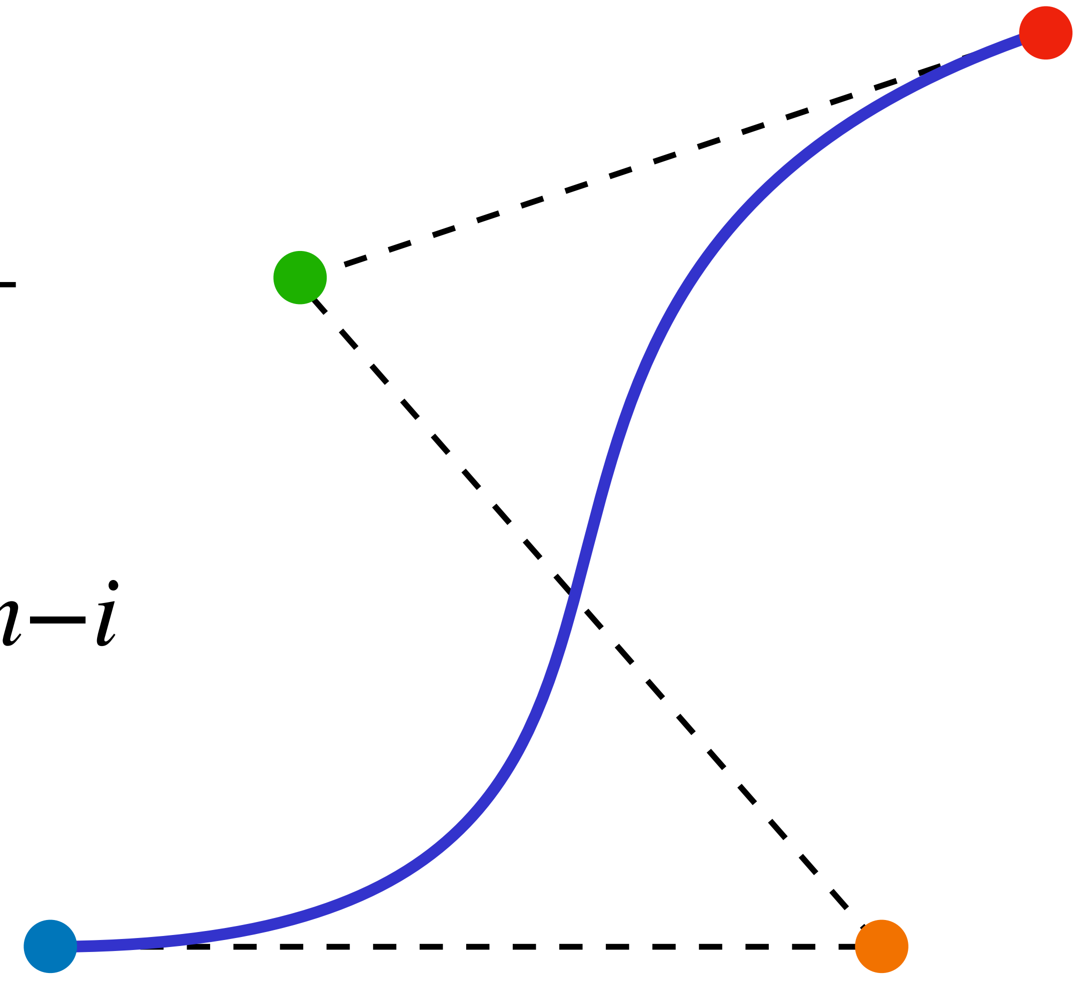


Pixar

Recap: Bézier curves







$$\mathbf{b}_0^1 = \mathrm{lerp}(t, \mathbf{b}_0, \mathbf{b}_1)$$

…

$$\mathbf{b}(t) = \mathrm{lerp}(t, \mathbf{b}_0^{n-1}, \mathbf{b}_1^{n-1})$$

Procedural form (De Casteljau's corner cutting algorithm)

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

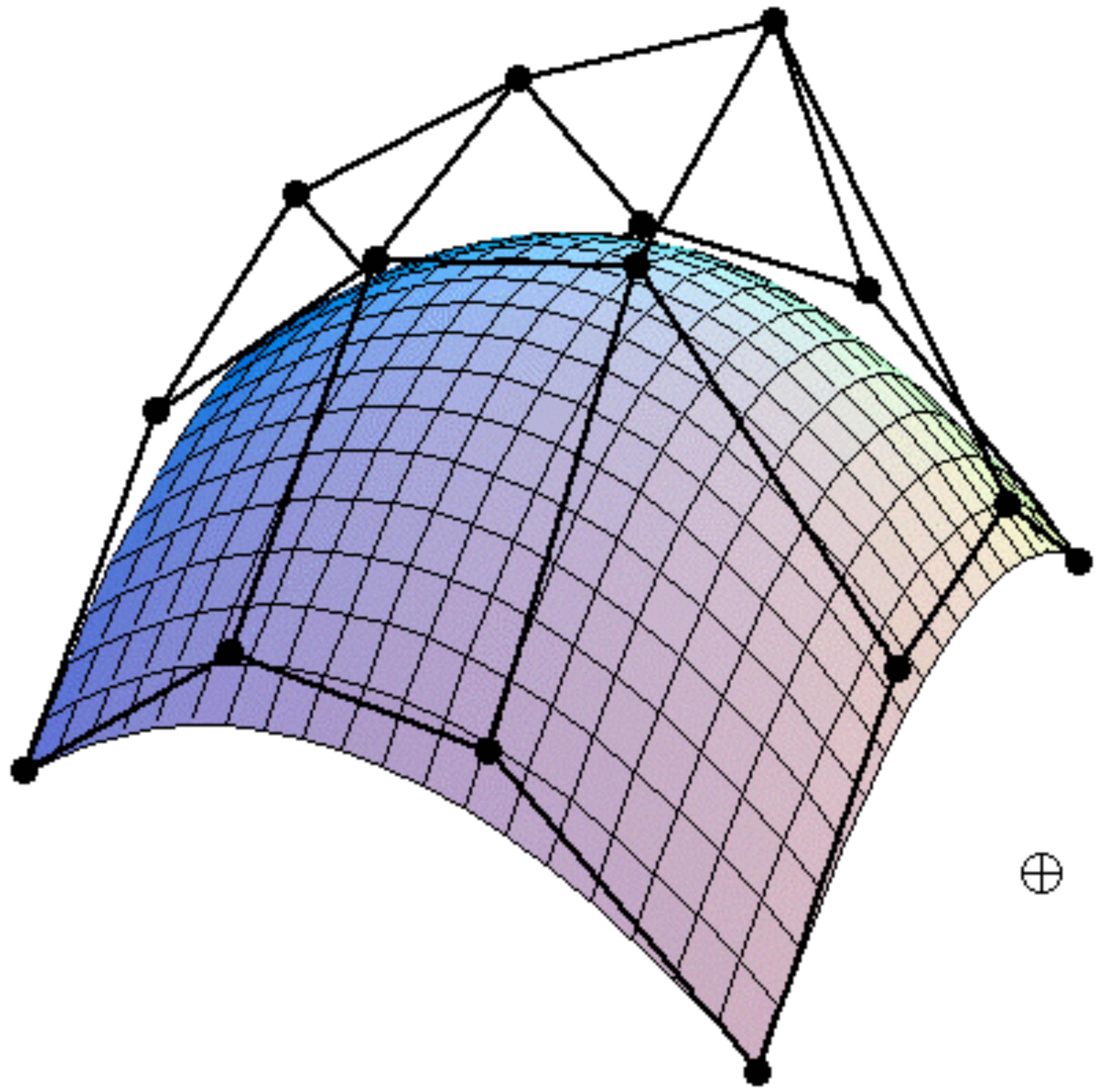
$$\mathbf{b}(t) = \sum_{i=0}^{n} B_i^n(t)\,\mathbf{b}_i$$

Analytical form (linear combination of Bernstein polynomials)

Bézier patches

Parametric surface **p**(*u*, *v*) made of Bézier curves

- Treat each row as a Bézier curve
- Evaluate at *u* to get one point per row
- Treat as control points of a Bézier curve
- Evaluate at *v* to get point **p**(*u*, *v*) on surface



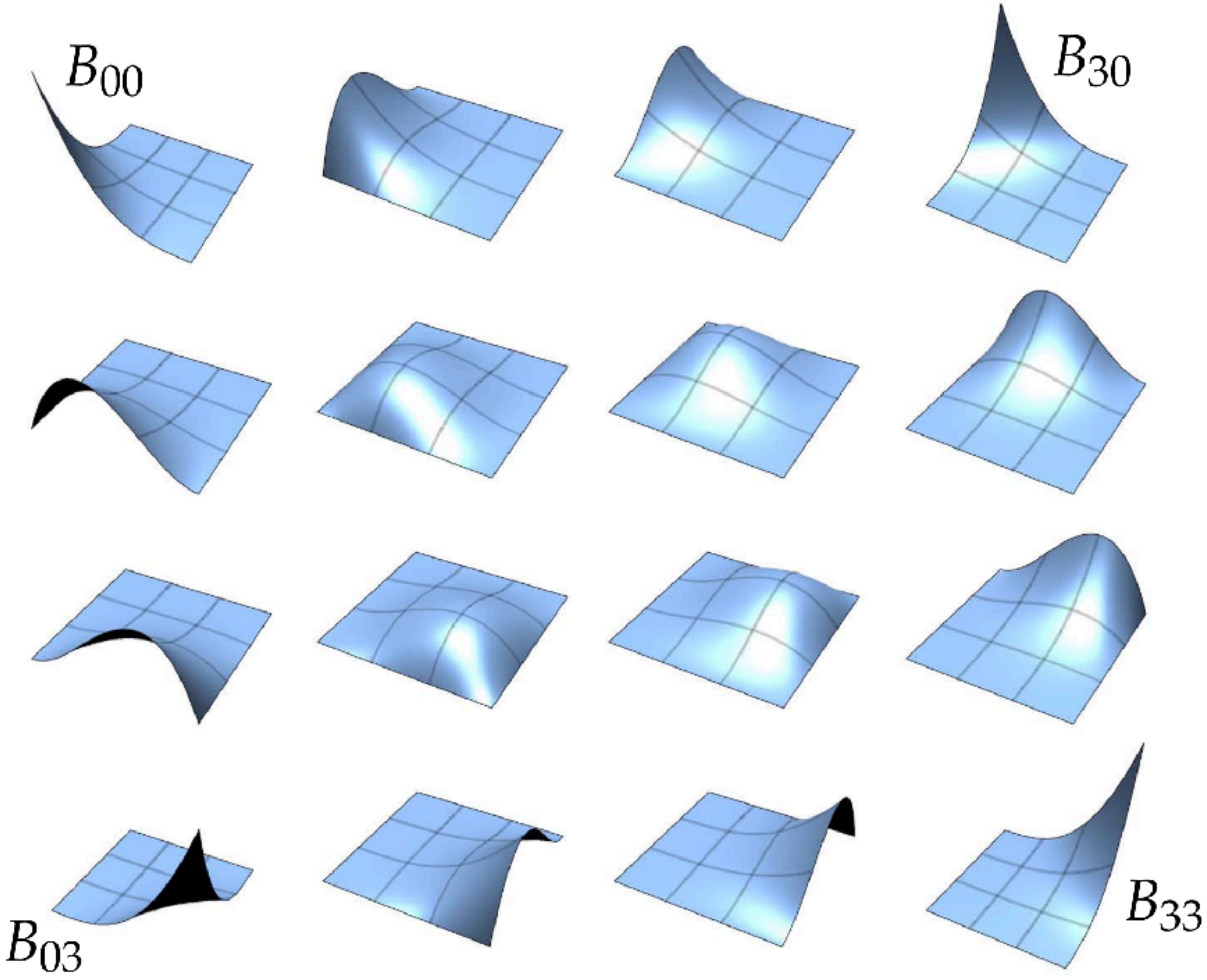
Algebraically:

$$\mathbf{p}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{n} B_i^3(u)\, B_j^3(v)\, \mathbf{p}_{ij}$$

