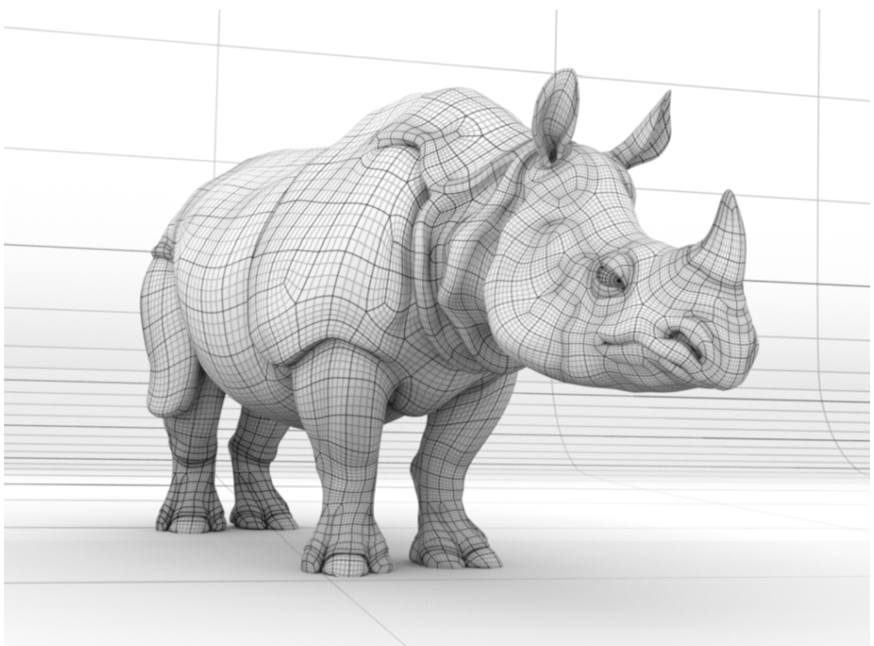
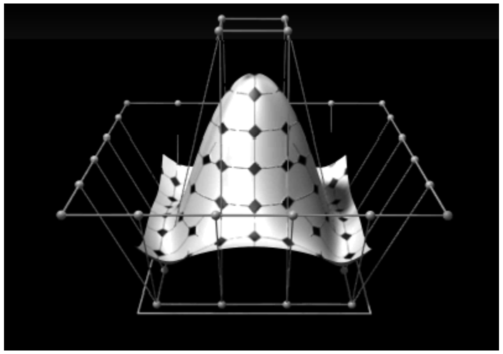
Subdivision Surfaces

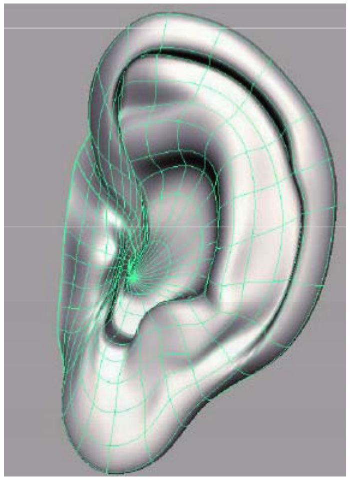


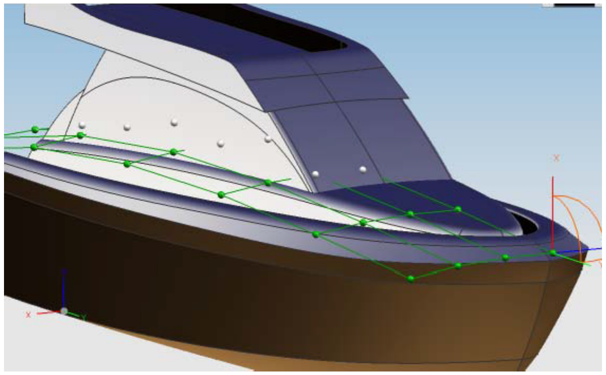
Geometric Modeling

- Sometimes need more than polygon meshes
  - Smooth surfaces



- Traditional geometric modeling used NURBS
  - Non uniform rational B-Spline

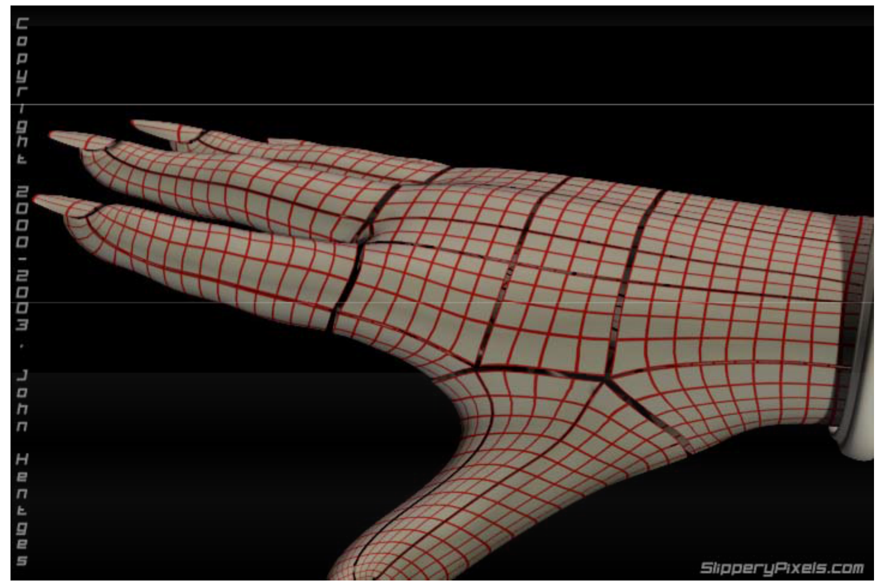




Problems with NURBS

- A single NURBS patch is either a topological disk, a tube or a torus

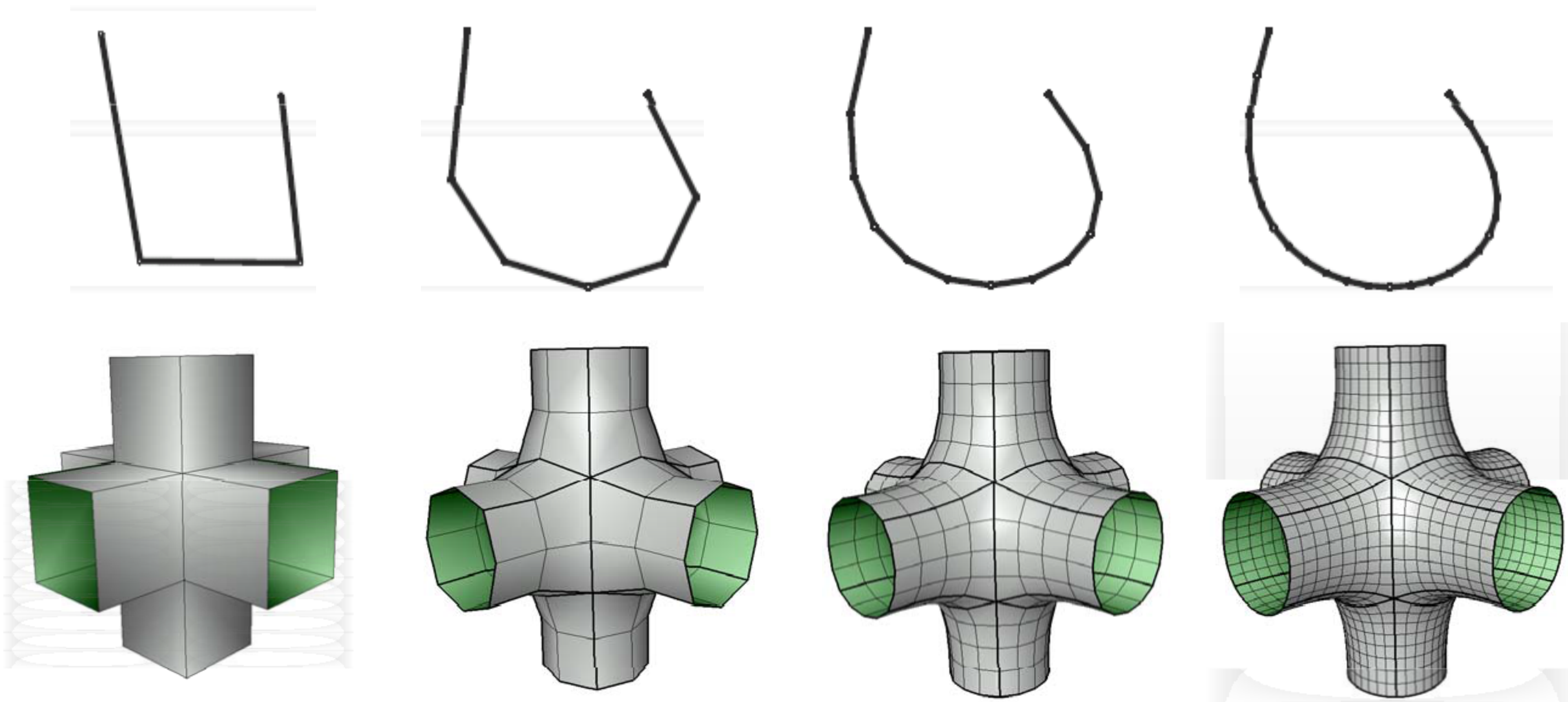
- Must use many NURBS patches to model complex geometry



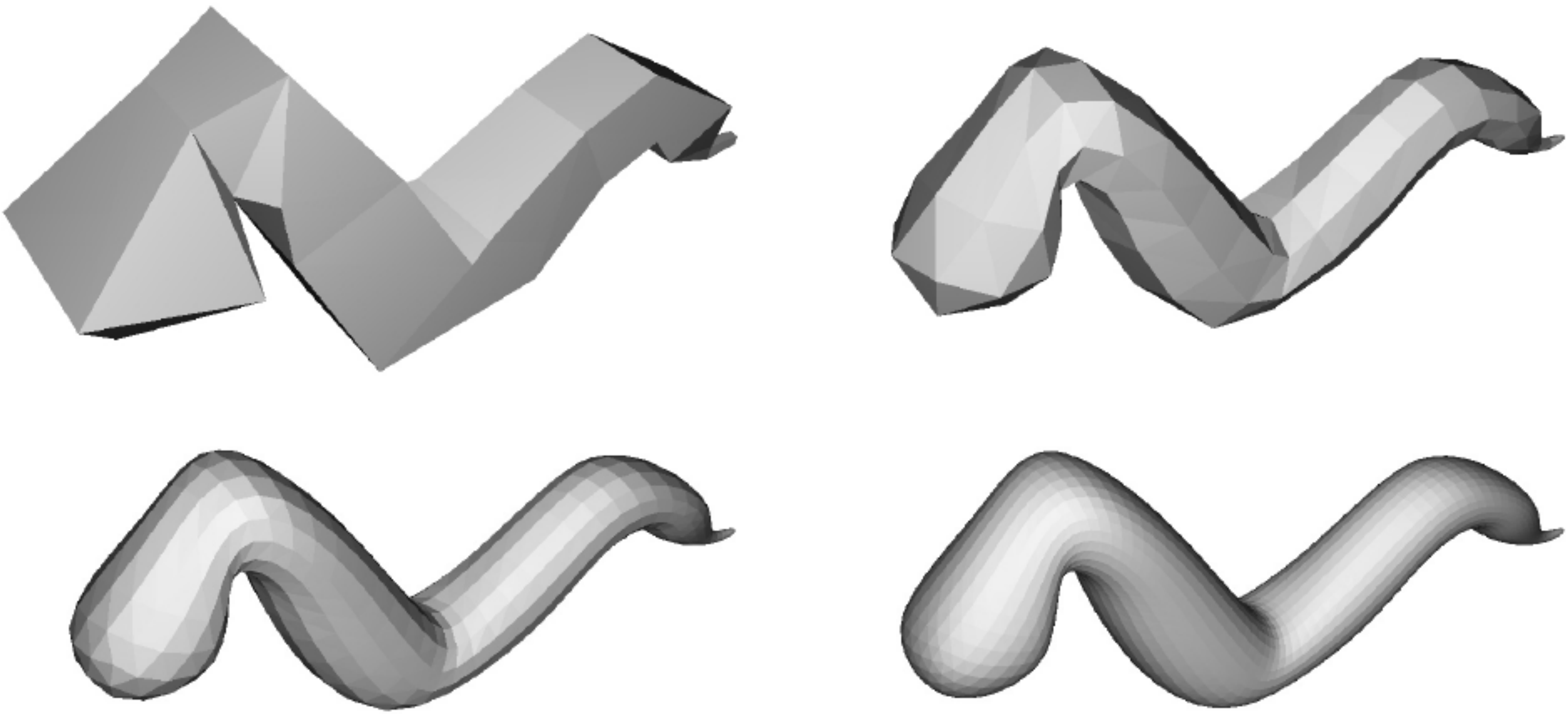
- When deforming a surface made of NURBS patches, cracks arise at the seams

Subdivision

"Subdivision defines a smooth curve or surface as the limit of a sequence of successive refinements"



Subdivision

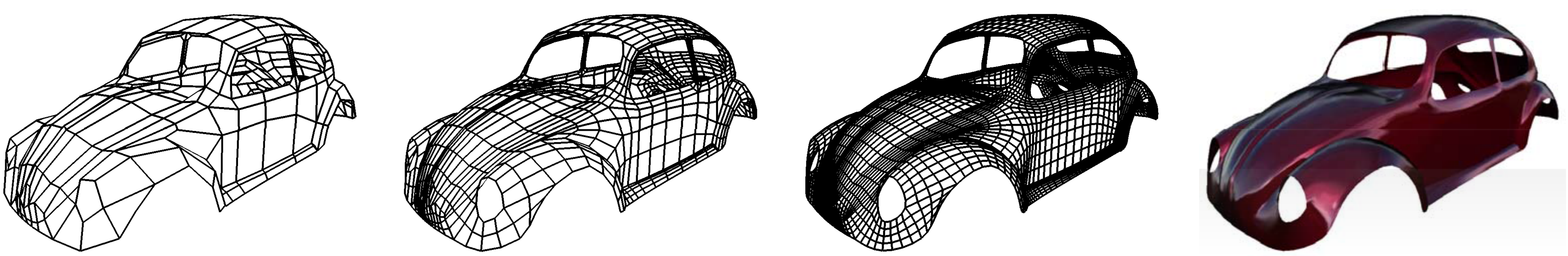


Subdivision Surfaces

- Generalization of spline curves / surfaces
  - Arbitrary control meshes
  - Successive refinement (subdivision)
  - Converges to smooth limit surface
  - Connection between splines and meshes

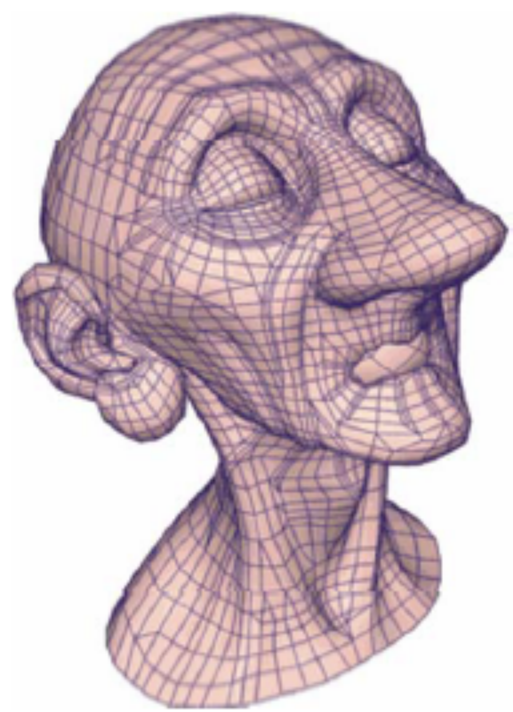
Subdivision Surfaces

- Generalization of spline curves / surfaces
  - Arbitrary control meshes
  - Successive refinement (subdivision)
  - Converges to smooth limit surface
  - Connection between splines and meshes



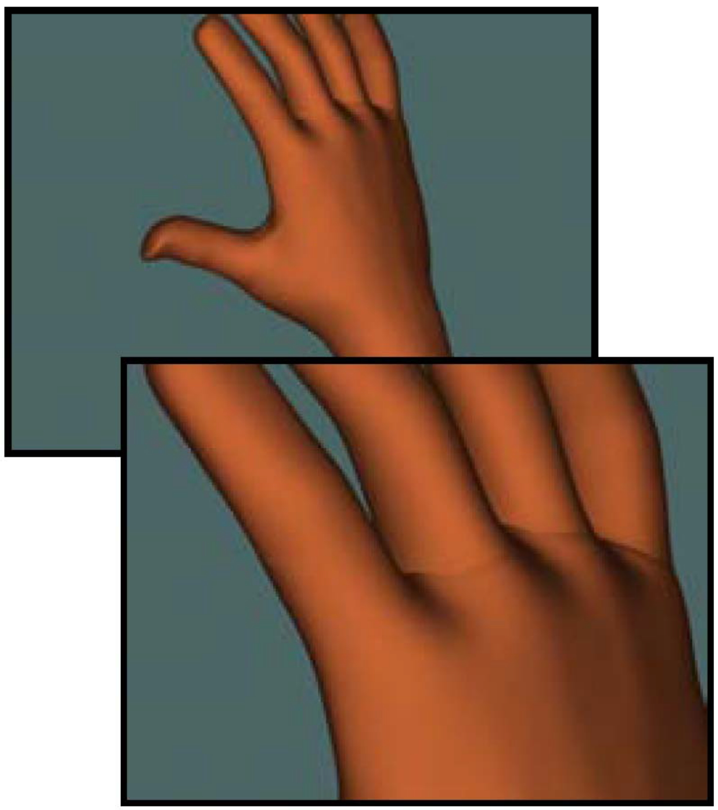
Example: Geri's Game (Pixar 1997)