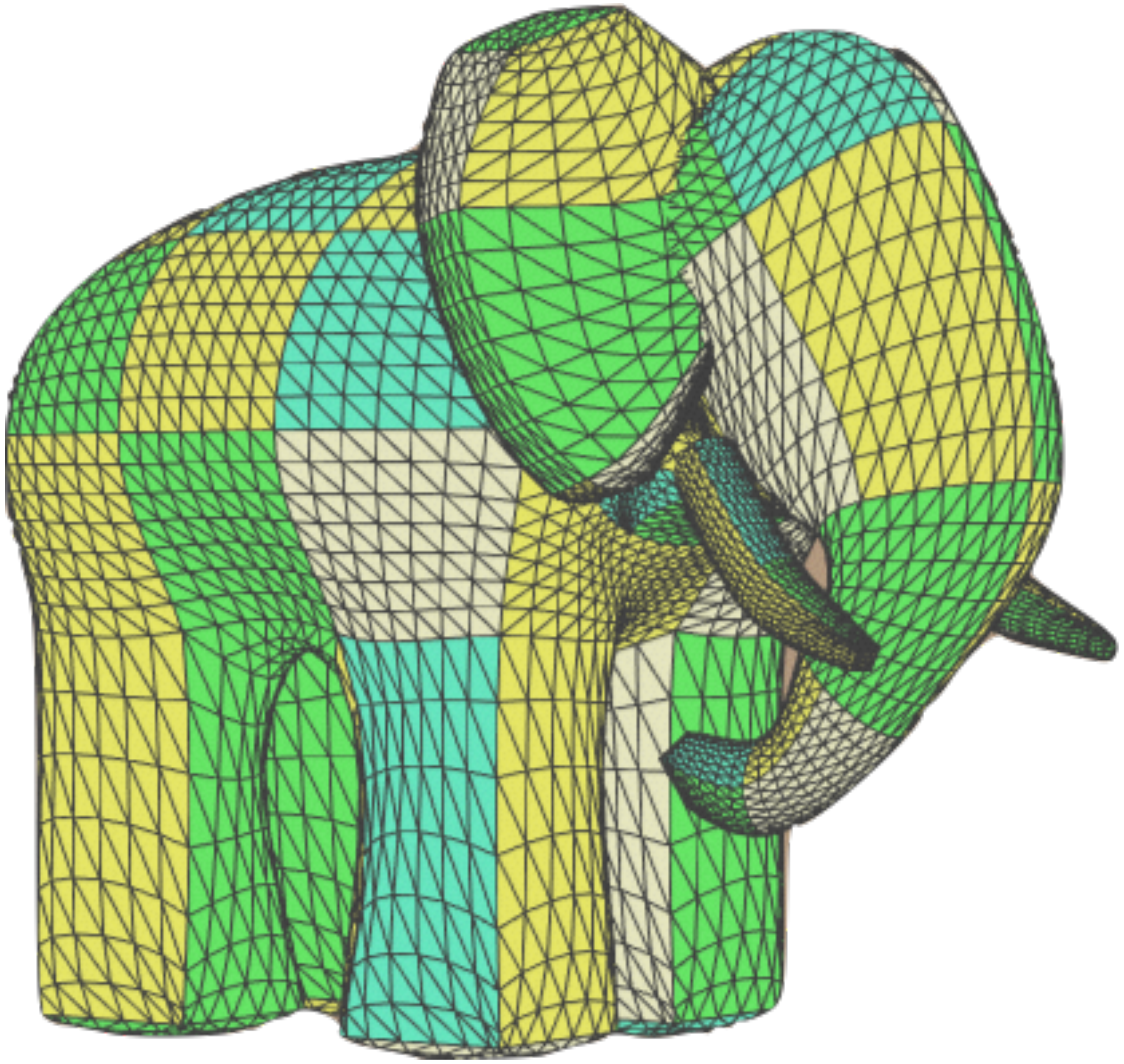
$$= \sum_{0 \le i,j \le n} B_{ij}(u,v)\, \mathbf{p}_{ij}$$

Basis functions B_{ij} are "**tensor products**" of Bernstein polynomials:

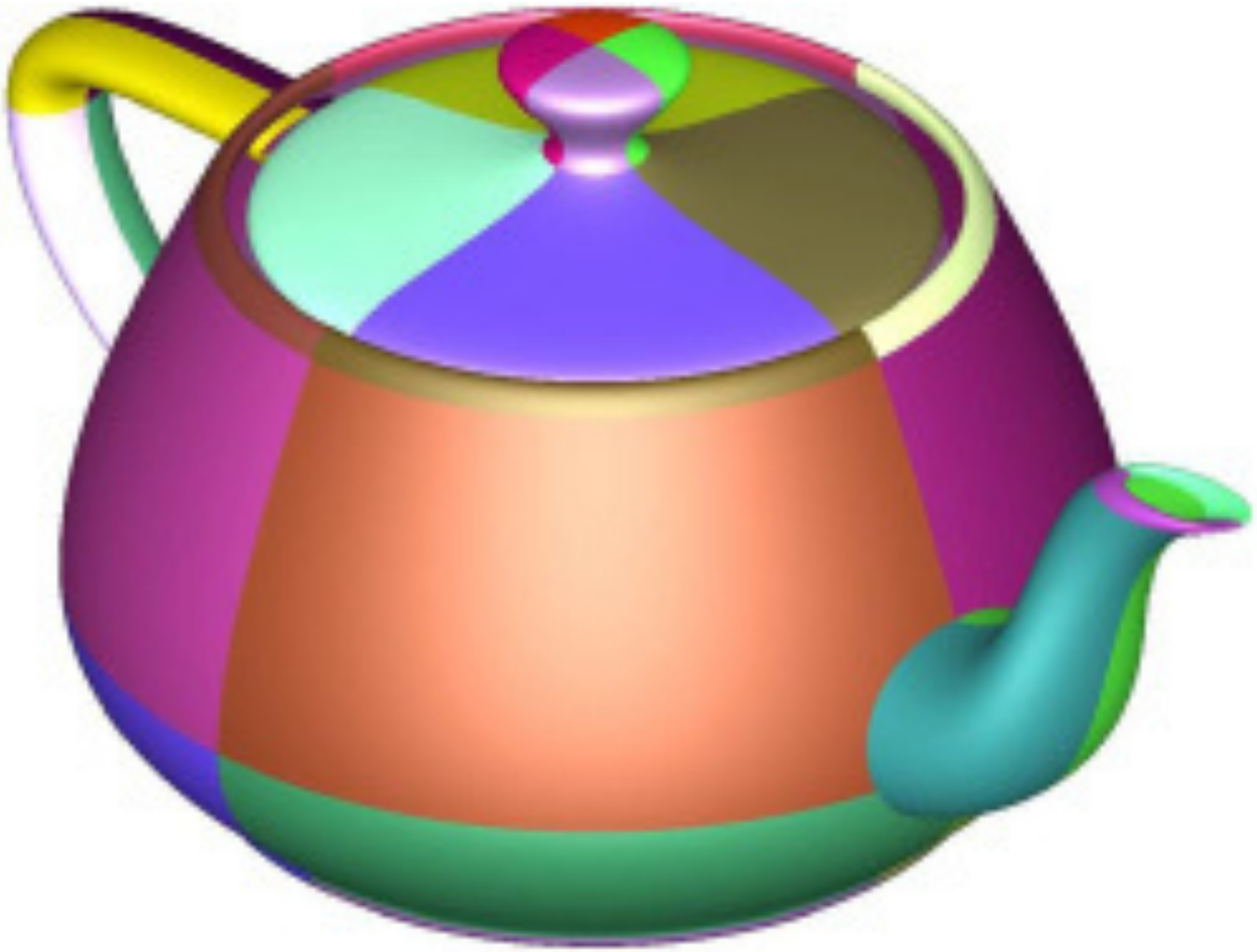
$$(f \otimes g)(x,y) = f(x)\, g(y)$$



Keenan Crane



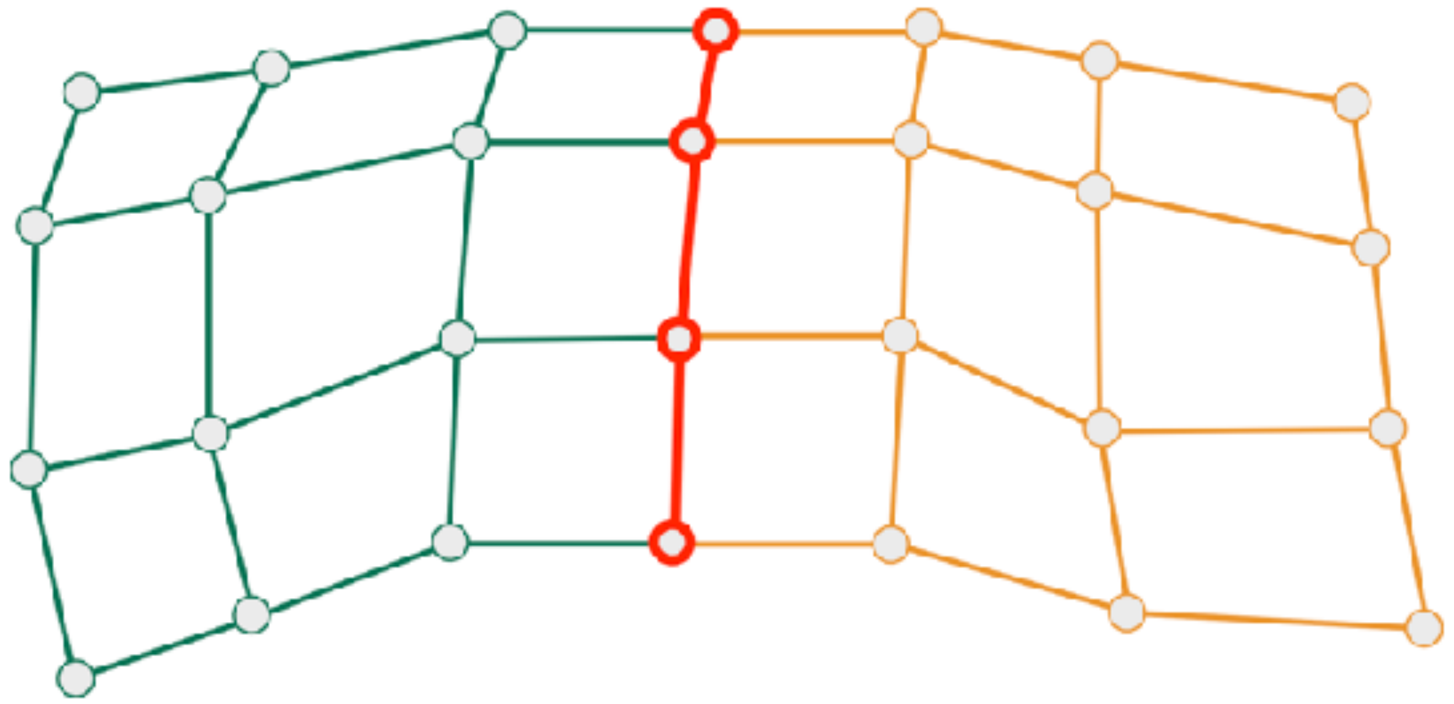
Ed Catmull's "Gumbo" model



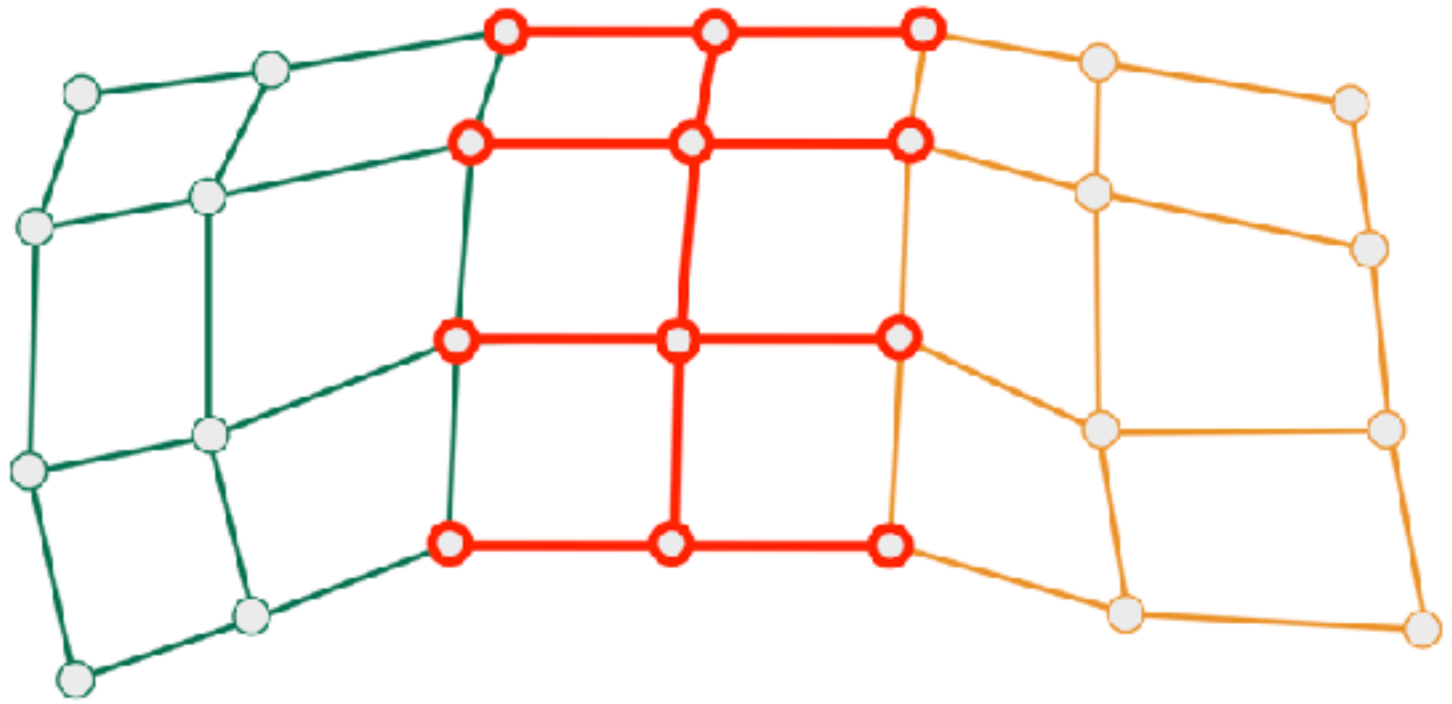
The Utah teapot,
modeled by Martin Newell