- Subdivision used for
  - Geri's hands and head
  - Clothing
  - Tie and shoes

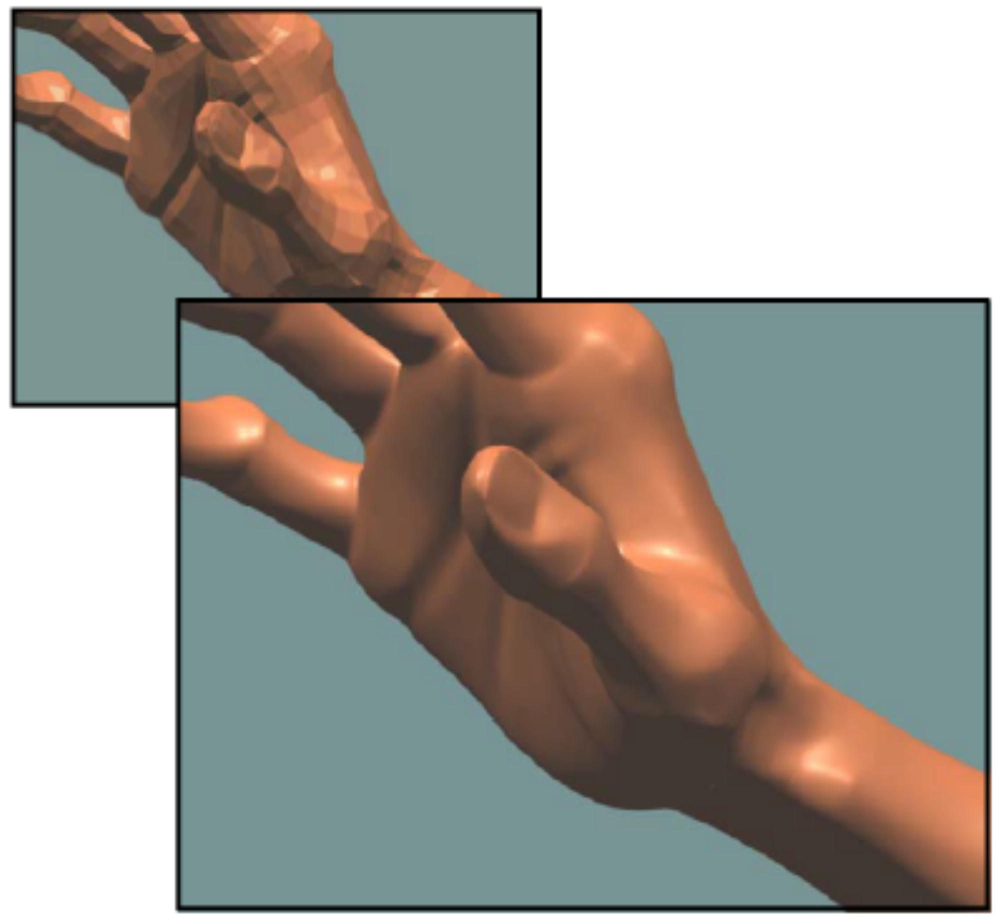


Example: Geri's Game (Pixar)

Woody's hand (NURBS)



Geri's hand (subdivision)

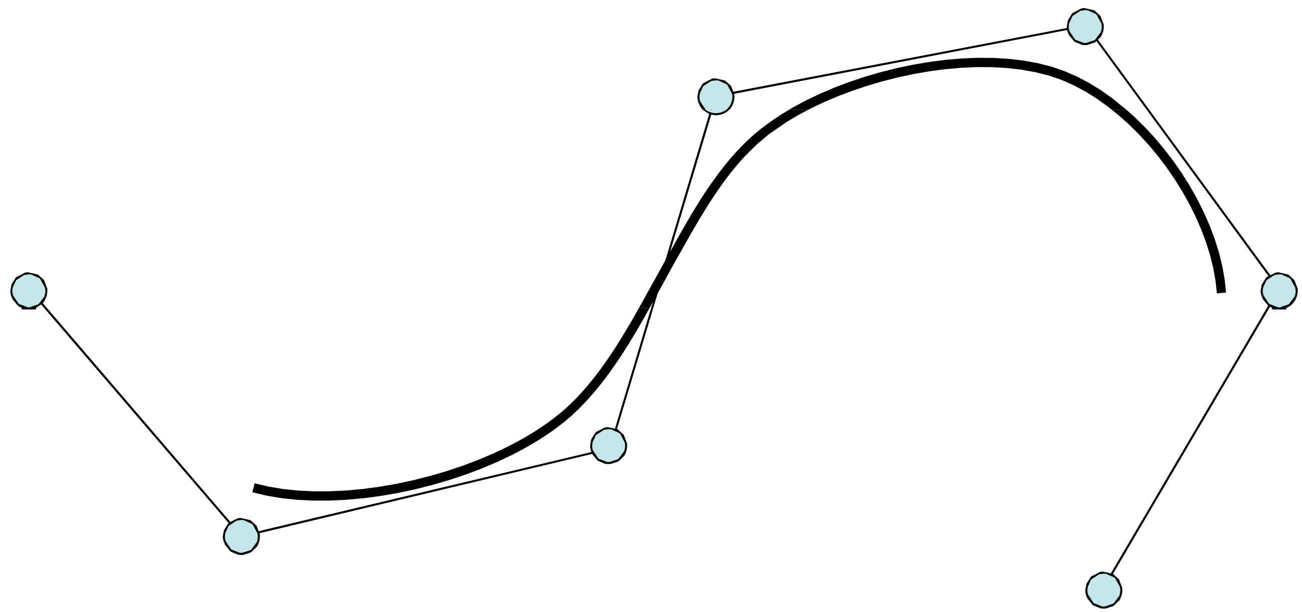


Example: Geri's Game (Pixar)

- Sharp and semi-sharp features

Subdivision Curves

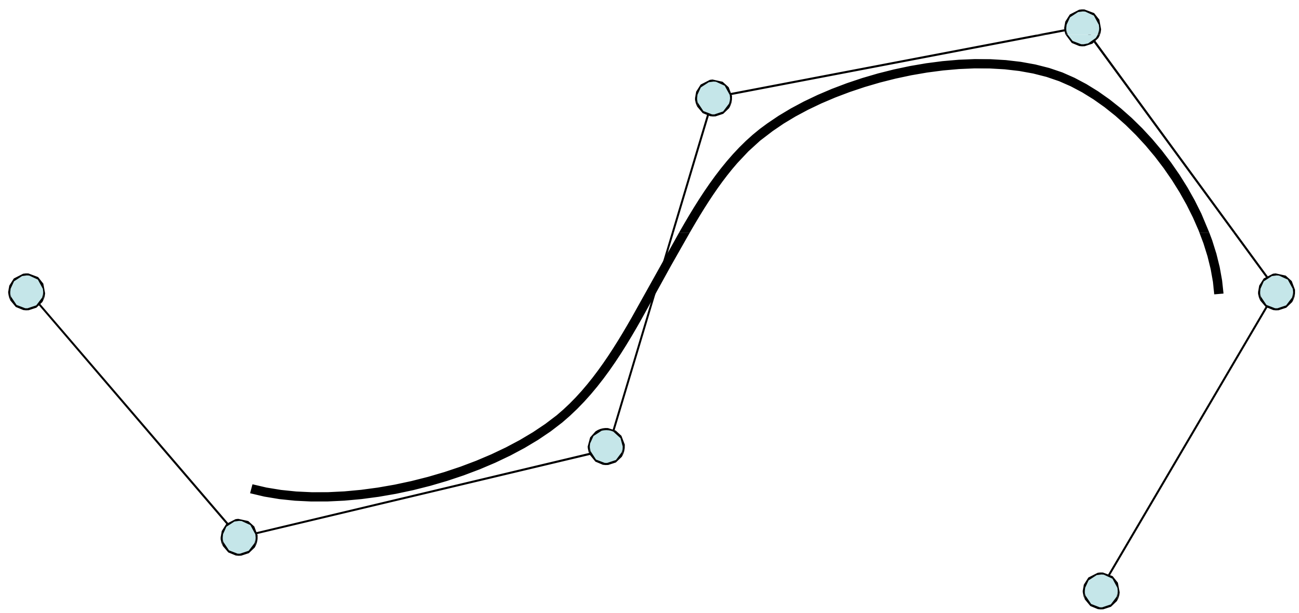
Given a control polygon...



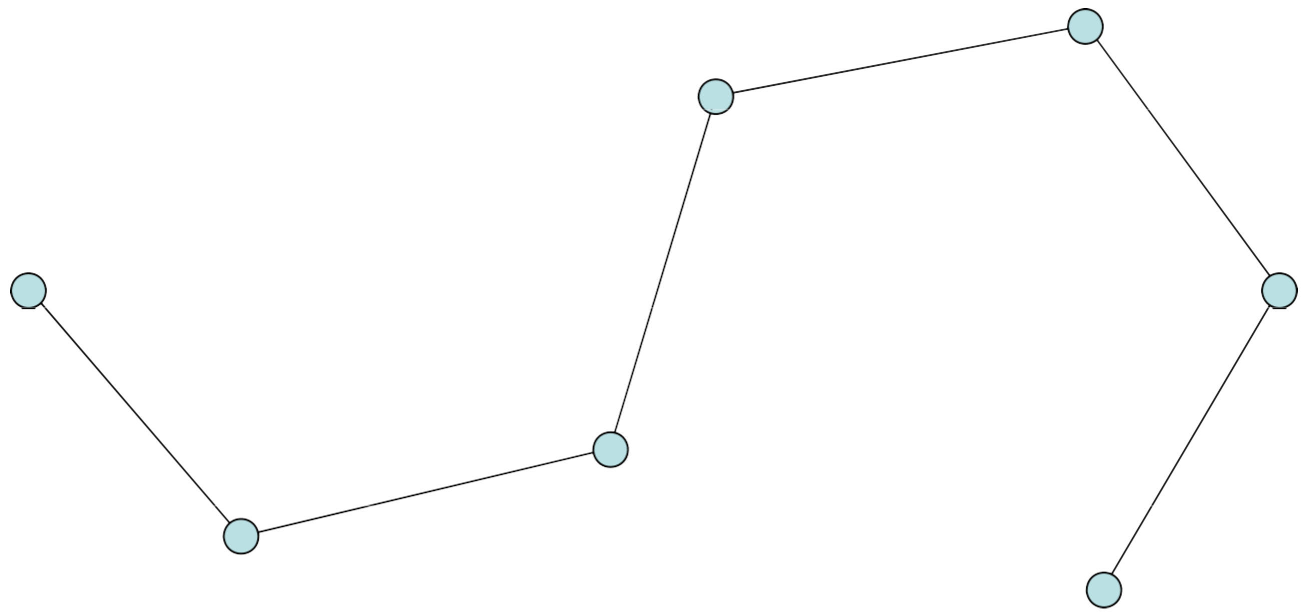
...find a smooth curve related to that polygon.

Subdivision Curve Types

- Approximating
- Interpolating

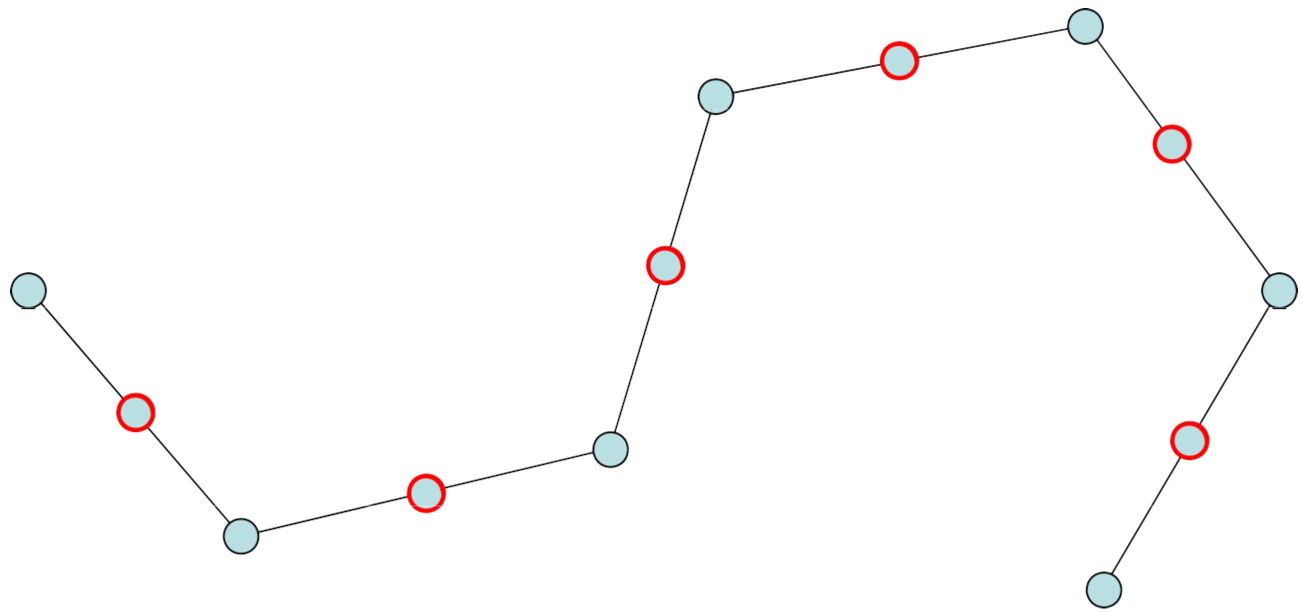


Approximating



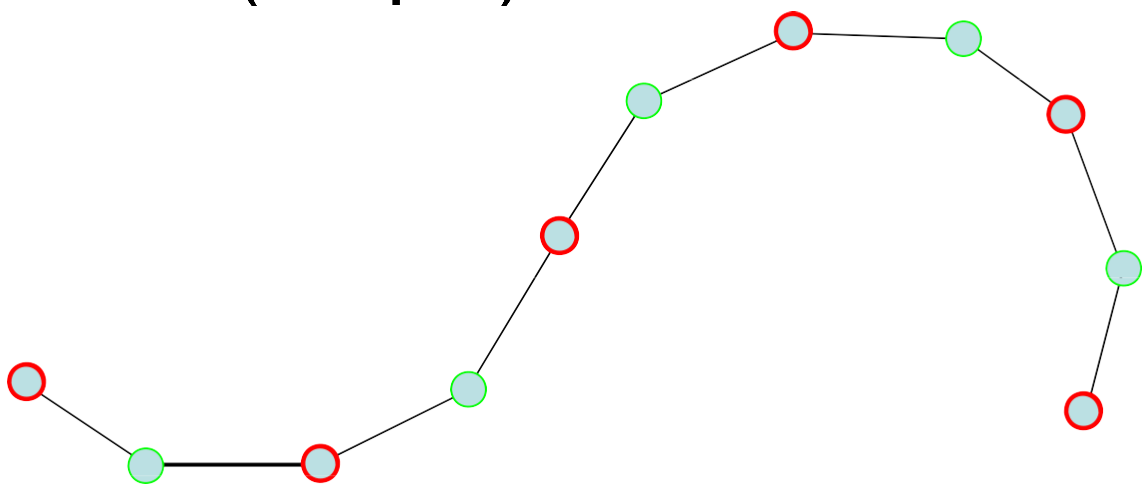
Approximating

Splitting step: split each edge in two



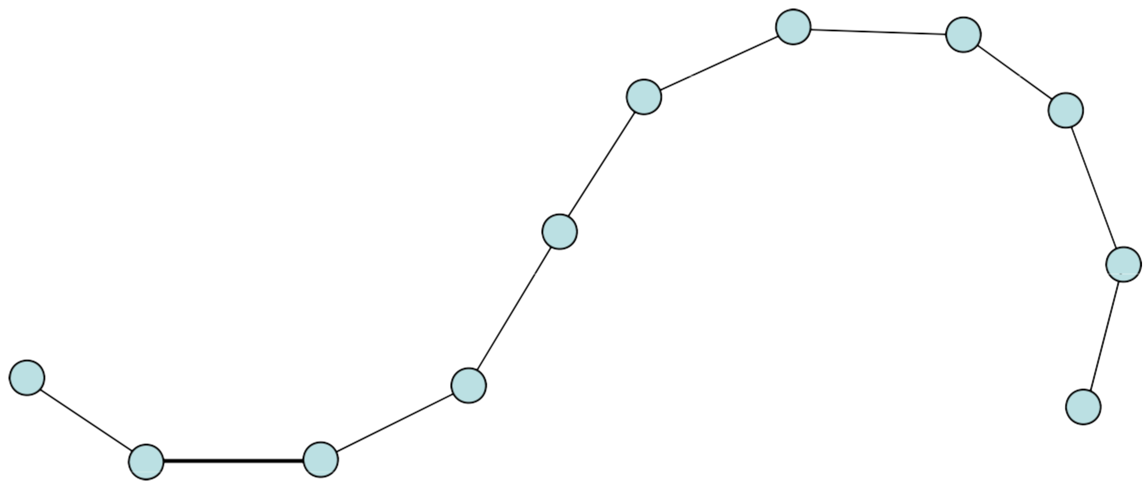
Approximating

Averaging step: relocate each (original) vertex according to some (simple) rule...