Continuity

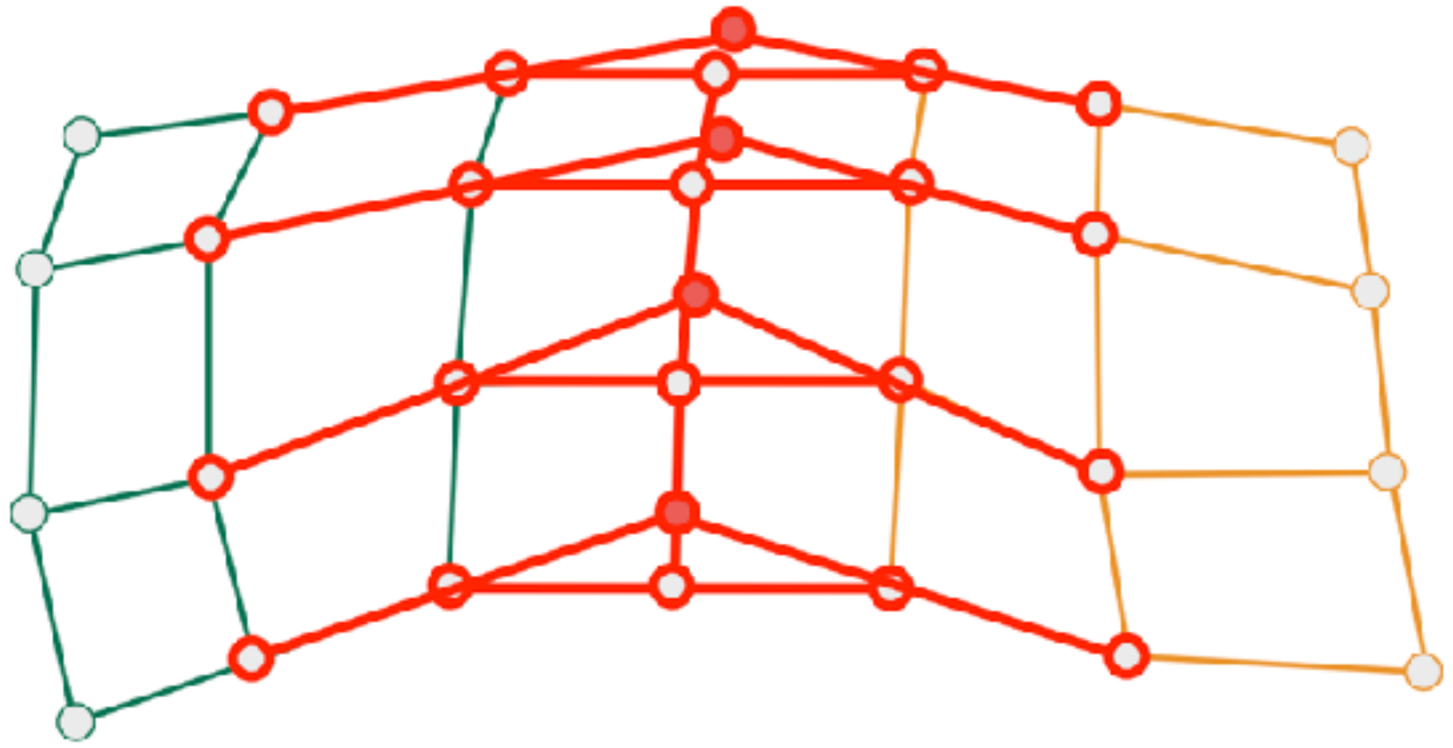
Continuity is now determined along each boundary edge between two patches.



C^0 continuity:
Boundary points agree

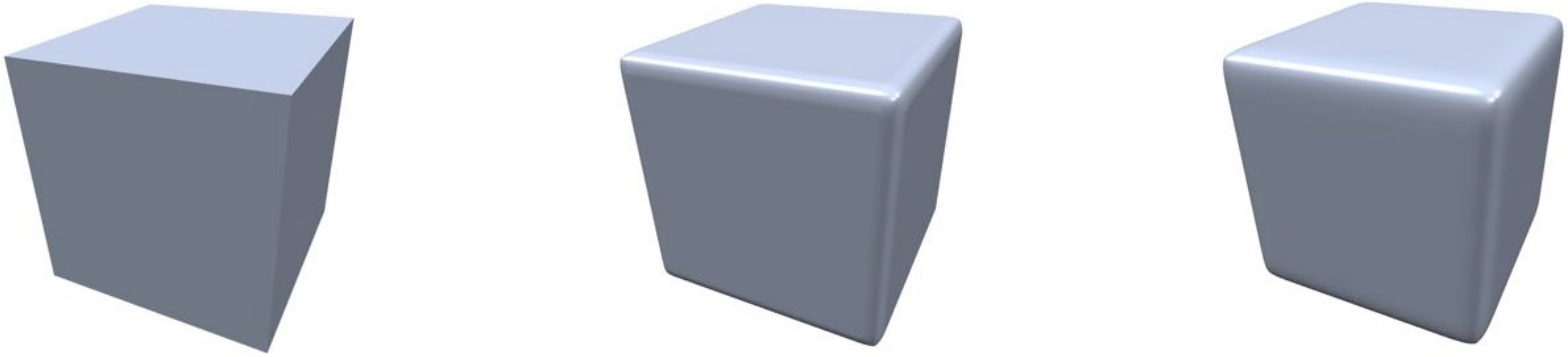


C^1 continuity:
Adjacent edges equal



C^2 continuity:
A-frames

Ren Ng



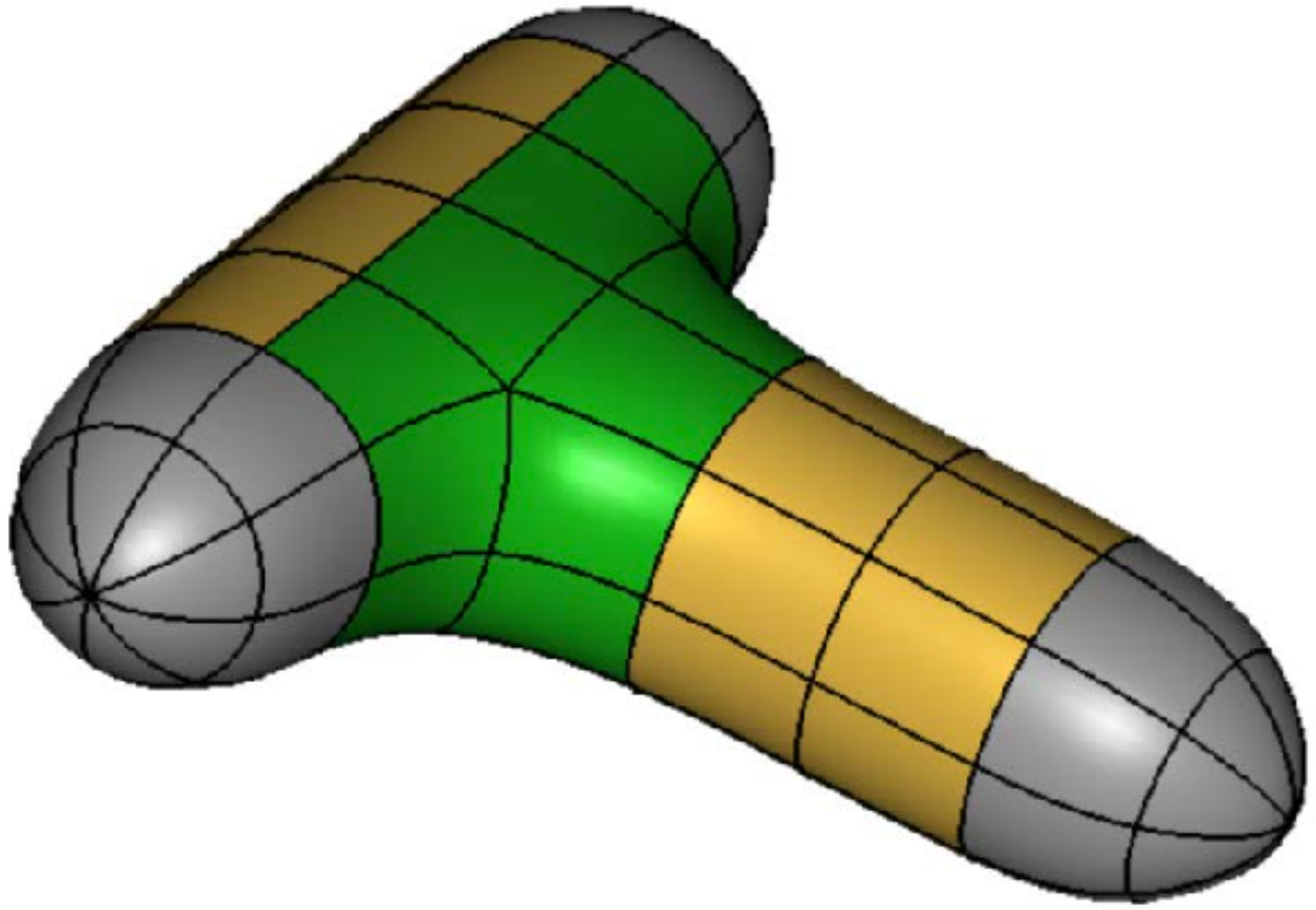
C^0 continuity | C^1 continuity | C^2 continuity

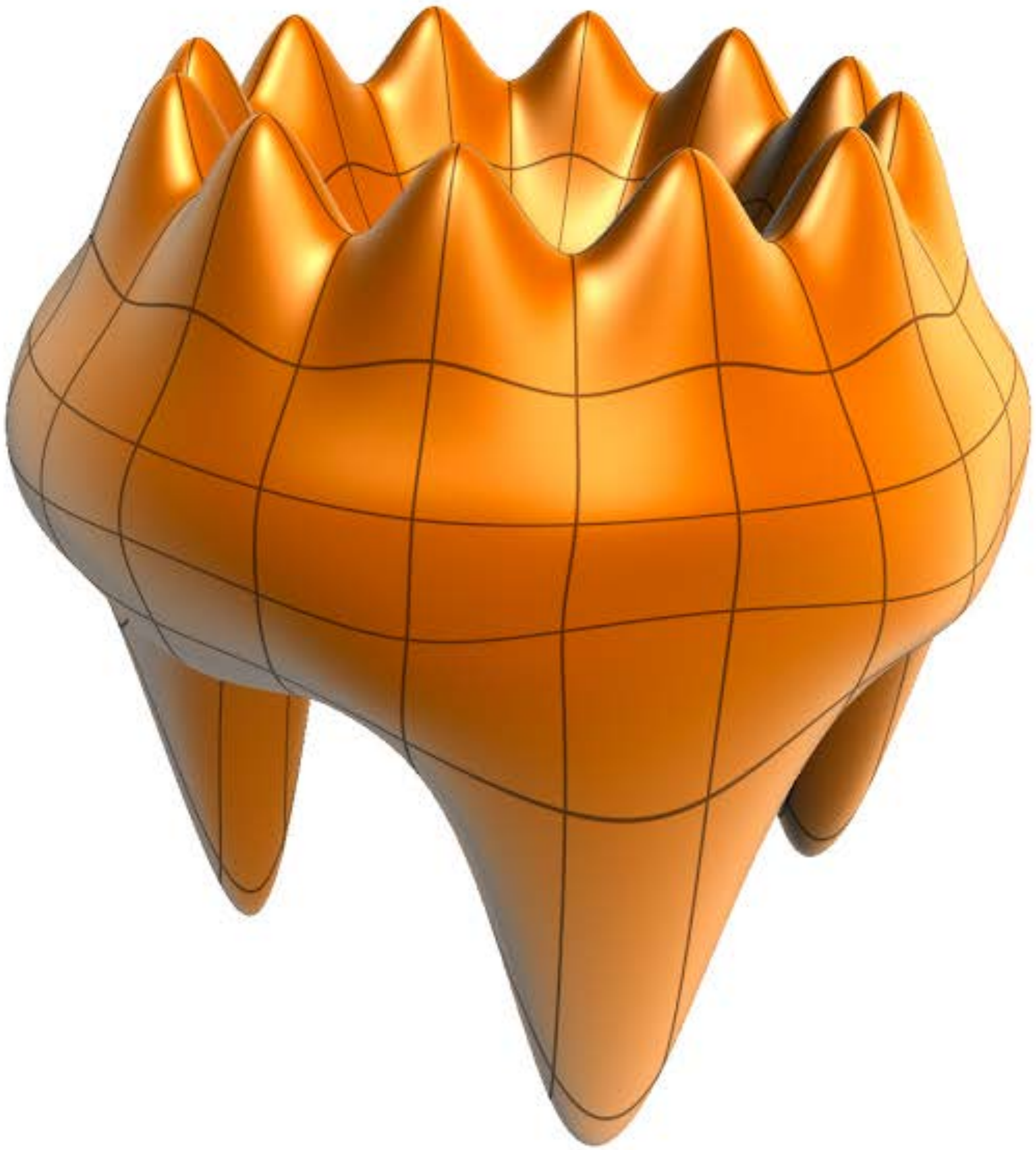
Lief Kobbelt

Continuity is easy to ensure only when

- All patches are quads
- Every corner has 4 adjacent patches

This can be too restrictive!