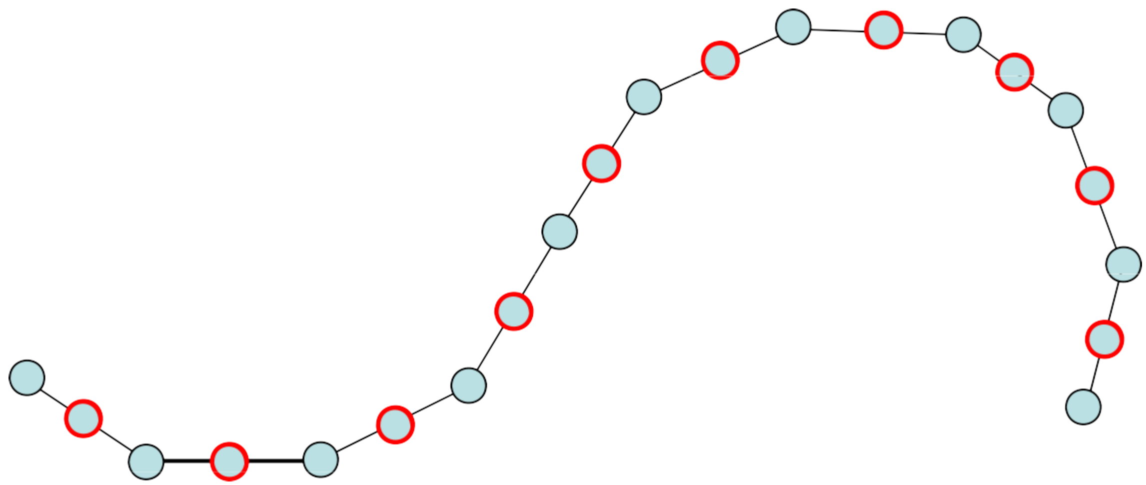
Approximating

Start over ...



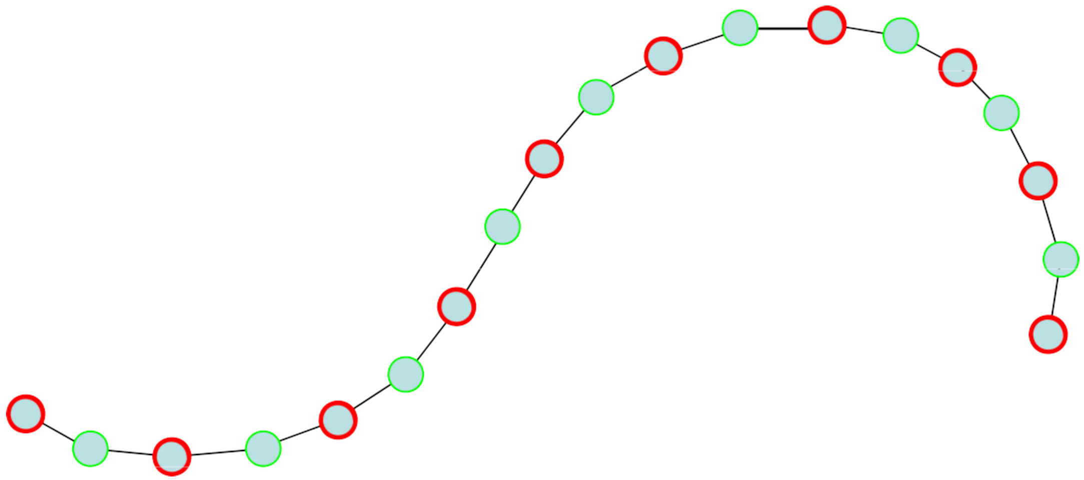
Approximating

...splitting...



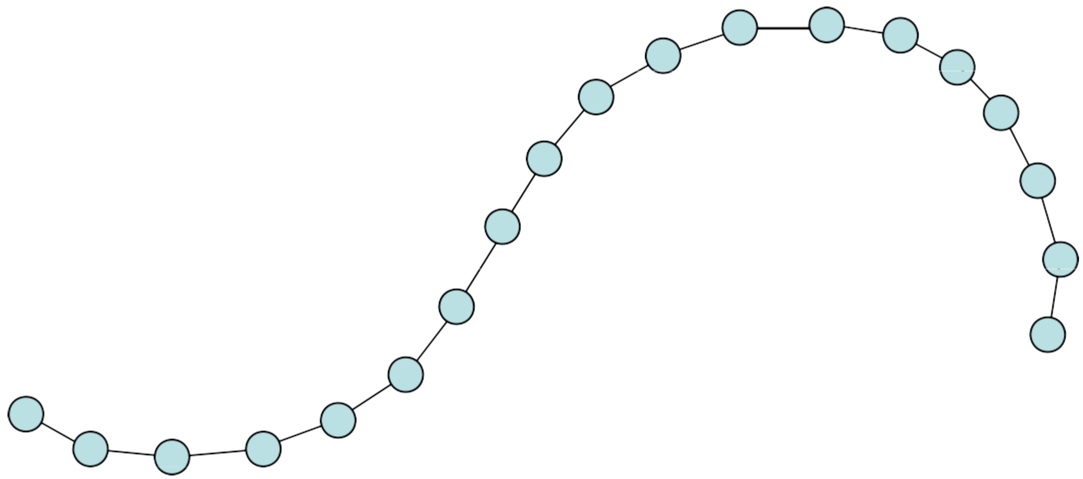
Approximating

...averaging...



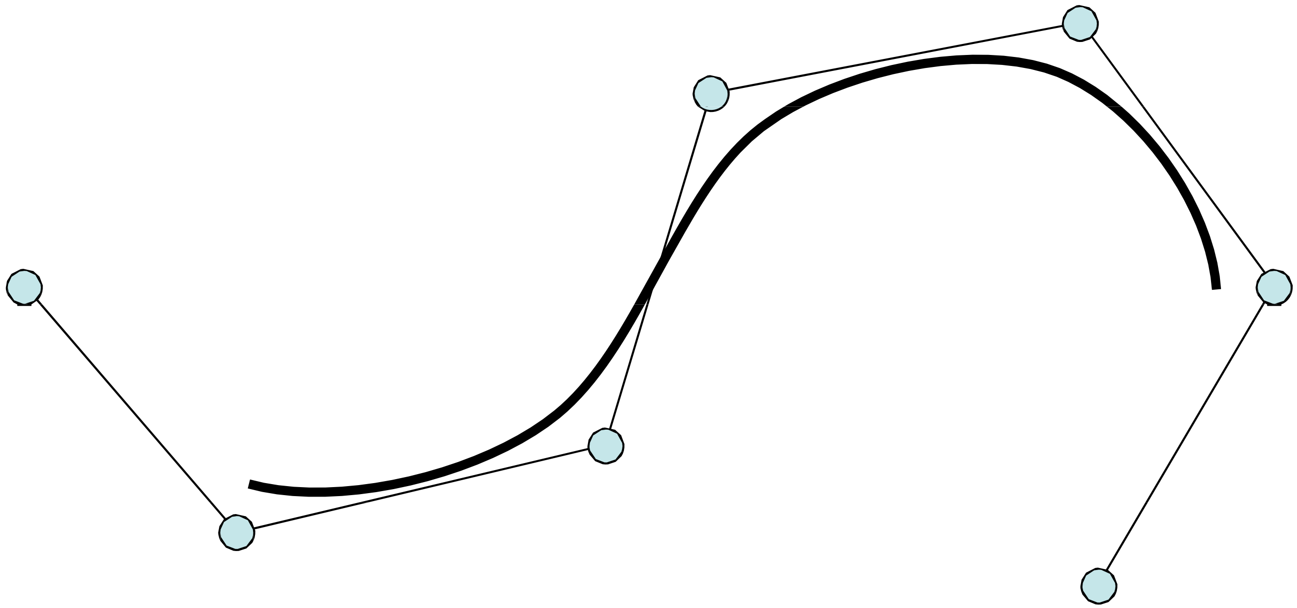
Approximating

...and so on...



Approximating

If the rule is designed carefully...



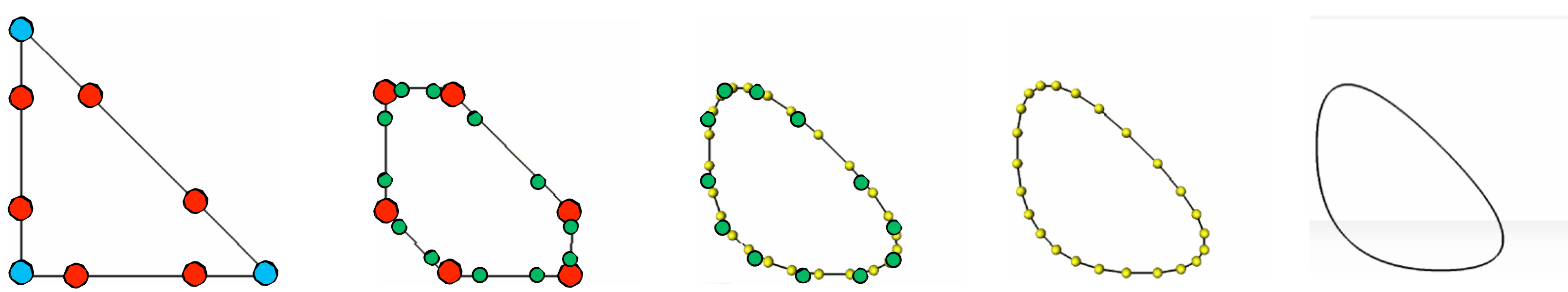
... the control polygons will converge to a smooth limit curve!

Equivalent to …

- Insert *single* new point at mid-edge
- *Filter* entire set of points.

Corner Cutting

- Subdivision rule:
  - Insert *two* new vertices at ¼ and ¾ of each edge
  - *Remove* the old vertices
  - Connect the new vertices
  - In the limit, generates a curve called a quadratic B-Spline



B-Spline Curves

- Piecewise polynomial of degree n

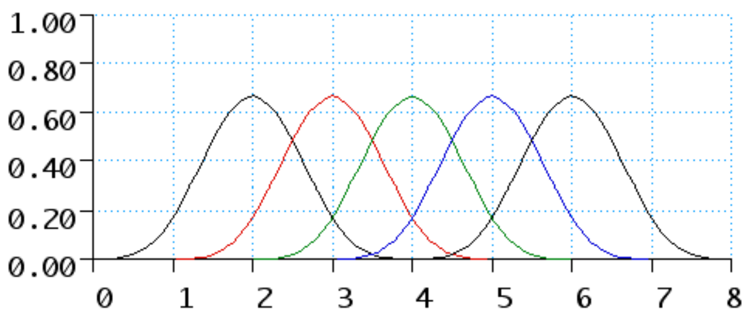
B-spline curve

control points

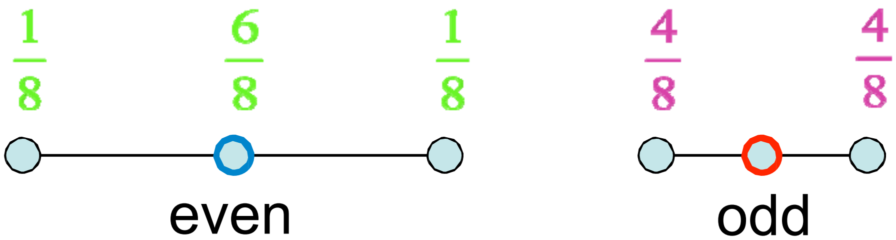
$$\mathbf{s}(u) = \sum_{i=0}^{k} \mathbf{d}_i N_i^n(u)$$