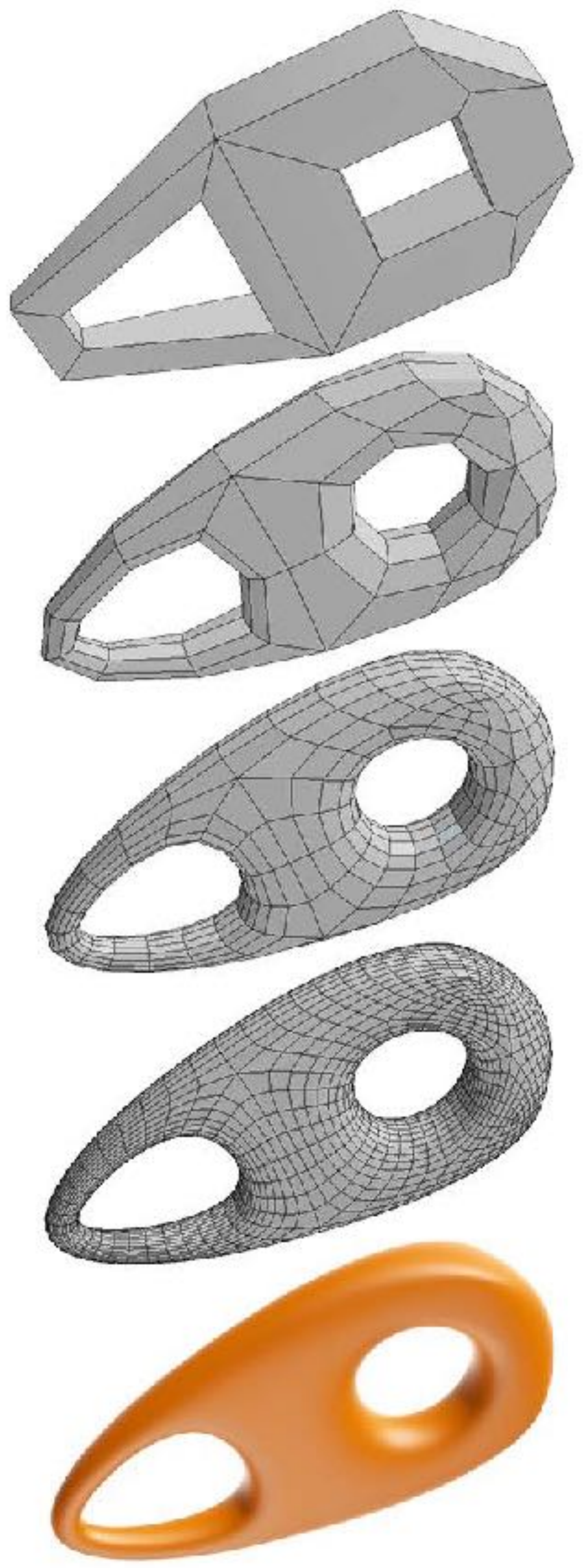


Subdivision

Another strategy to create smooth shapes from a coarse mesh of control points: **subdivision**

- Split each element by inserting new vertices
- Update positions of all vertices by local averaging
- Repeat…

The desired shape is what we converge to in the limit.

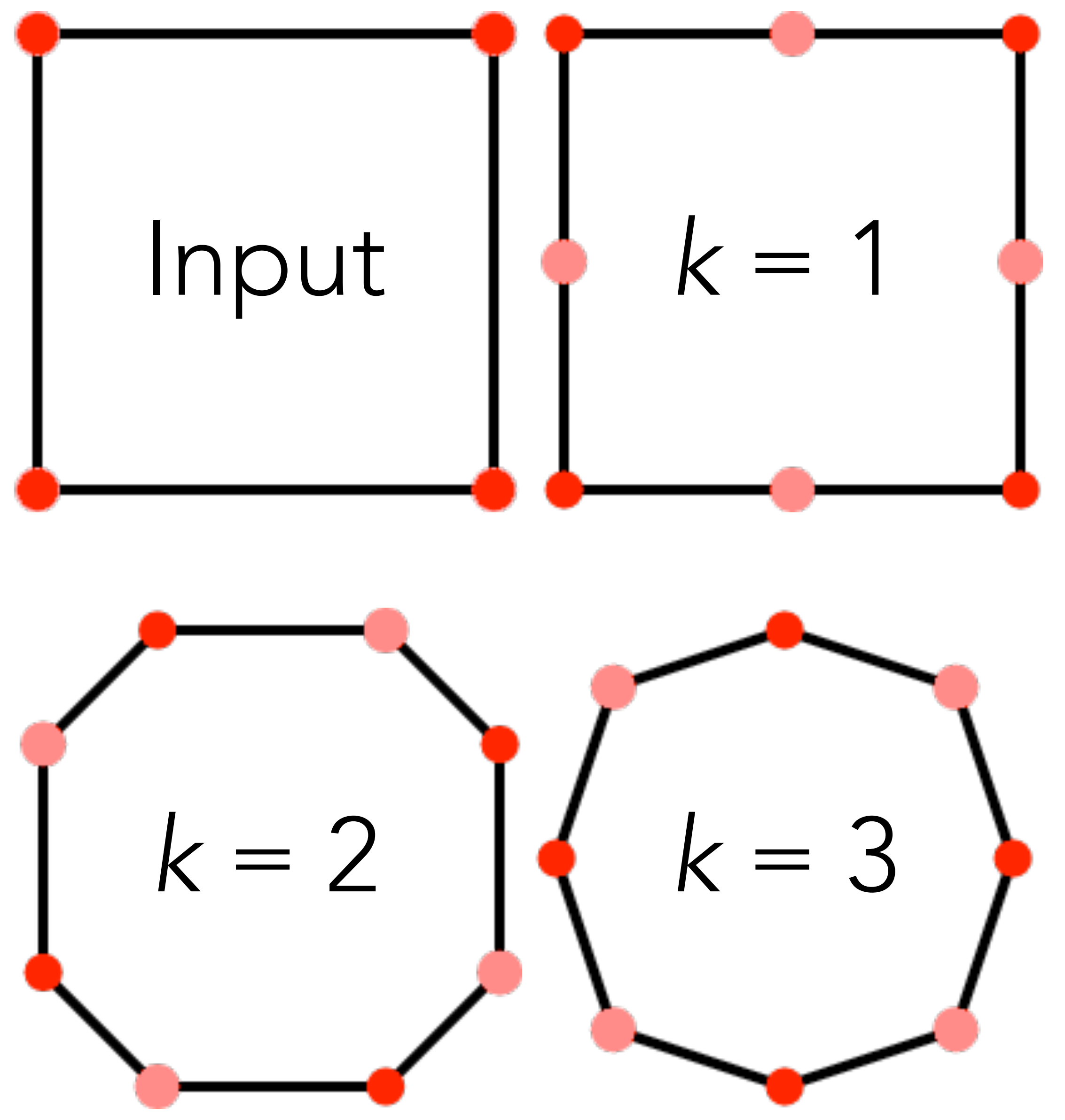


Subdivision curves

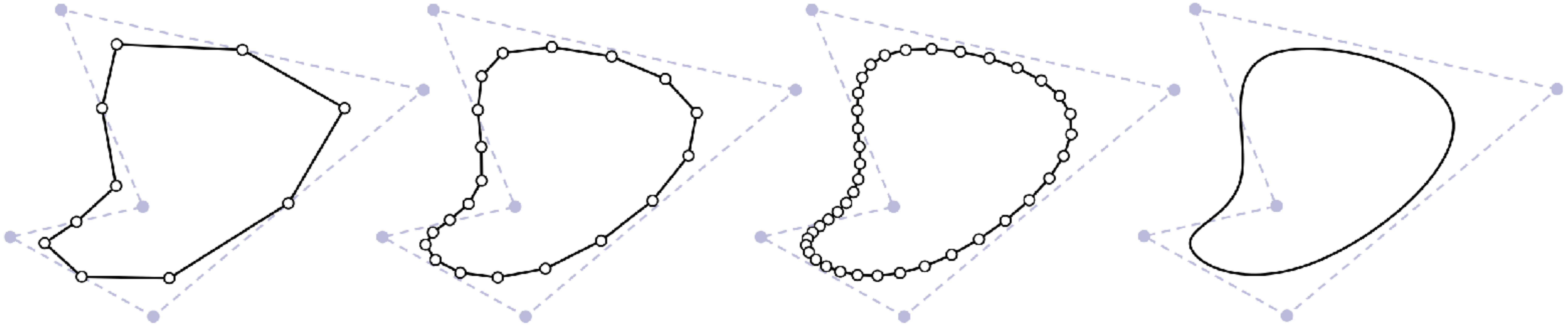
One possible method: Lane-Riesenfeld

- Insert midpoint of each edge
- Repeat k−1 times: Average adjacent vertices

Limit is a degree-k B-spline!



gilgamec

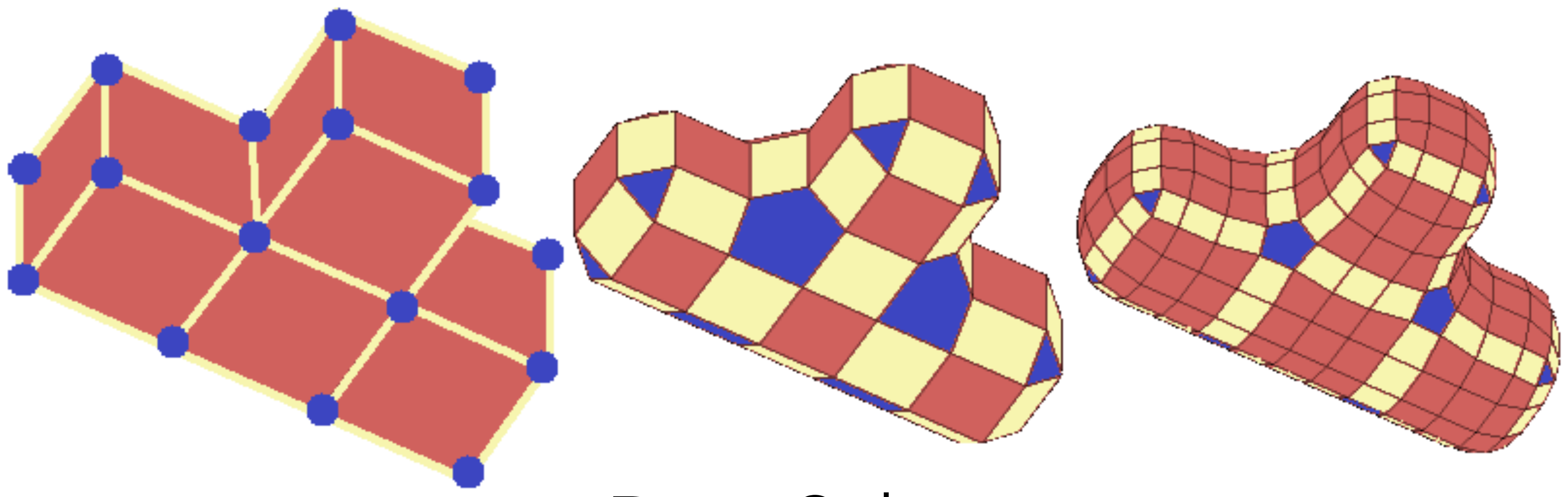


Keenan Crane

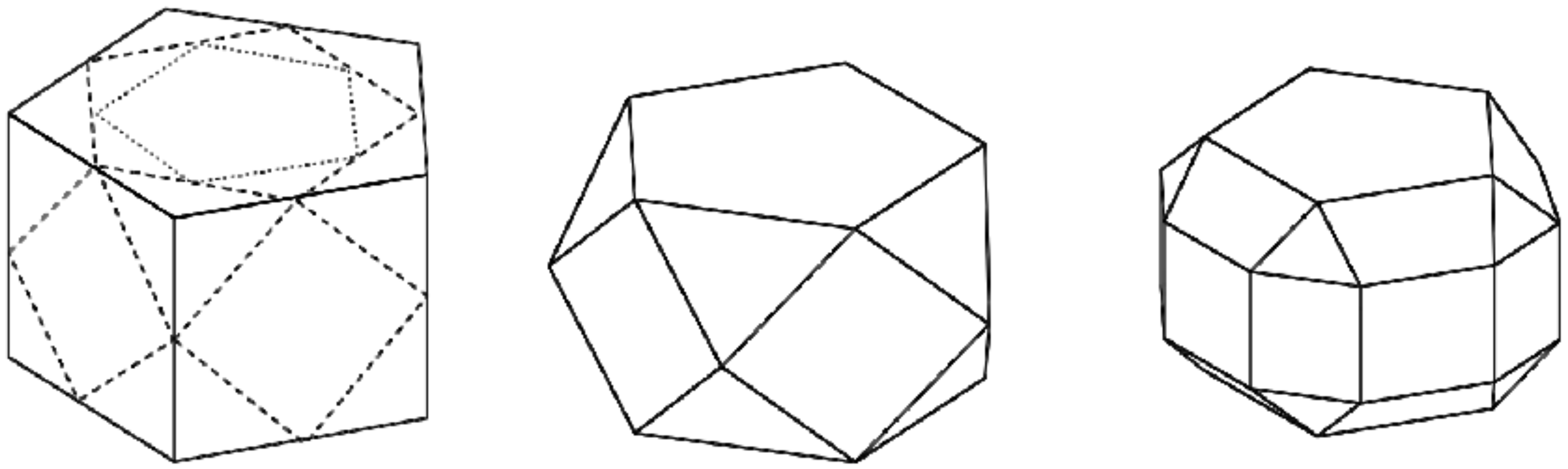
Subdivision surfaces

Connectivity of surfaces is more complicated. Many different subdivision schemes are possible:

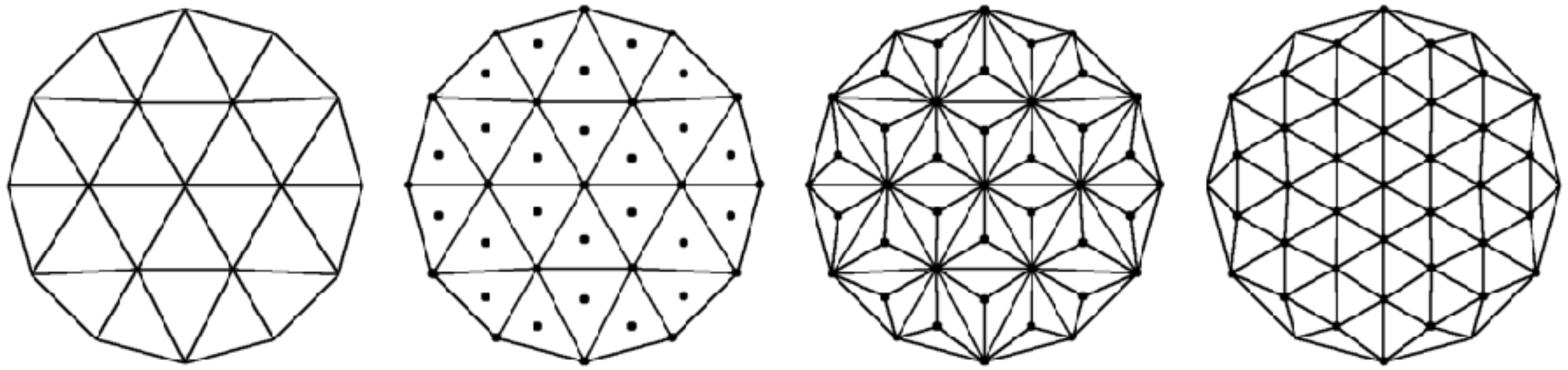
- **General polygon meshes:** Catmull-Clark, Doo-Sabin, mid-edge [Peters & Reif], …
- **Triangle meshes:** Loop, modified butterfly [Zorin et al.], Sqrt(3) [Kobbelt], …



Doo-Sabin



mid-edge



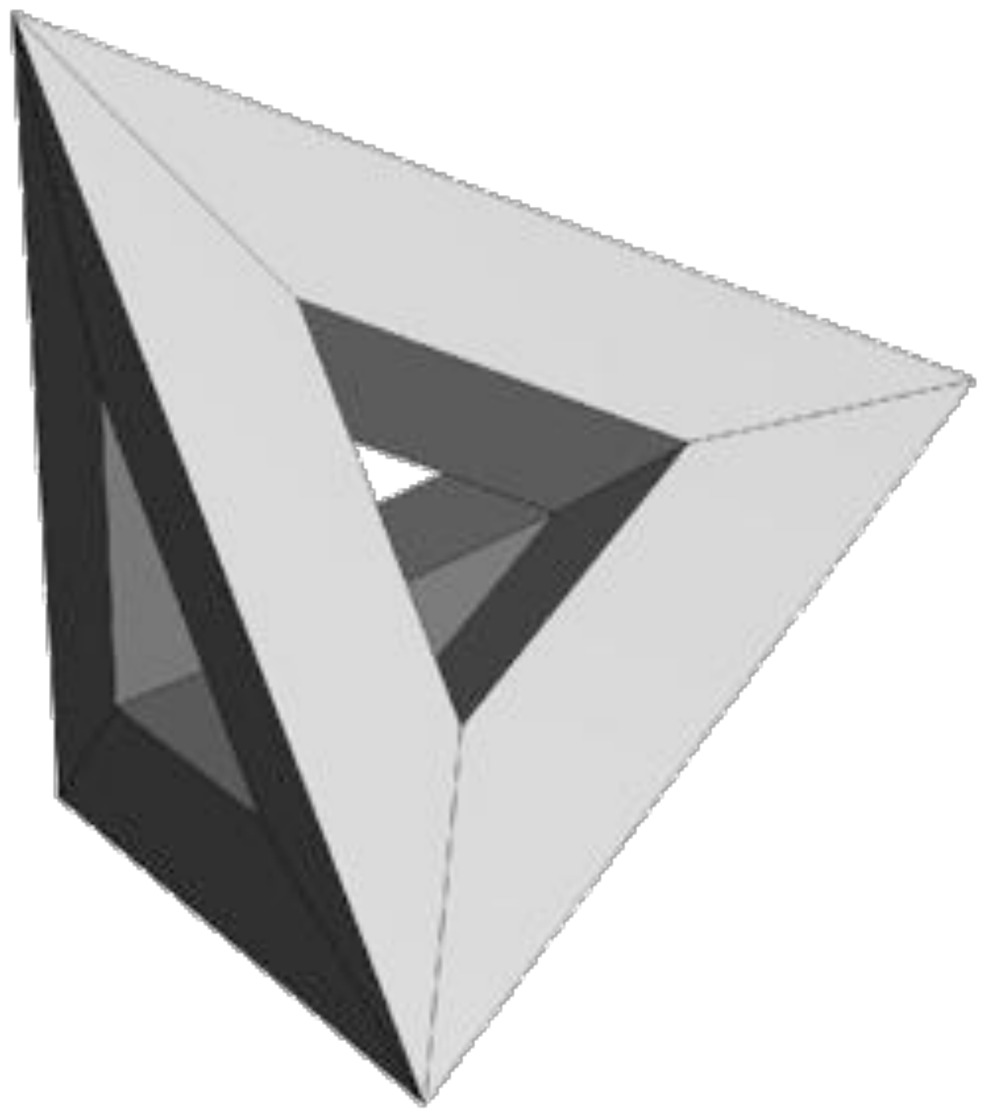
Sqrt(3)

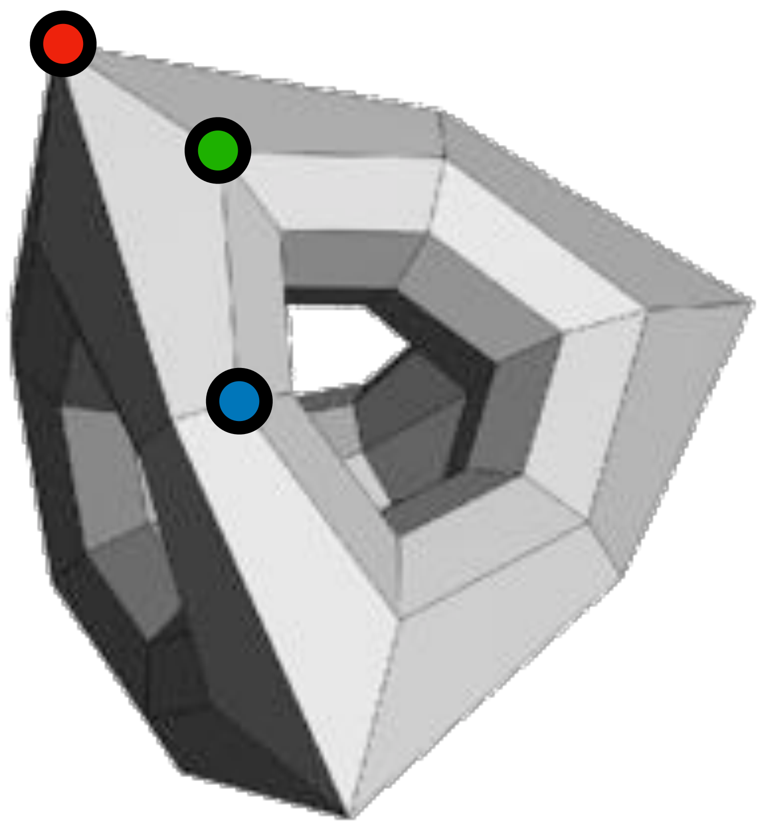
Catmull-Clark subdivision

Split each *n*-sided face into *n* quads

Update vertex positions by averaging:

- New face point = average of old face vertices
- New edge point = average of 2 old vertices and 2 new face points
- Updated vertex = $\frac{1}{n}(Q + 2R + (n-3)S)$
  where Q = average of *n* new face points,
  R = average of *n* new edge points, S = old vertex





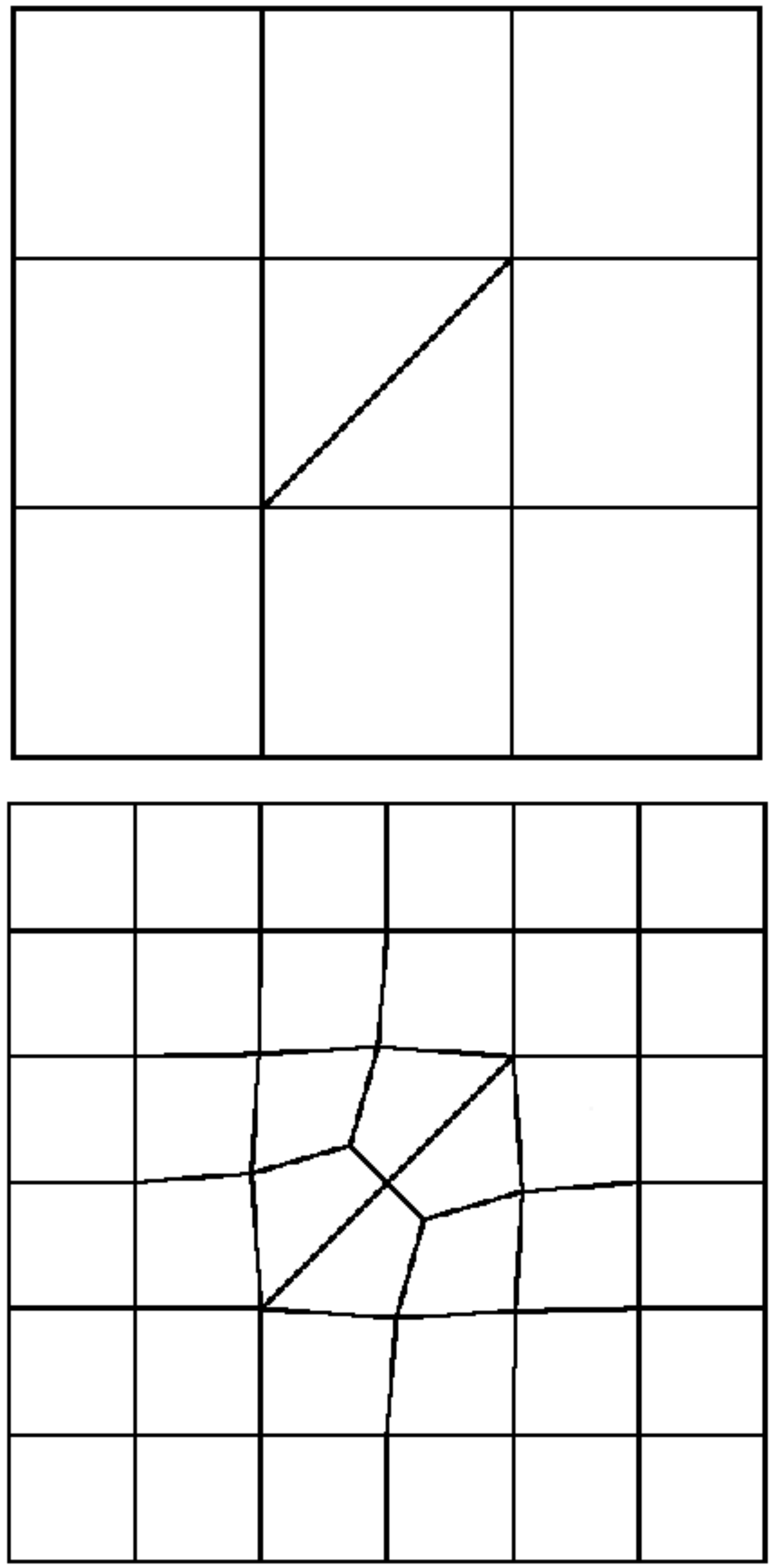
Pixar

After 1 iteration: All faces are quads

After 2 iterations: All new vertices are degree-4

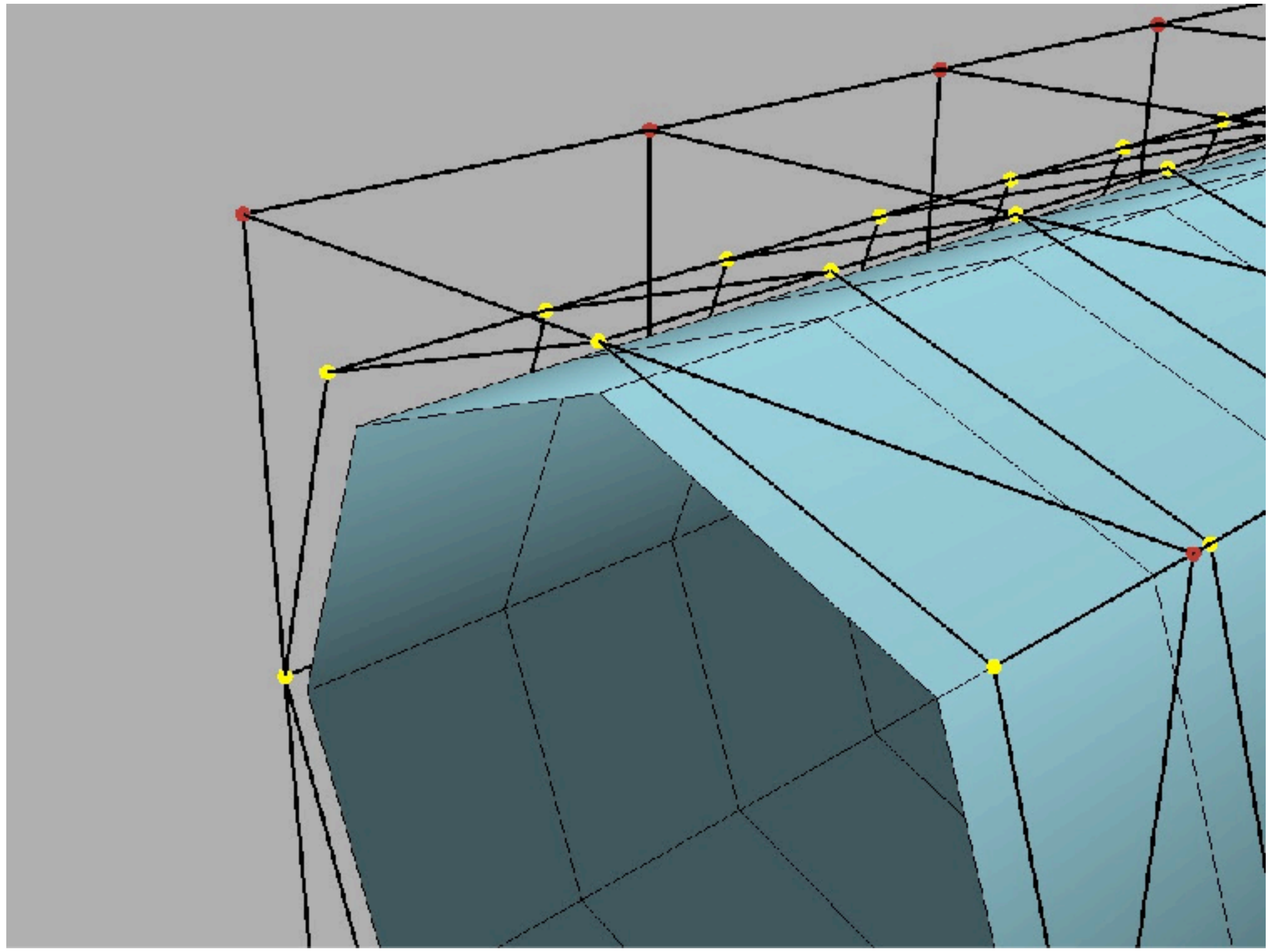
Limit surface has C^2 continuity except at "extraordinary vertices" (with degree ≠ 4).