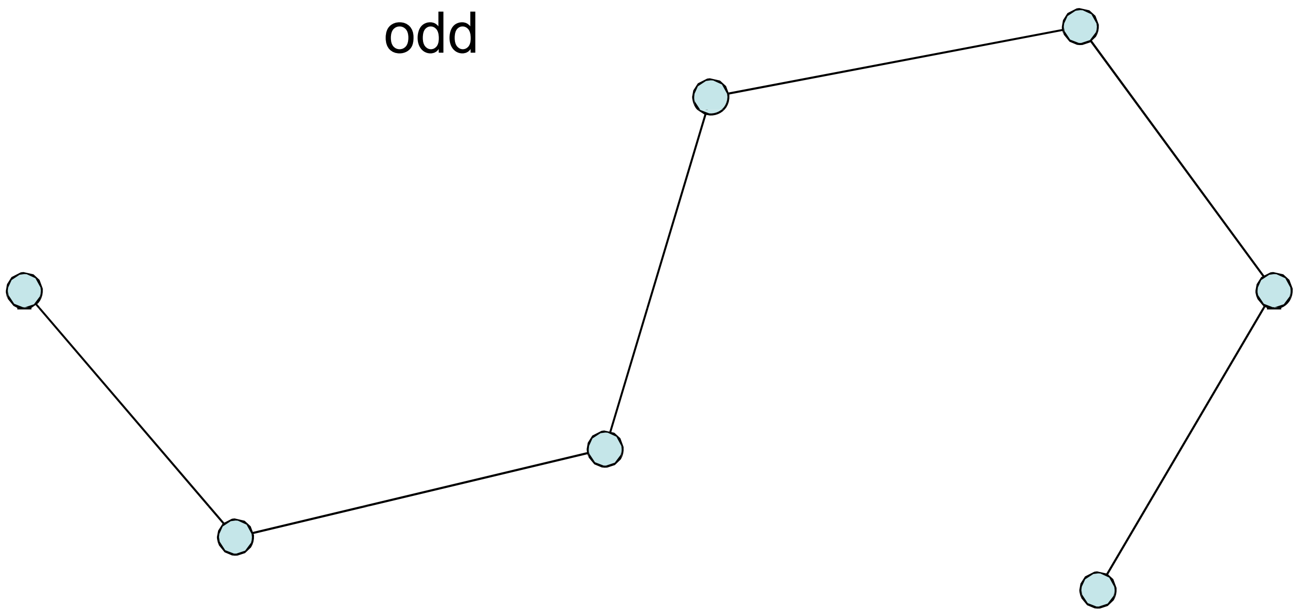
parameter value

basis functions

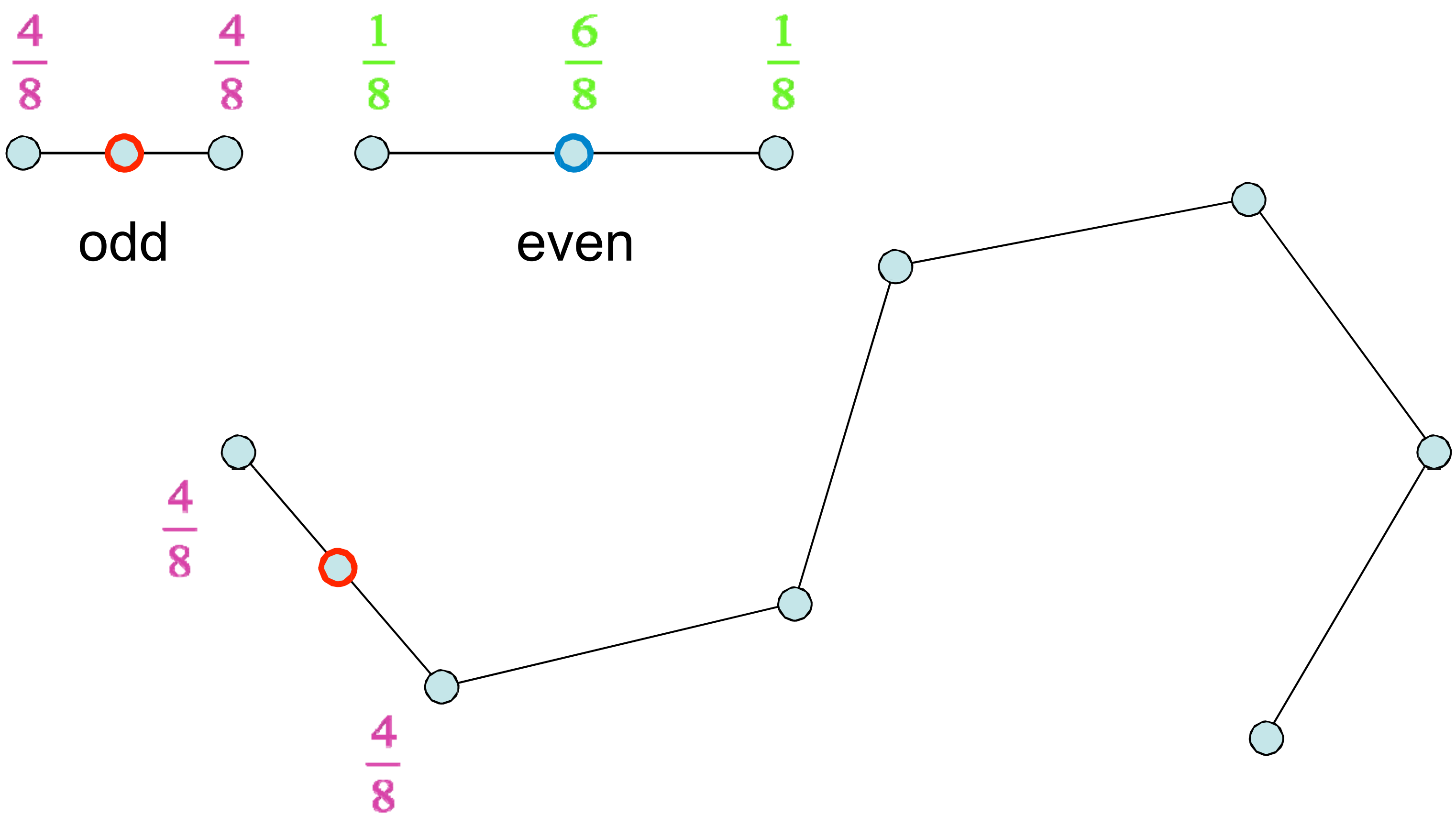


Cubic B-Spline

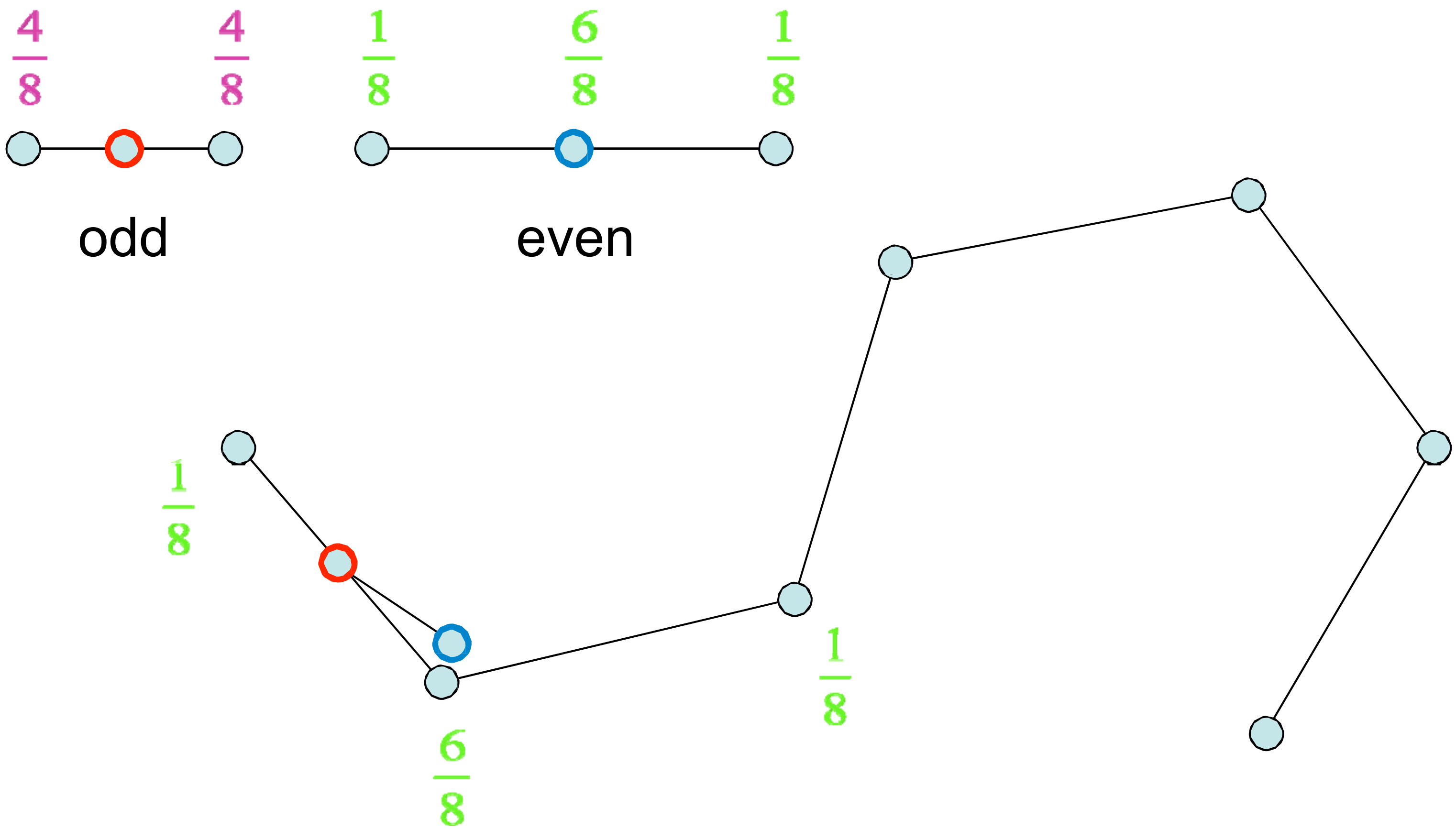




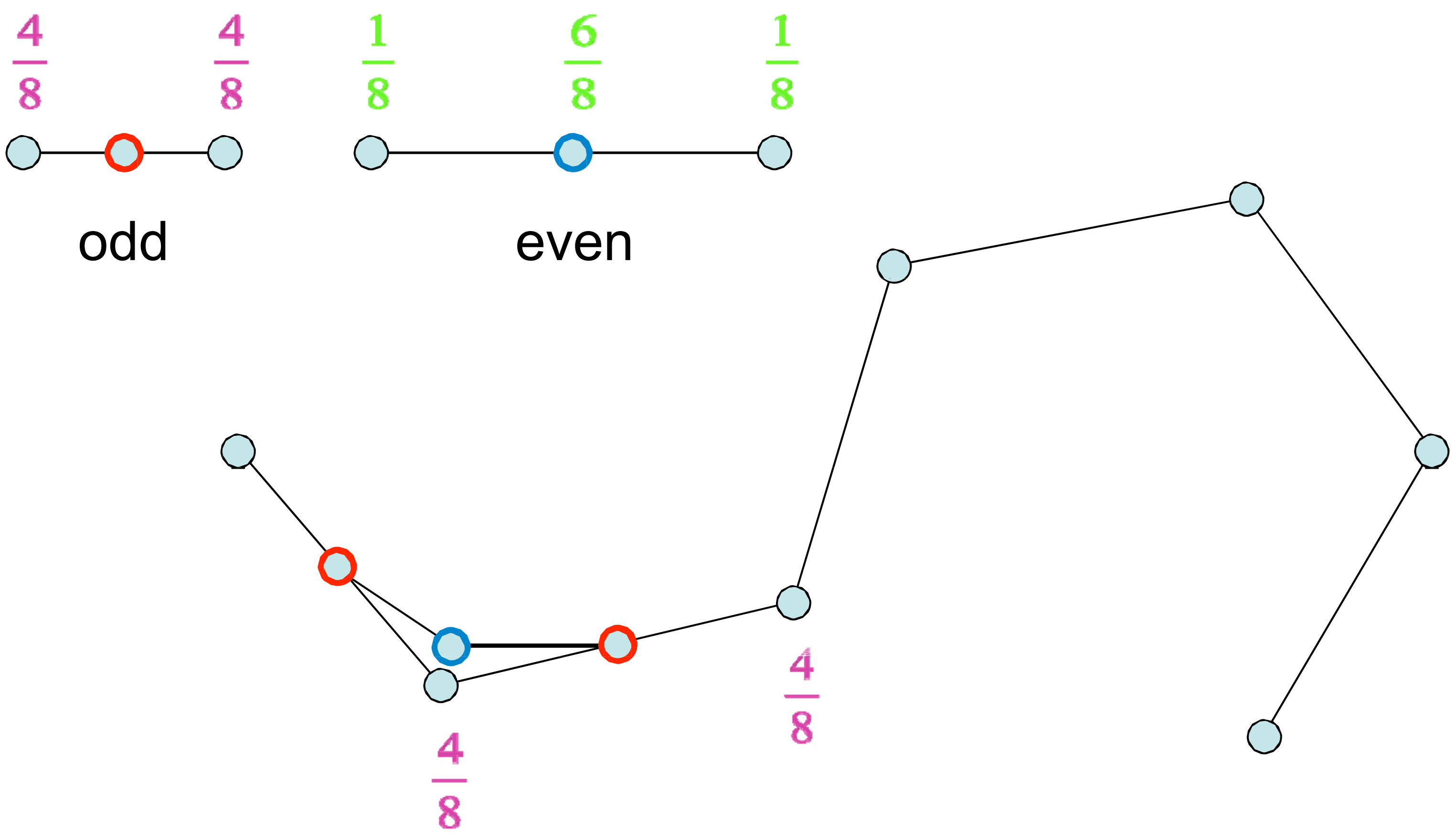
Cubic B-Spline



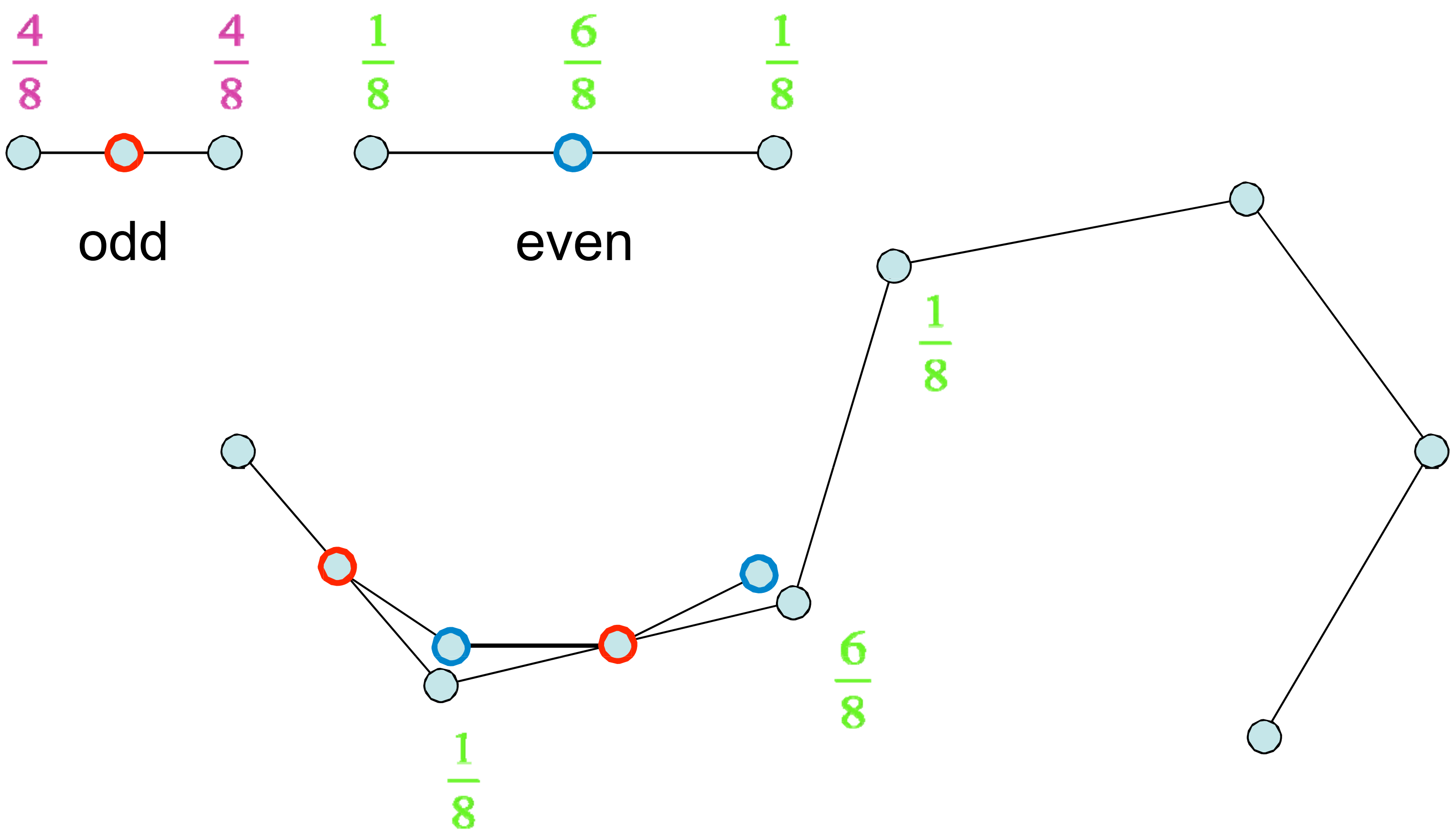
Cubic B-Spline



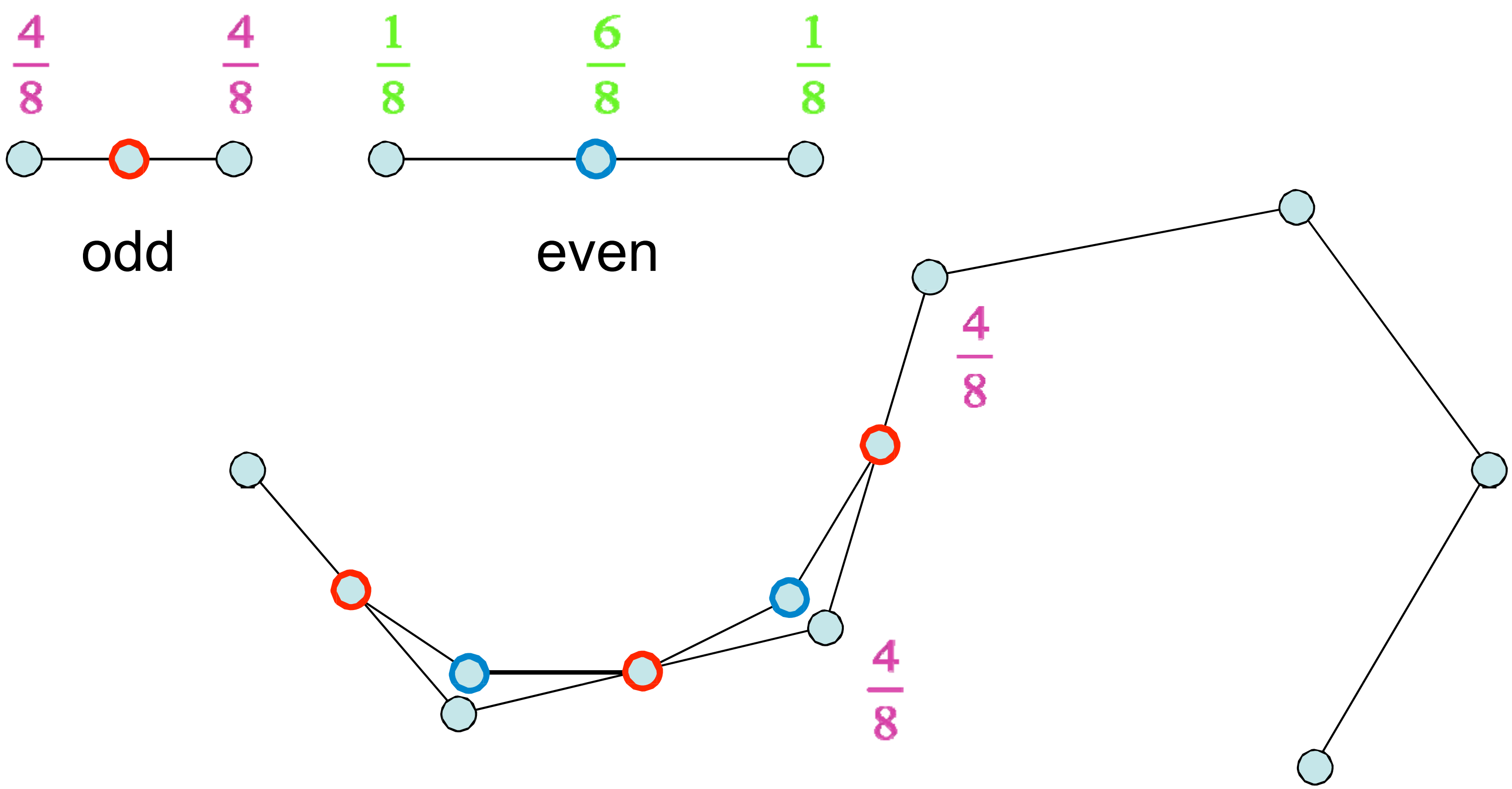
Cubic B-Spline



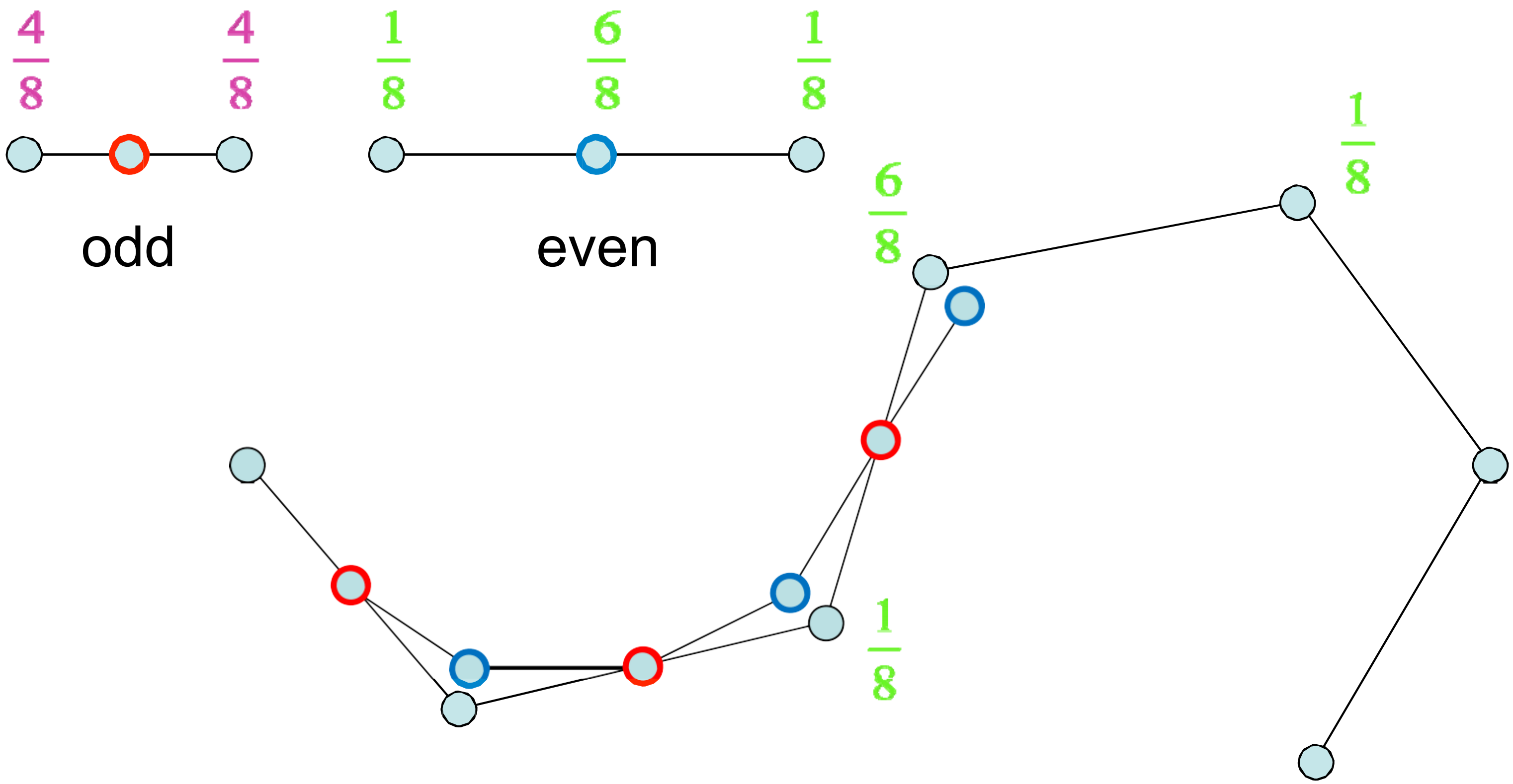
Cubic B-Spline



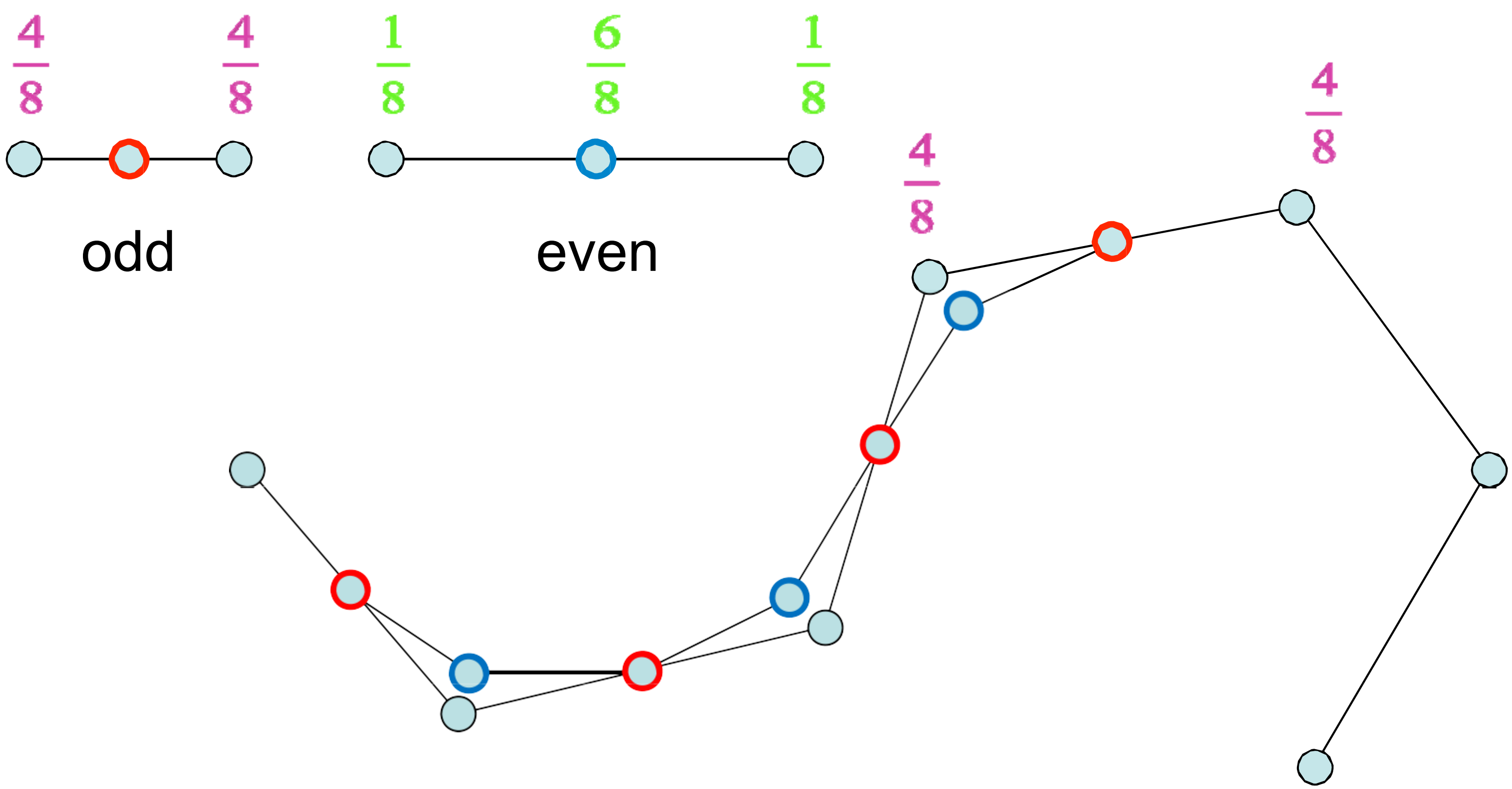
Cubic B-Spline



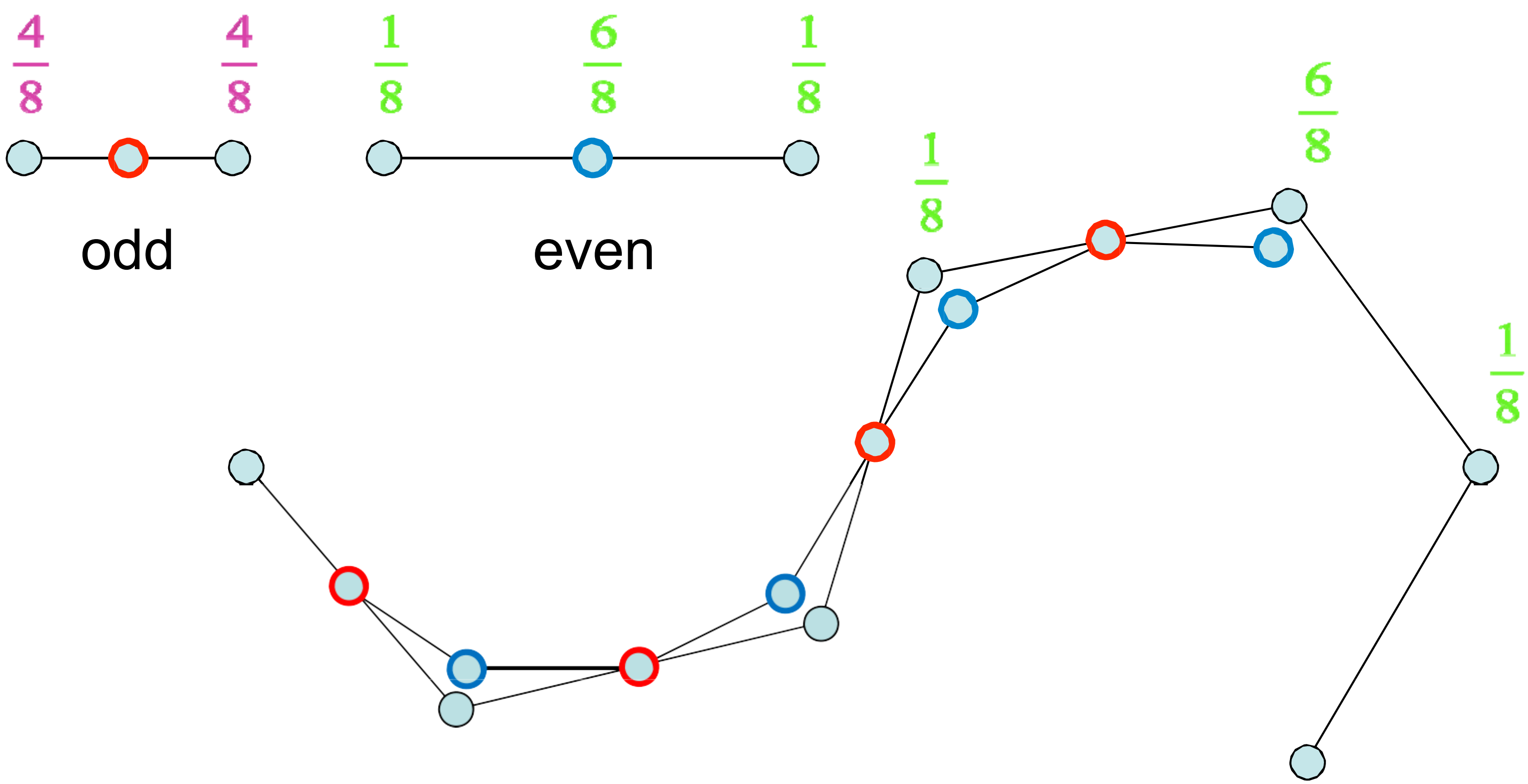
Cubic B-Spline



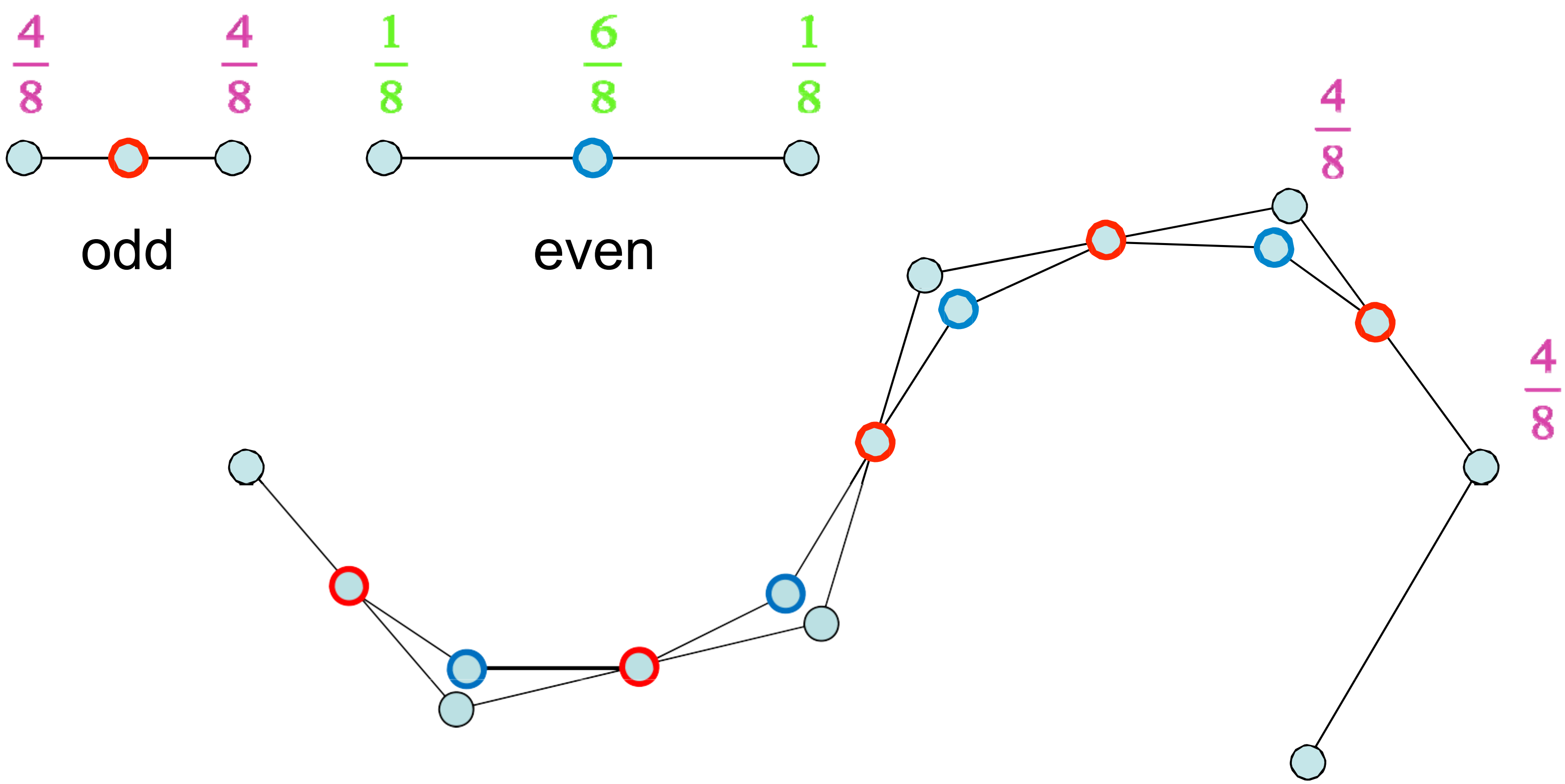
Cubic B-Spline