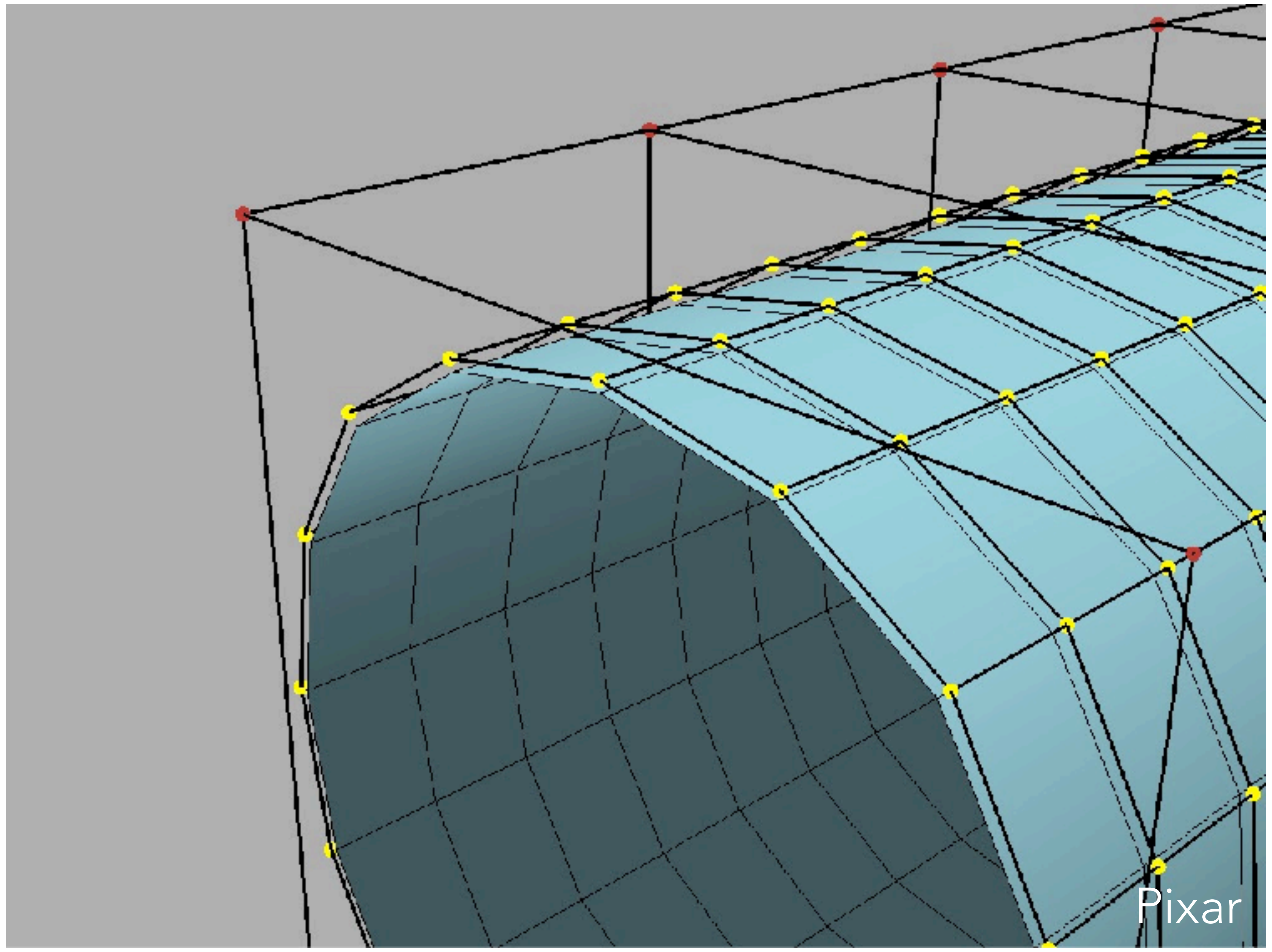
Still C^1 at extraordinary vertices



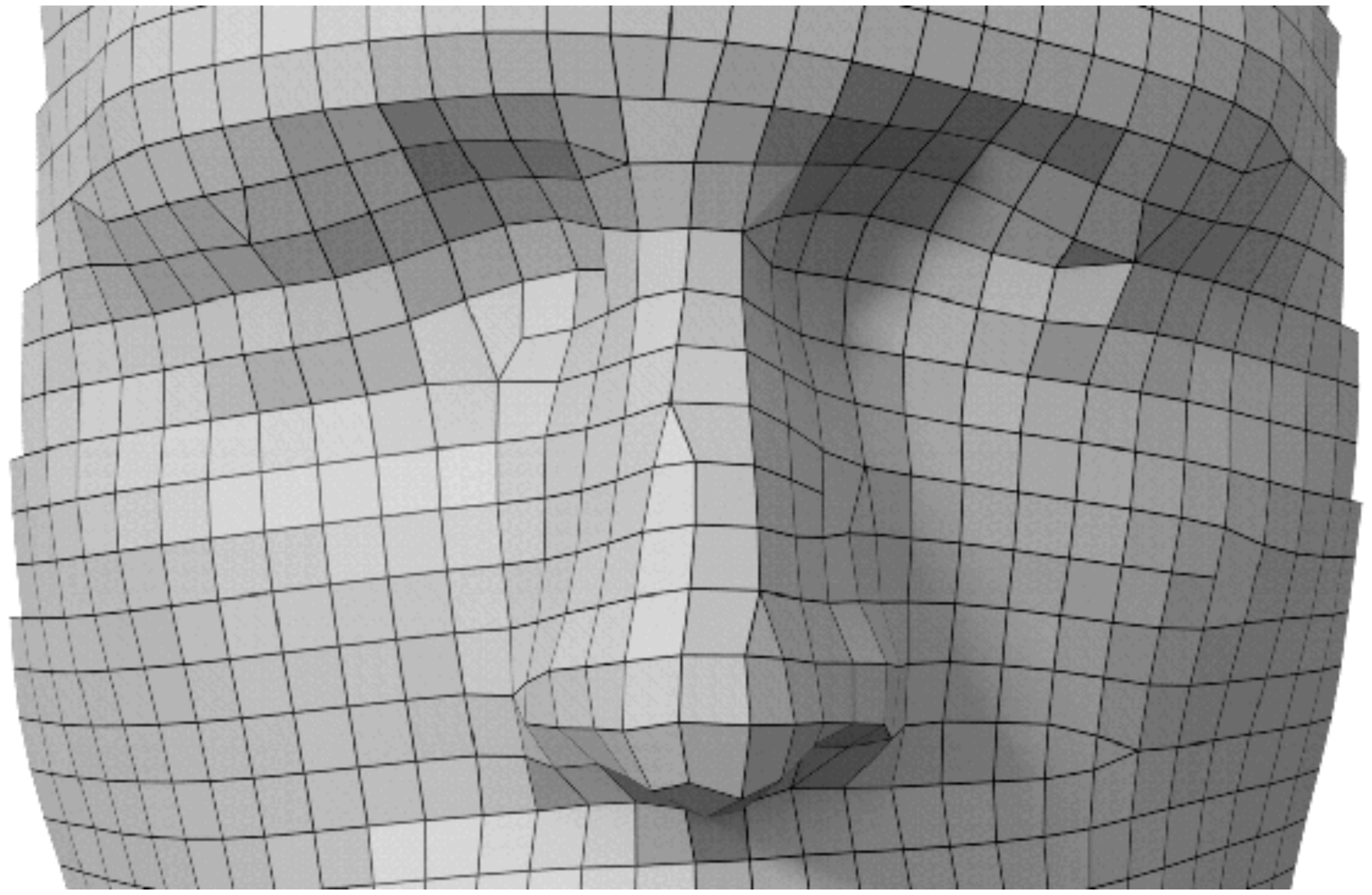
Also possible to directly evaluate limiting position of any point on the surface without recursion! [Stam 1998]

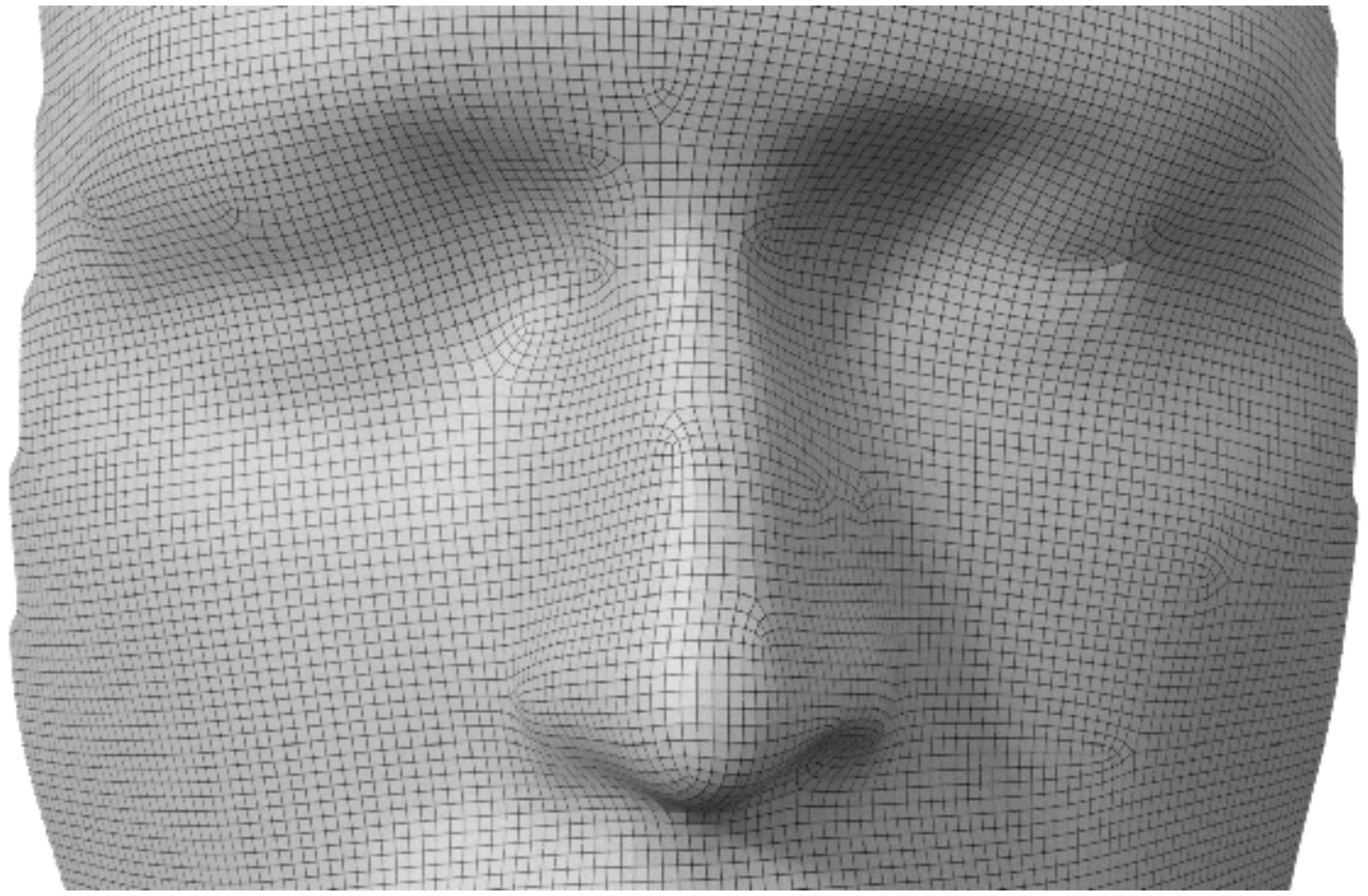
Outside the scope of this course :)

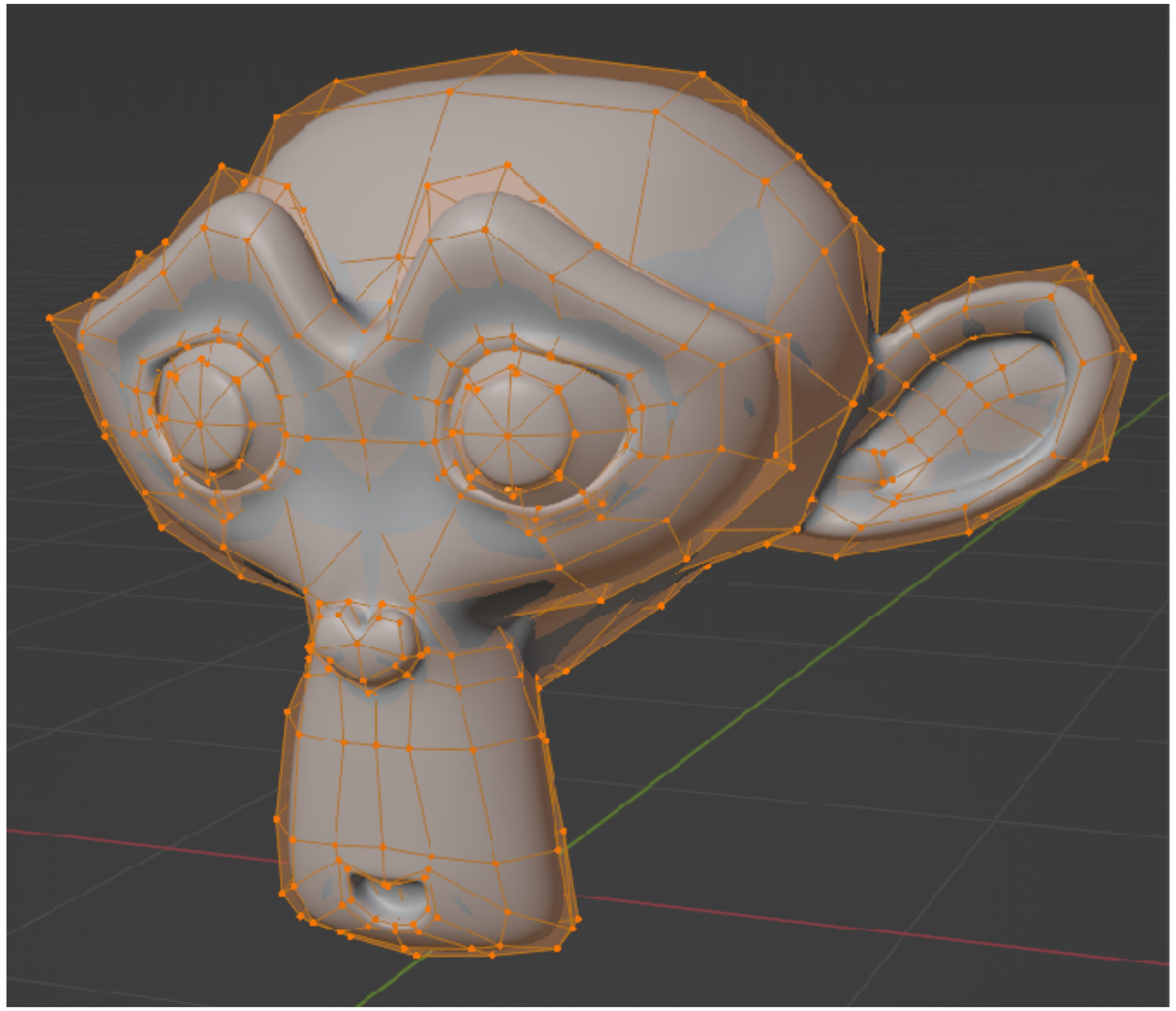


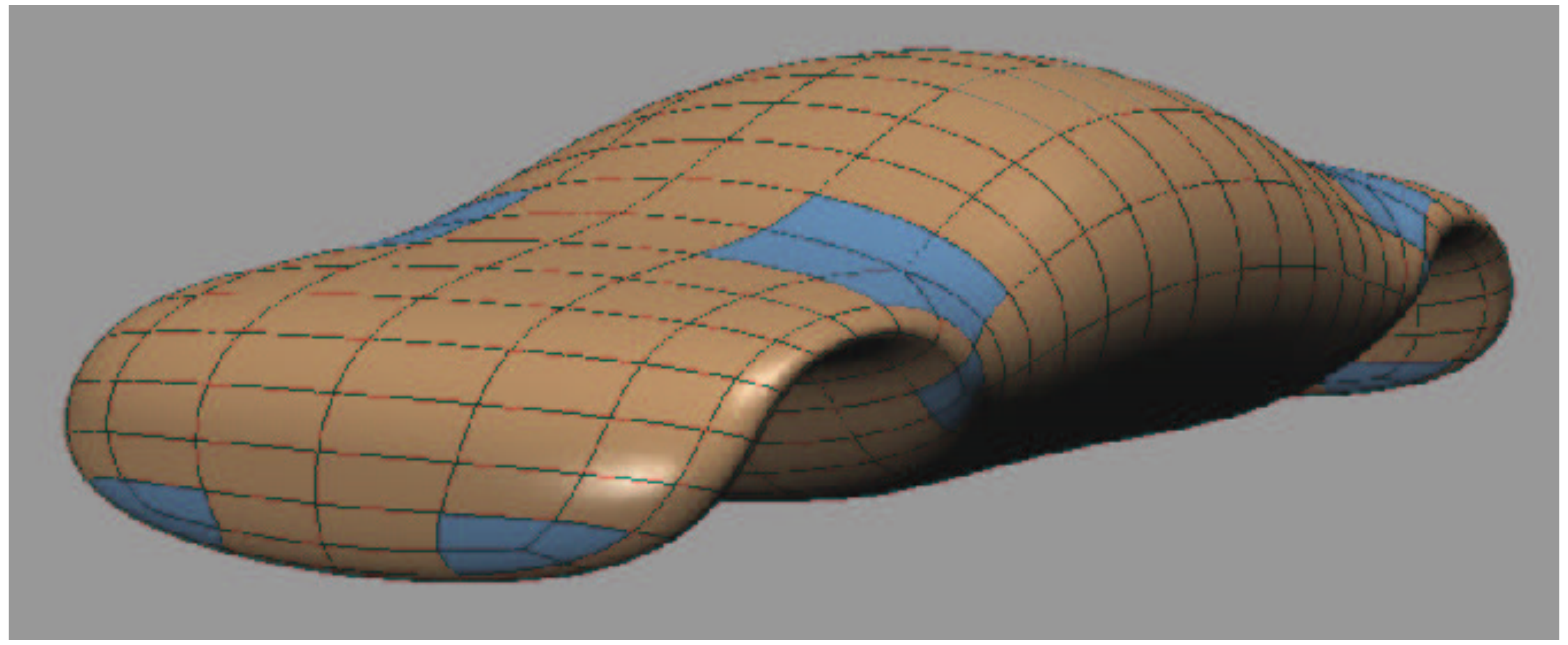


Examples





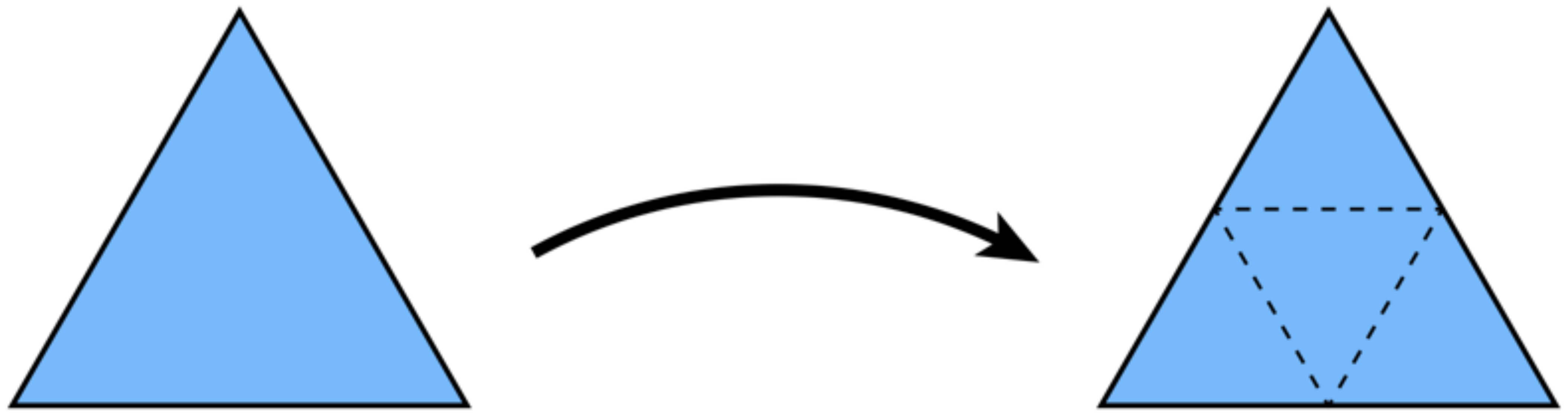




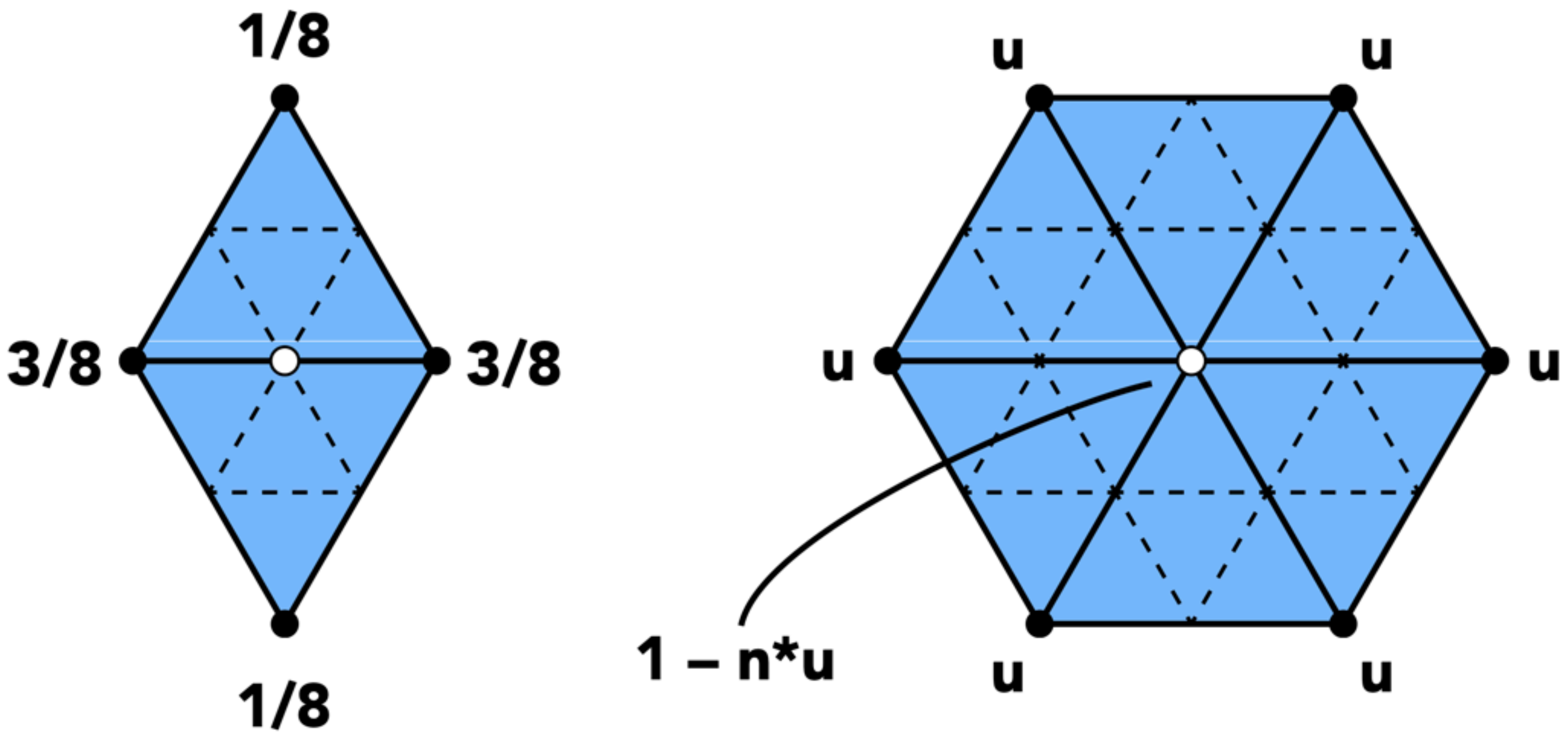
Loop subdivision

(Named after its inventor, Charles T. Loop)