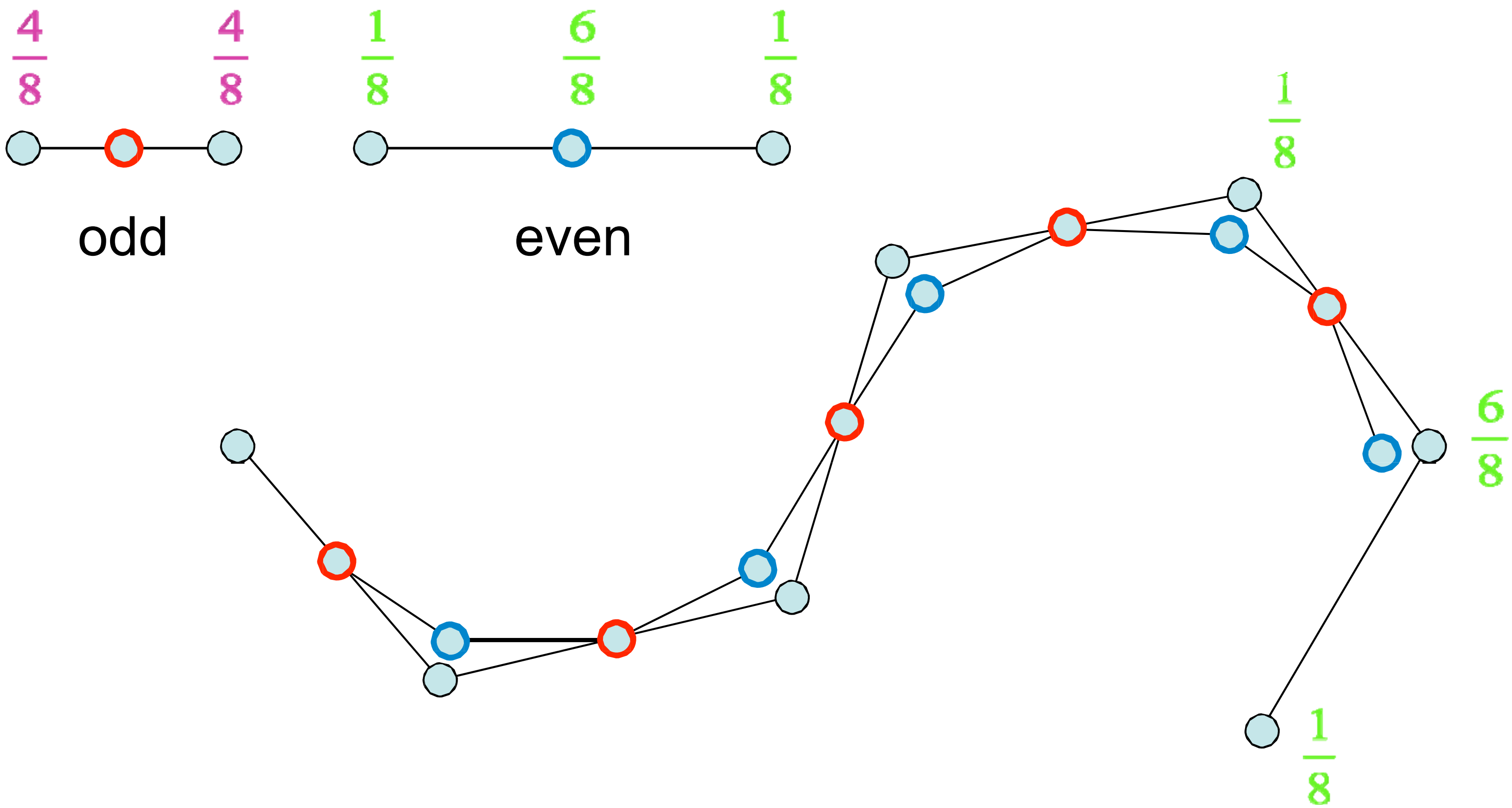
Cubic B-Spline



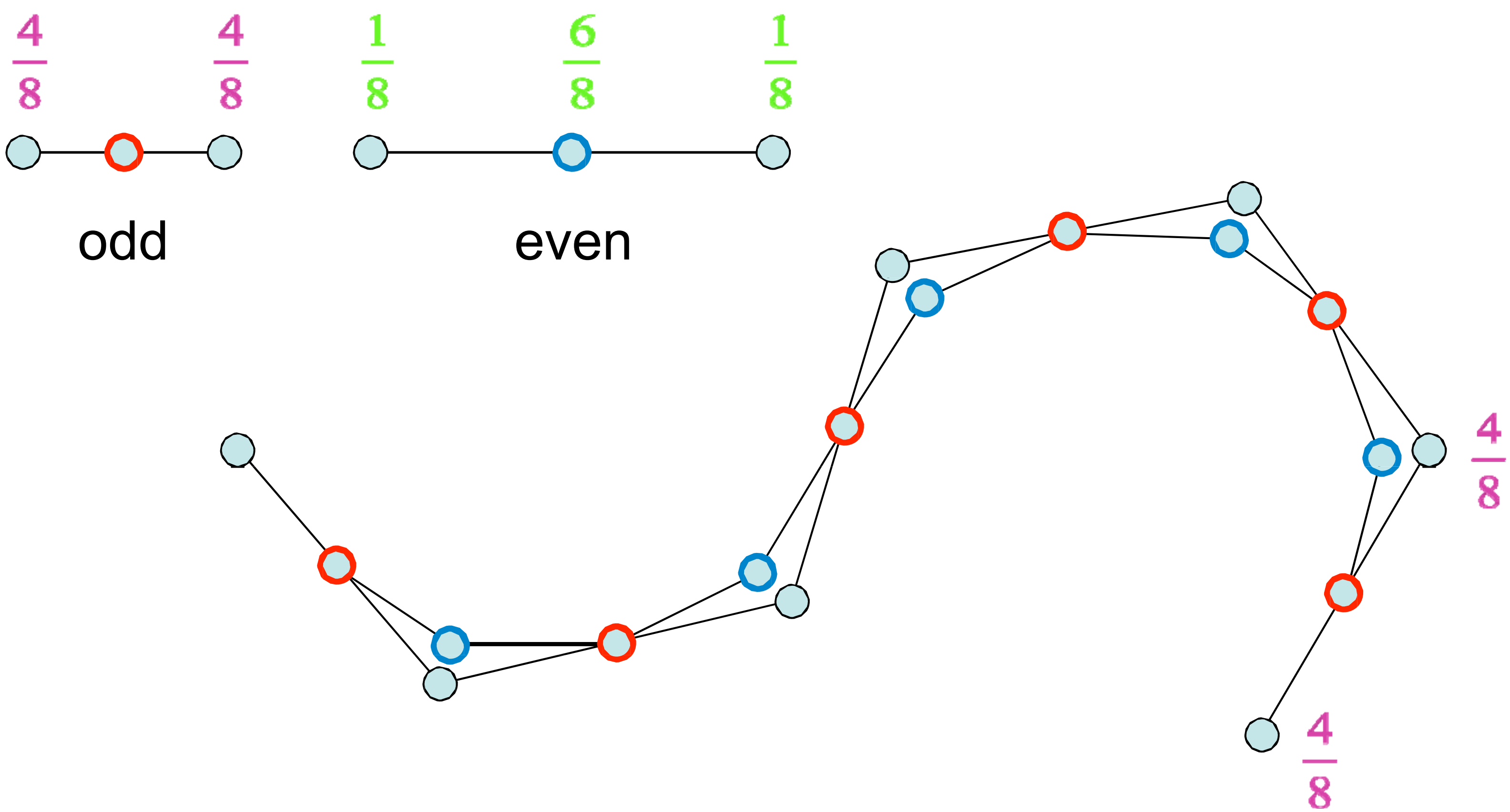
Cubic B-Spline



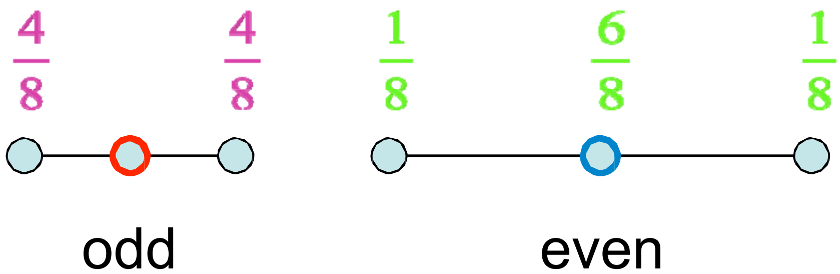
Cubic B-Spline

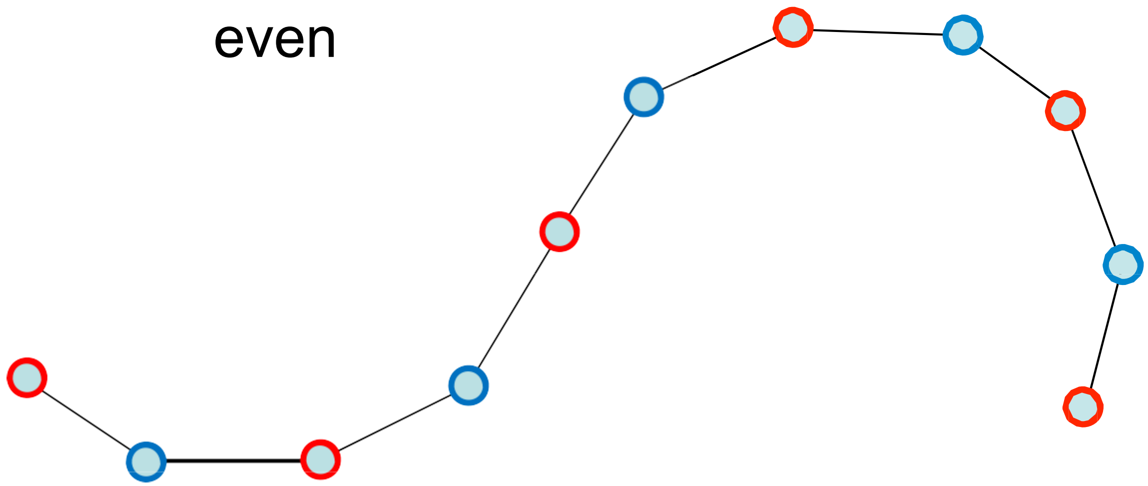


Cubic B-Spline

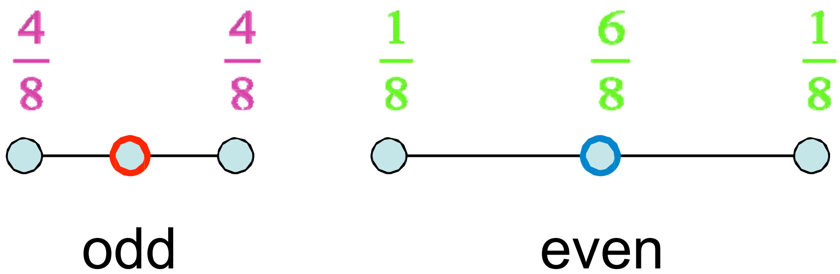


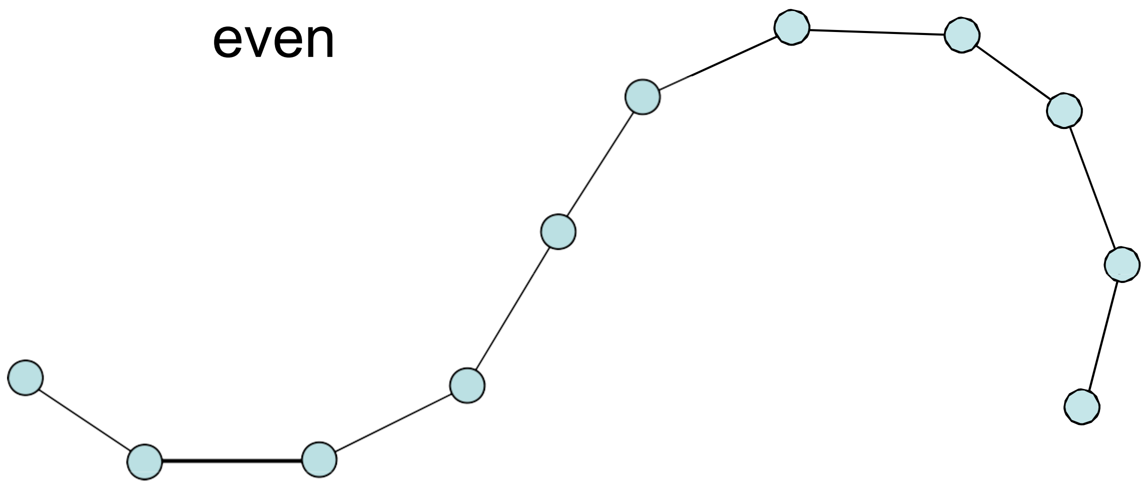
Cubic B-Spline





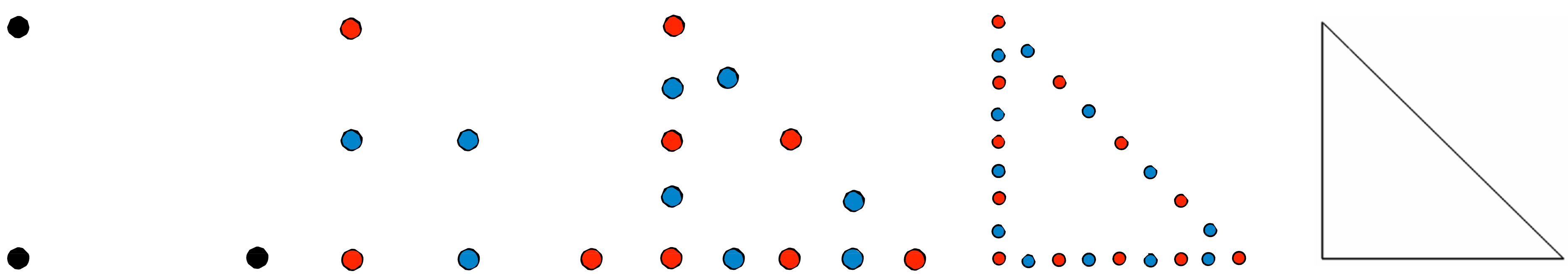
Cubic B-Spline





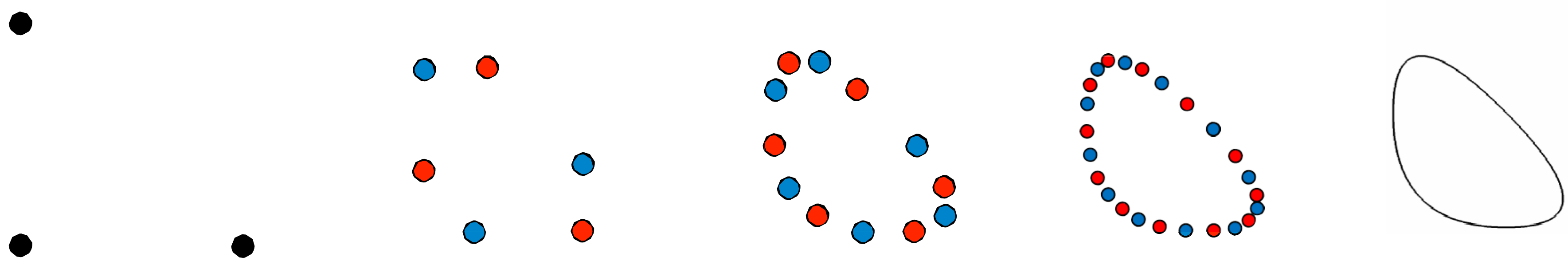
B-Spline Curves

- Distinguish between odd and even points
- **Linear B-spline**
  - Odd coefficients (1/2, 1/2)
  - Even coefficient (1)



B-Spline Curves

- **Quadratic B-Spline (Chaikin)**