Split each triangle into 4 triangles

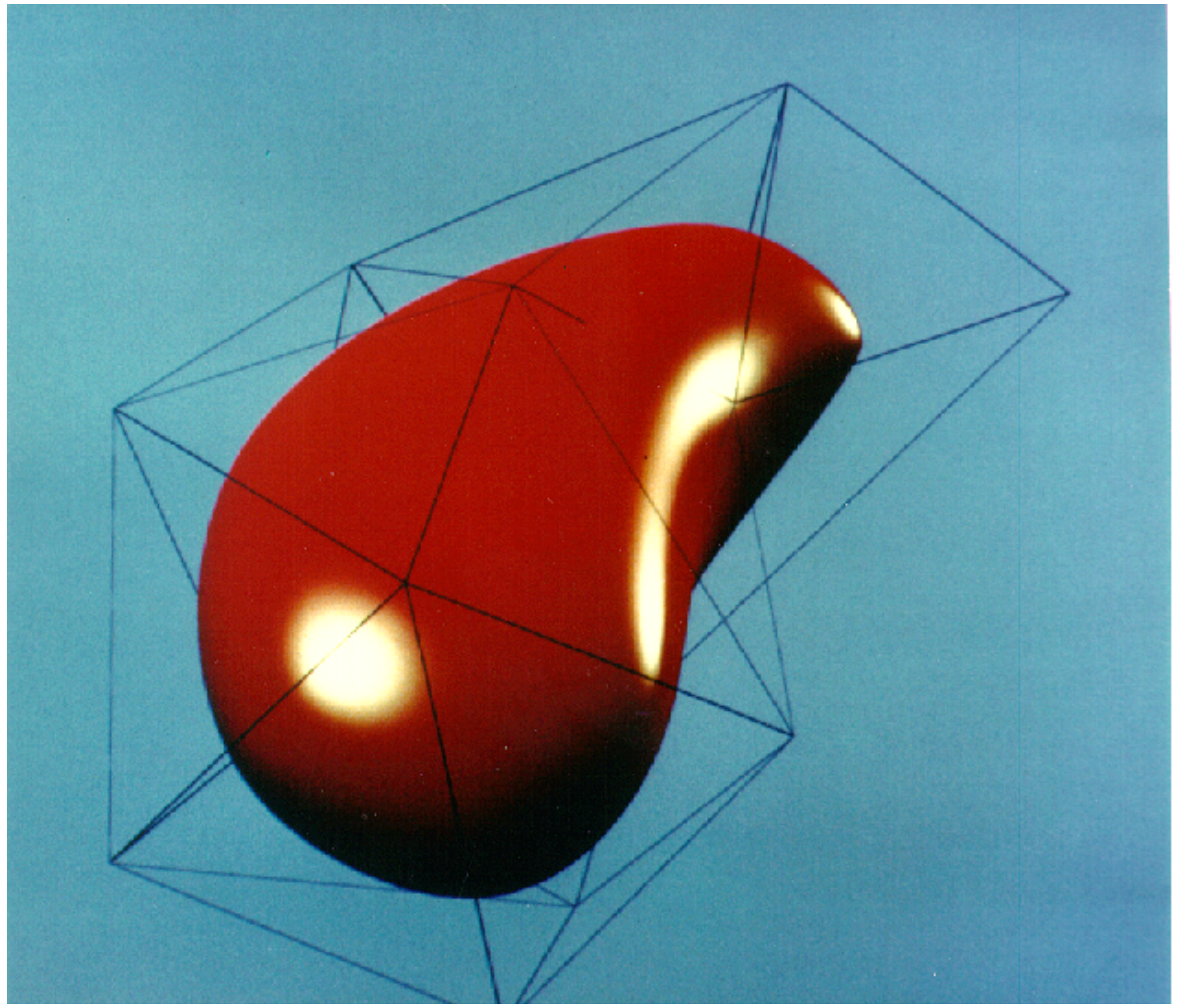


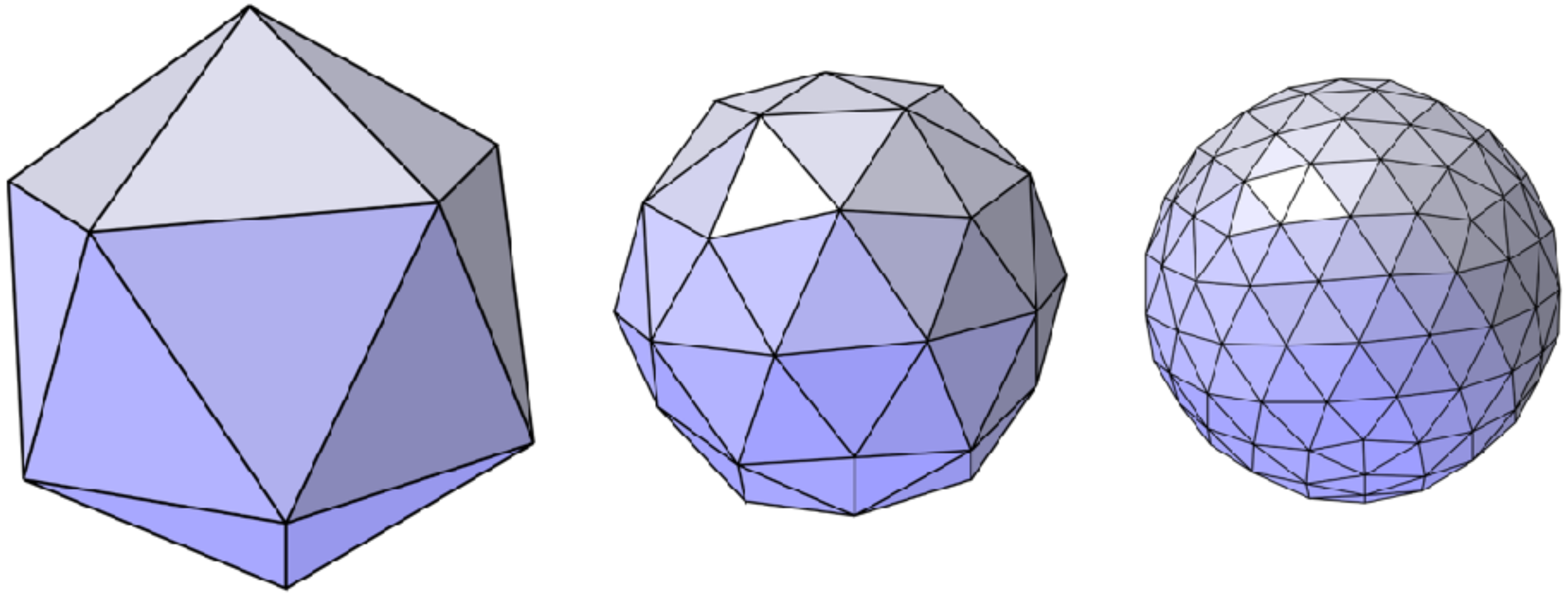
Update vertex positions by averaging:

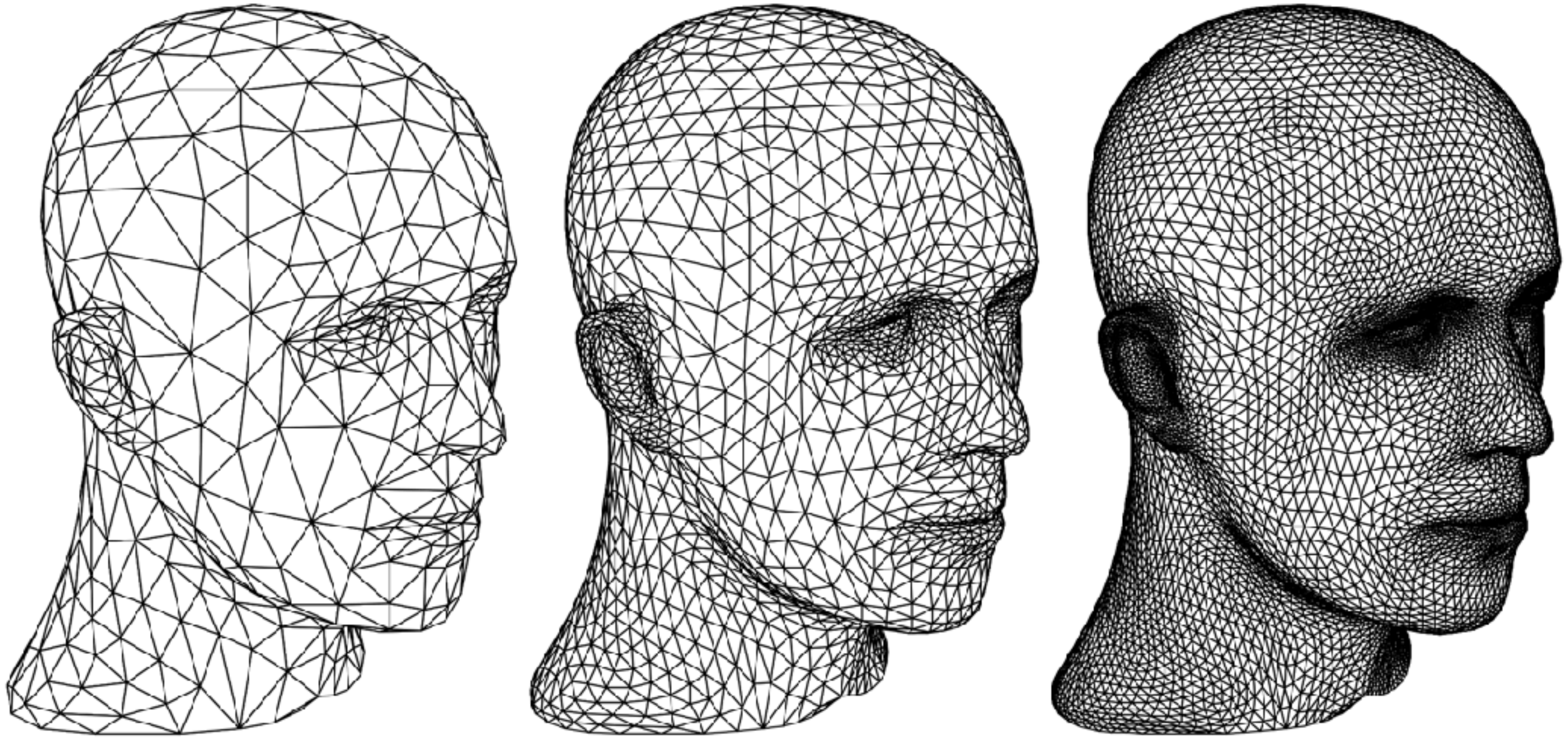


where $u = \begin{cases} 3/16 & \text{if } n = 3, \\ 3/(8n) & \text{otherwise} \end{cases}$

Examples

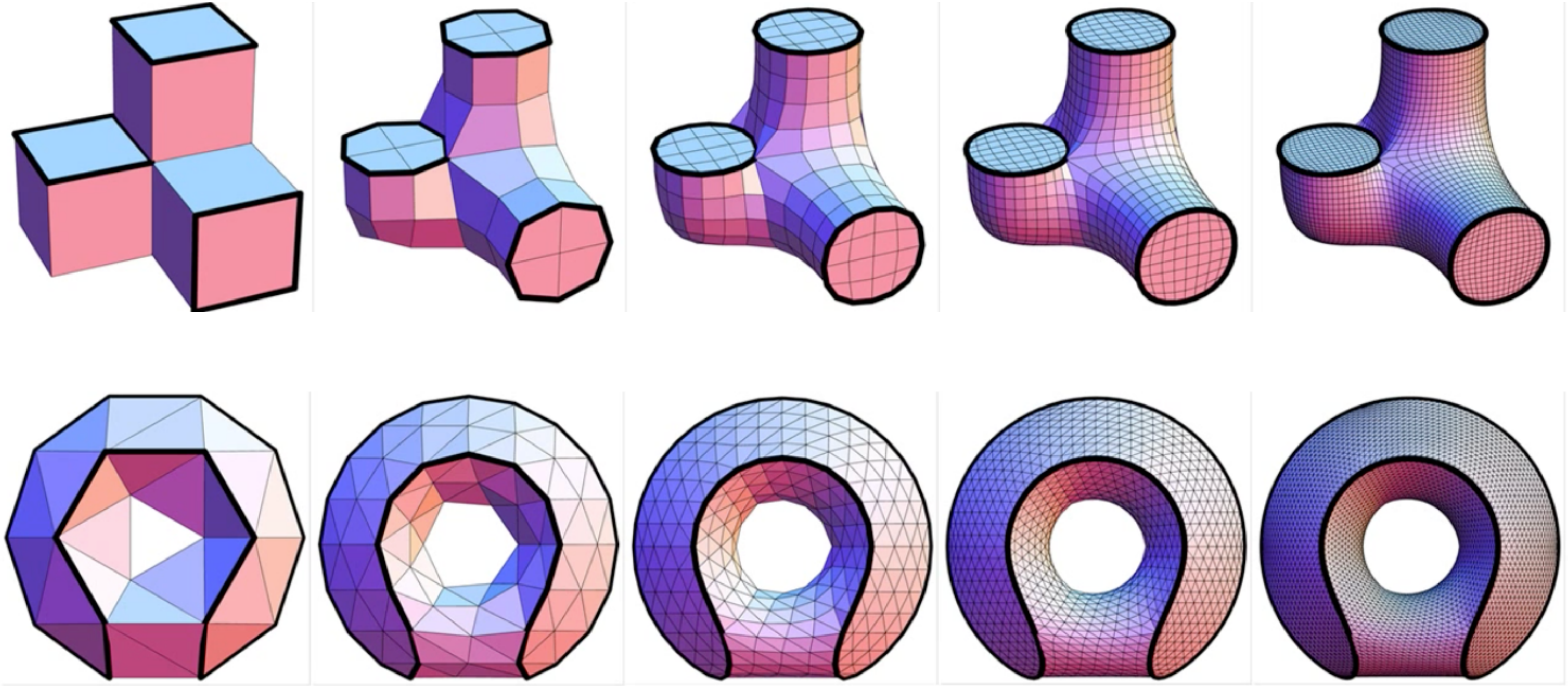






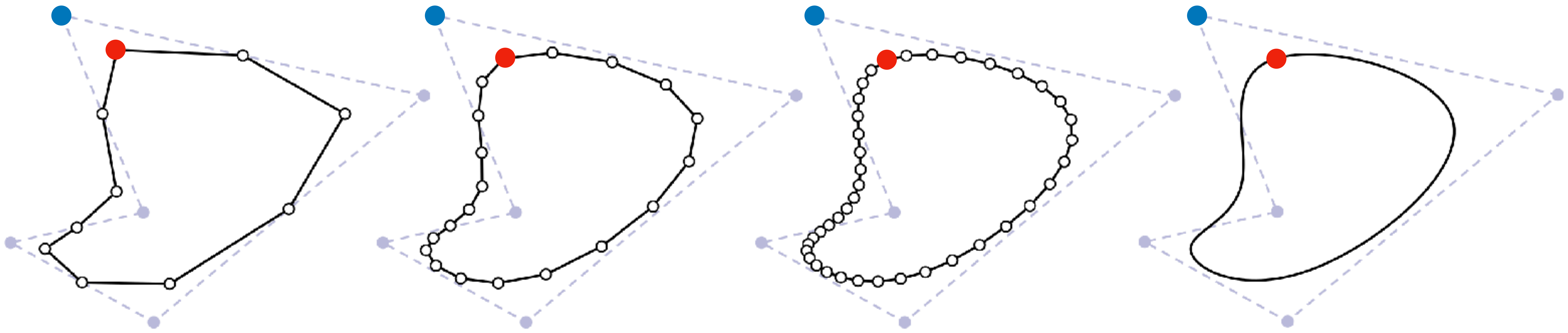
Can also mark creases on the control mesh

Simple: Just use curve subdivision rules for vertices & edges lying on crease



Homework problem

Show that the Lane-Riesenfeld algorithm gives a curve with local control: the limiting position of a vertex depends only on a few adjacent vertices.



Hard mode: For $k = 3$, find a closed-form expression for its limiting position!