  - Odd coefficients (¼, ¾)
  - Even coefficients (¾ , ¼)



- **Cubic B-Spline (Catmull-Clark)**
  - Odd coefficients (4/8, 4/8)
  - Even coefficients (1/8, 6/8, 1/8)

B-Spline Curves

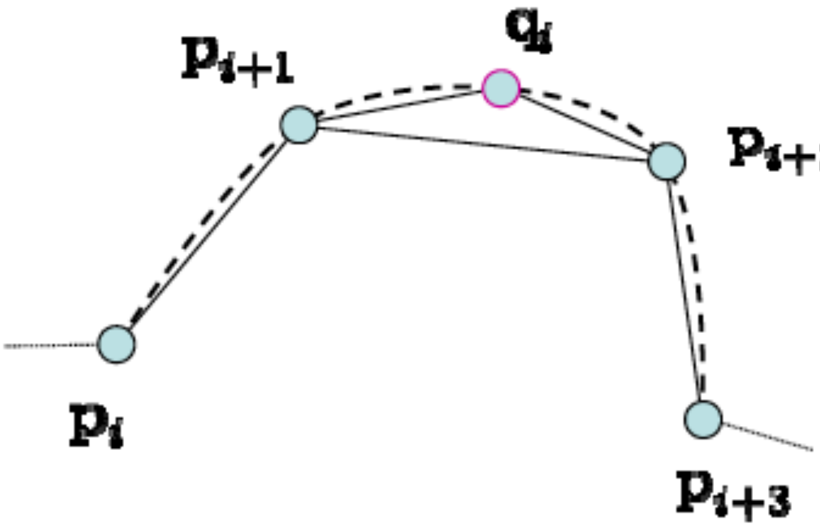
- Subdivision rules for control polygon

$$\mathbf{d}^0 \rightarrow \mathbf{d}^1 = S\mathbf{d}^0 \rightarrow \ldots \rightarrow \mathbf{d}^j = S\mathbf{d}^{j-1} = S^j\mathbf{d}^0$$

- Mask of size *n* yields C^{n-1} curve

Interpolating (4-point Scheme)

- Keep old vertices
- Generate new vertices by fitting cubic curve to old vertices
- C^1 continuous limit curve

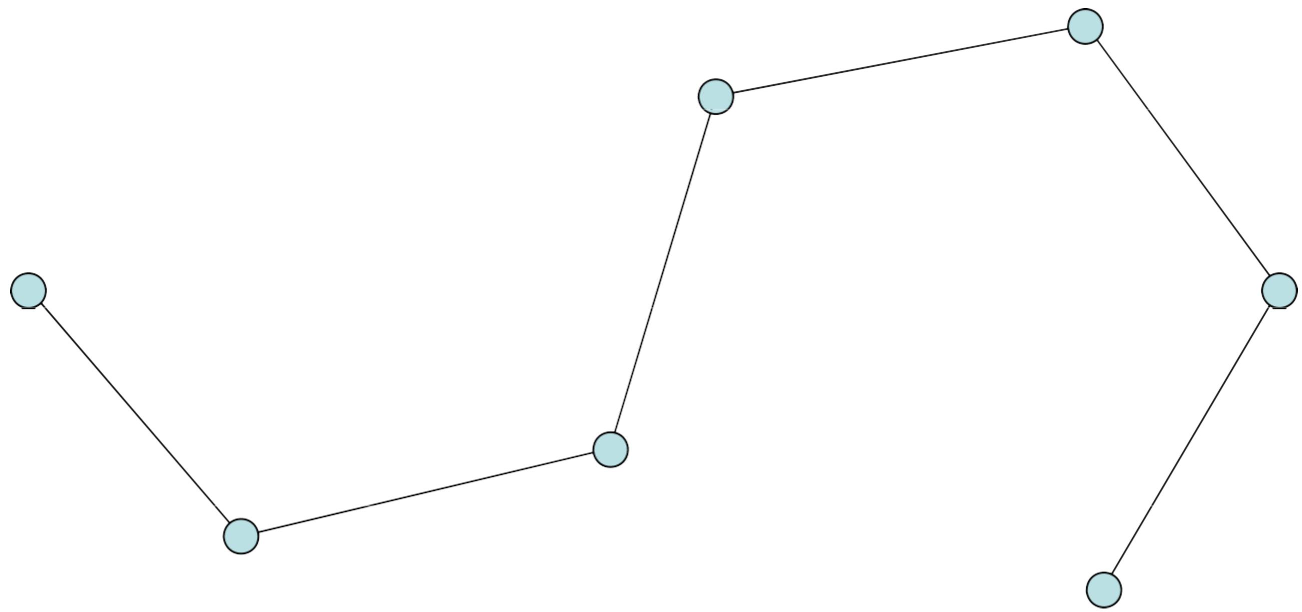


$$f(x) = ax^3 + bx^2 + cx + d$$

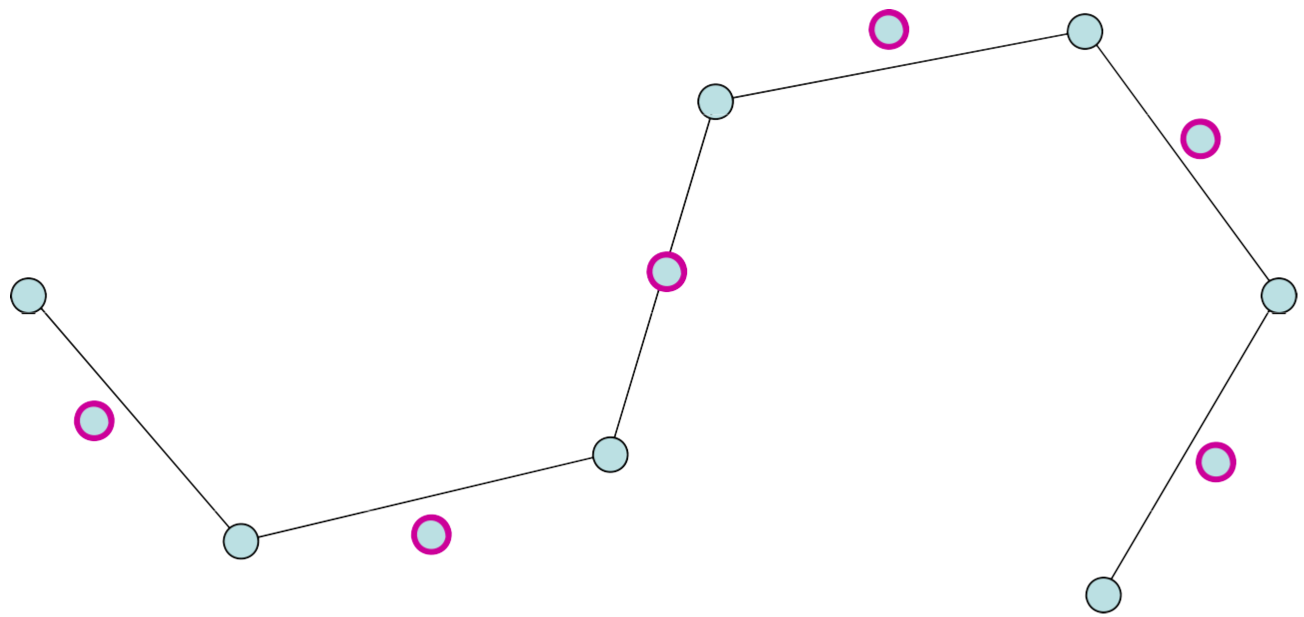
$$f(j) = \mathbf{p}_{i+j}, \quad j = 0, \ldots, 3$$

$$\begin{aligned} \mathbf{q}_i &= f(3/2) \\ &= \frac{1}{16}\left(-\mathbf{p}_i + 9\mathbf{p}_{i+1} + 9\mathbf{p}_{i+2} - \mathbf{p}_{i+3}\right) \end{aligned}$$

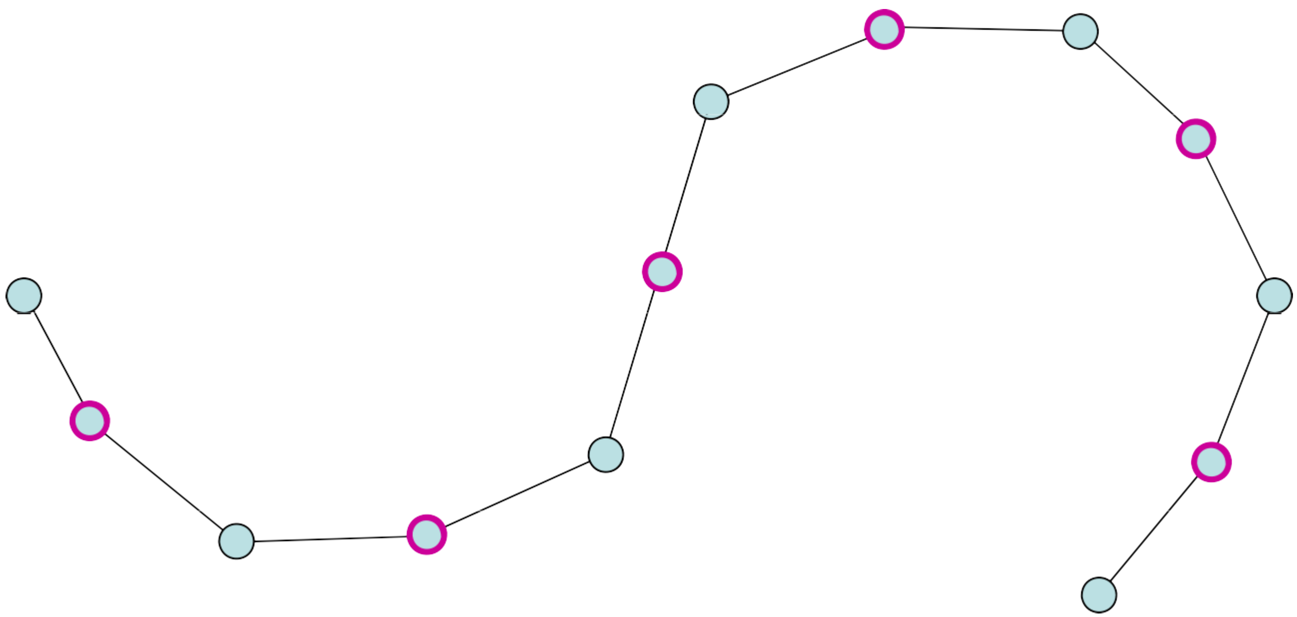
Interpolating



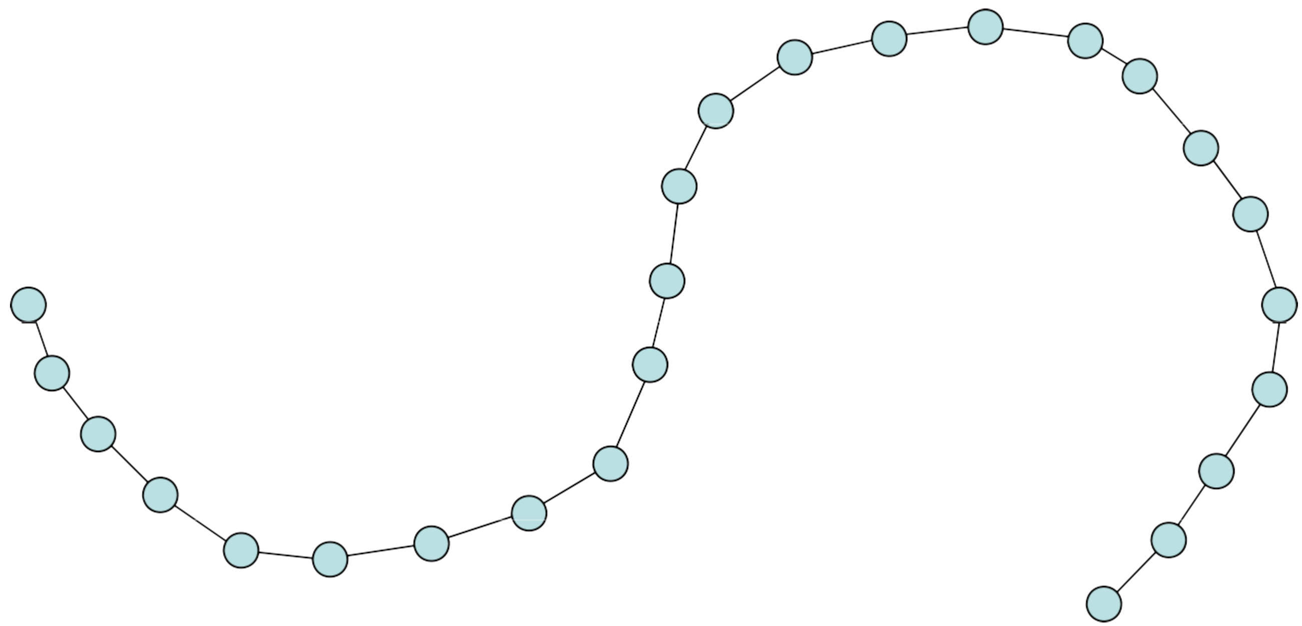
Interpolating



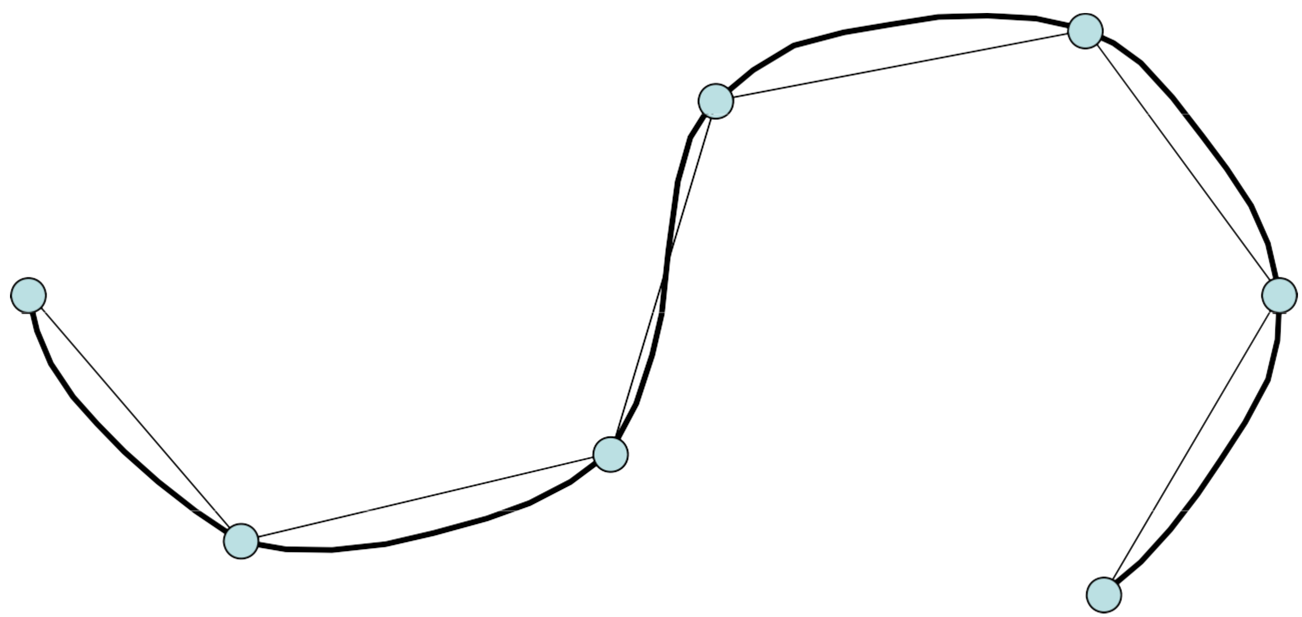
Interpolating



Interpolating

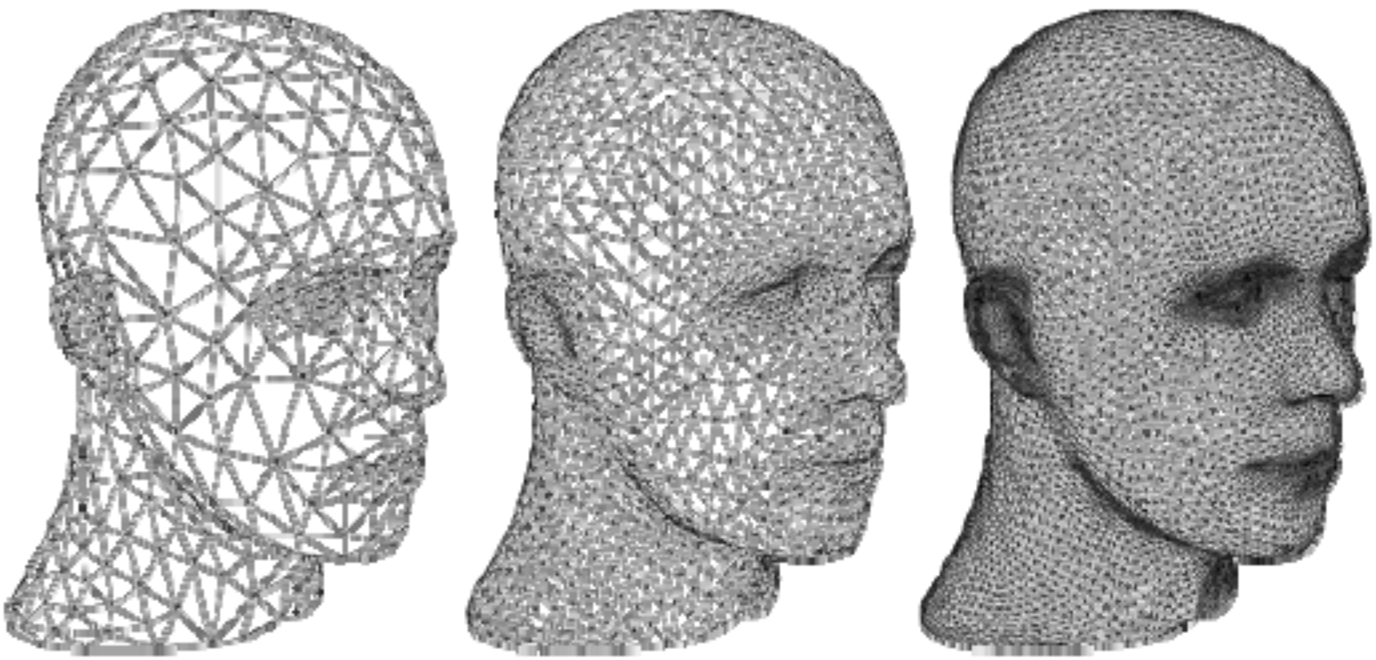


Interpolating



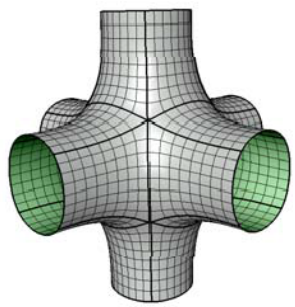
Subdivision Surfaces

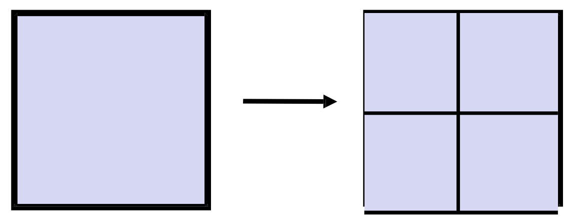
- No regular structure as for curves
  - Arbitrary number of edge-neighbors
  - Different subdivision rules for each valence



Subdivision Rules

- How the connectivity changes





- How the geometry changes
  - Old points
  - New points

Subdivision Zoo

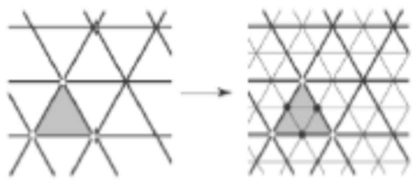
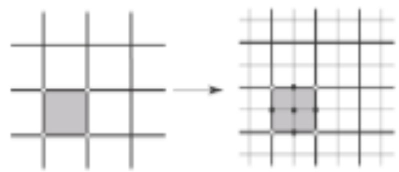
- Classification of subdivision schemes

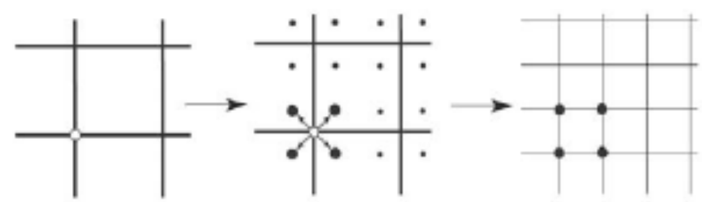
Primal	Faces are split into sub-faces
Dual	Vertices are split into multiple vertices

Approximating	Control points are not interpolated
Interpolating	Control points are interpolated

Subdivision Zoo

- Classification of subdivision schemes

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

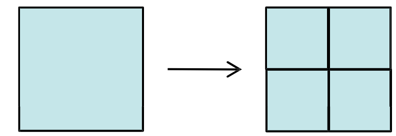
 Dual (vertex split)
Doo-Sabin, Midedge(C^1)
Biquartic (C^2)

Subdivision Zoo

- Classification of subdivision schemes

	Primal		Dual
	Triangles	**Rectangles**	
Approximating	Loop	**Catmull-Clark**	Doo-Sabin Midedge
Interpolating	Butterfly	Kobbelt	

Catmull-Clark Subdivision



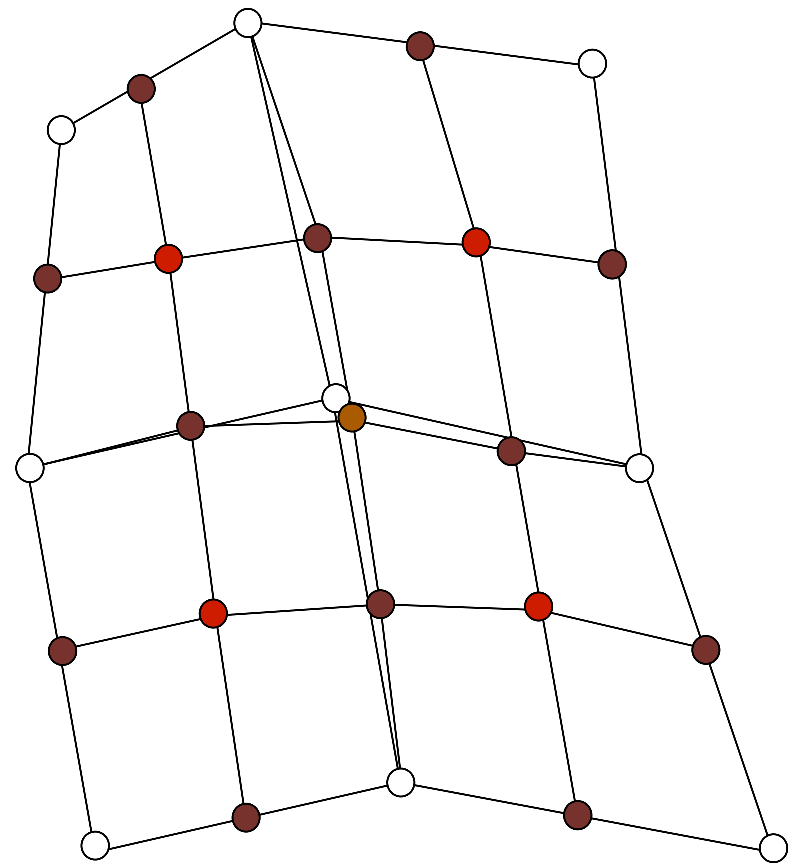
- Generalization of *bi-cubic B-Splines*
- Primal, approximation subdivision scheme
- Applied to *polygonal* meshes
- Generates G^2 *continuous* limit surfaces:
  - C^1 for the set of finite extraordinary points
    - Vertices with valence $\neq 4$
  - C^2 continuous everywhere else

Catmull Clark Subdivision

NOTE: valence = number of neighboring vertices

First subdivision generates quad mesh

Some vertices extraordinary (valence ≠ 4)



<u>Rules</u>

Face vertex = average of face's vertices

Edge vertex = average of edge's two vertices & adjacent face's two vertices

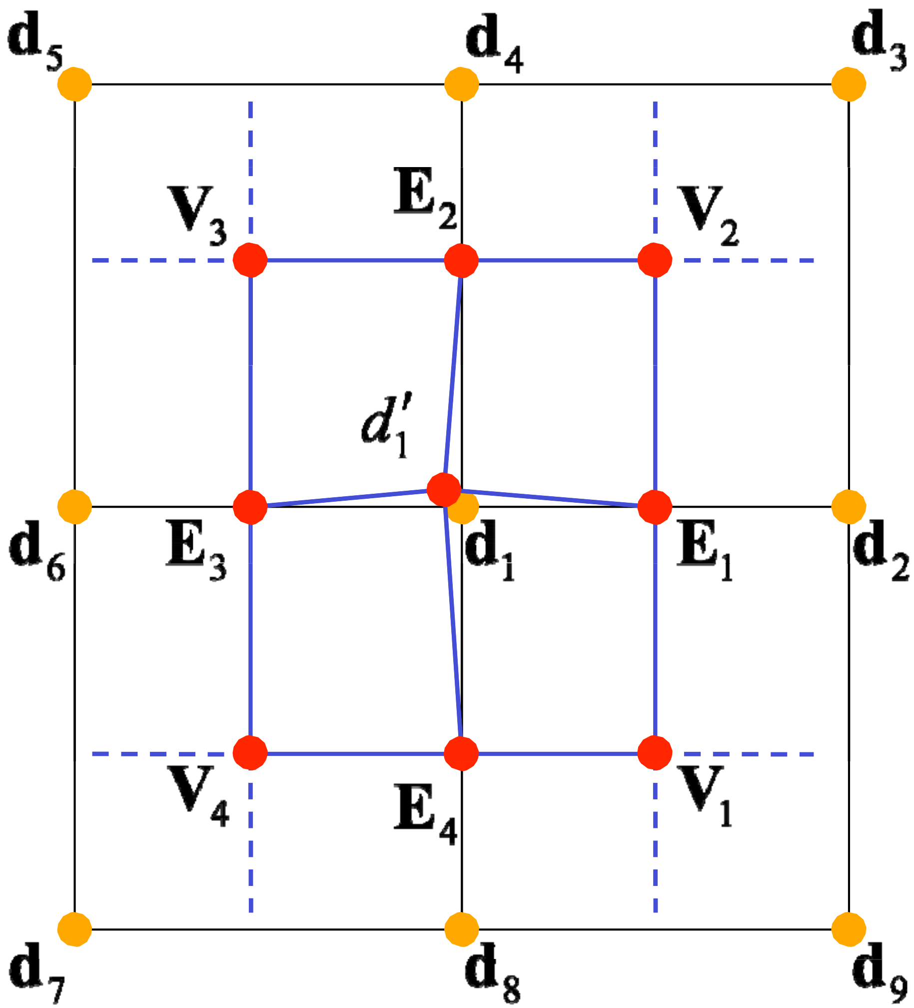
New vertex position = (1/valence) x **sum** of…

- Average of neighboring face points
- 2 x average of neighboring edge points
- (valence – 3) x original vertex position

Boundary edge points set to edge midpoints

Boundary vertices stay put

Catmull-Clark Subdivision



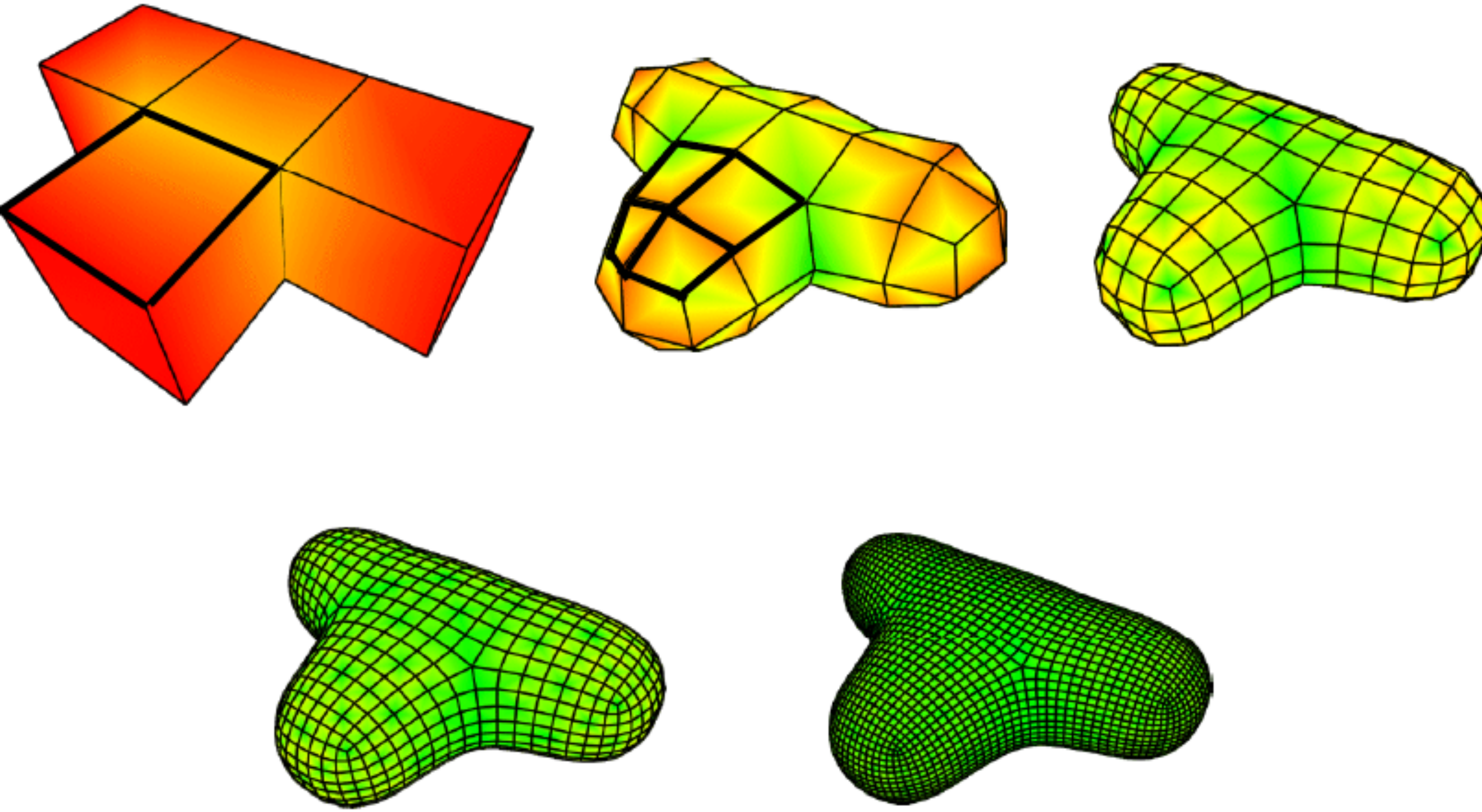
$$\mathbf{V}_2 = \frac{1}{n} \times \sum_{j=1}^{n} \mathbf{d}_j$$

$$\mathbf{E}_i = \frac{1}{4}\left(\mathbf{d}_1 + \mathbf{d}_{2i} + \mathbf{V}_i + \mathbf{V}_{i+1}\right)$$

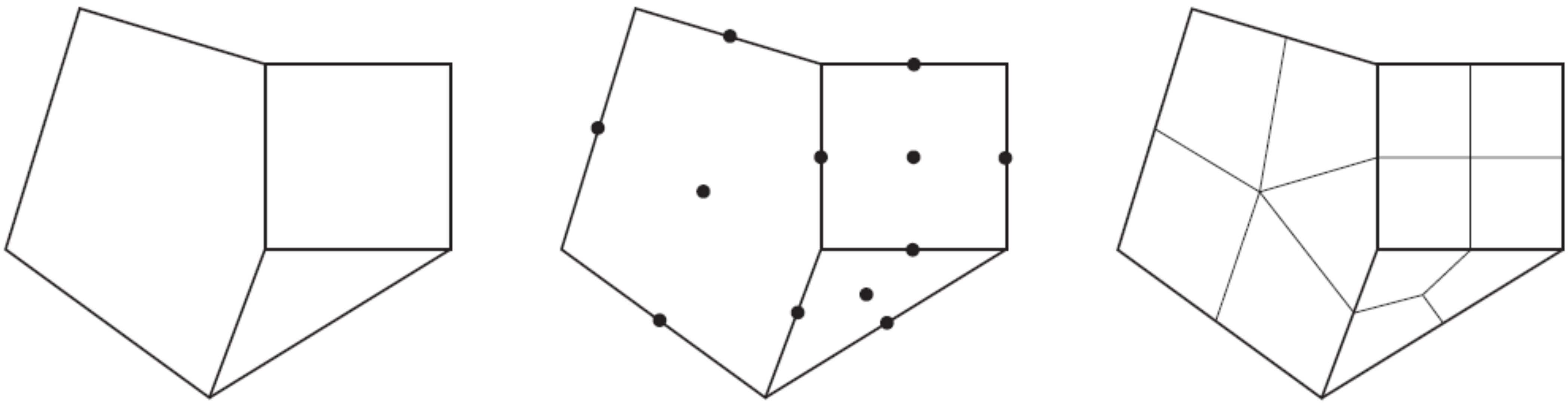
$$\mathbf{d}_1' = \frac{(n-3)}{n}\mathbf{d}_1 + \frac{2}{n}\mathbf{R} + \frac{1}{n}\mathbf{S}$$

$$\mathbf{R} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{E}_i \quad \mathbf{S} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{V}_i$$

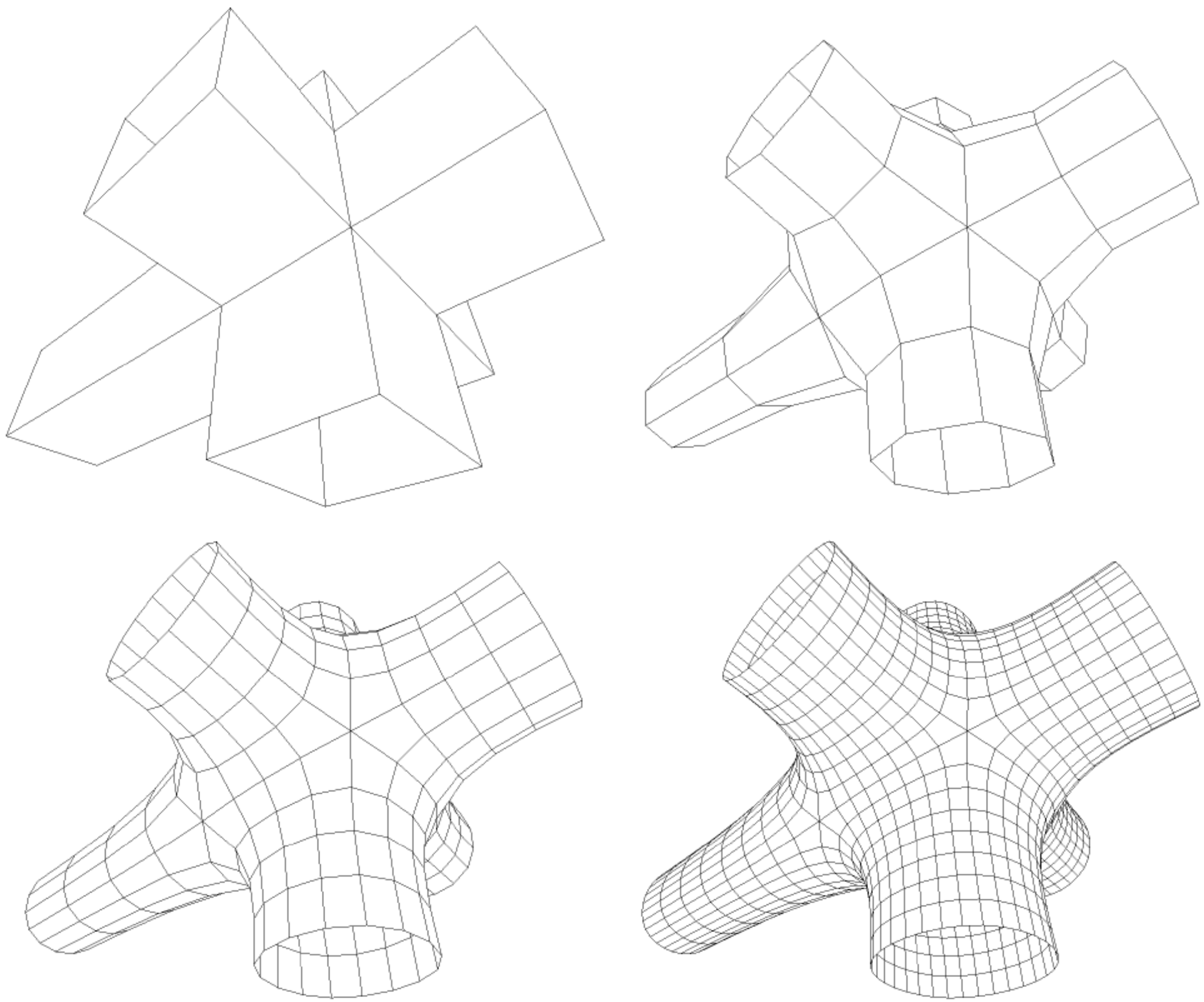
Catmull-Clark Subdivision



Catmull-Clark Subdivision



Catmull-Clark Subdivision



Implementation

Face vertex = average of face's vertices
Edge vertex = average of edge's two vertices
& adjacent face's two vertices
New vertex position = (1/valence) x sum of…
Average of neighboring face points
2 x average of neighboring edge points
(valence – 3) x original vertex position

Face vertex

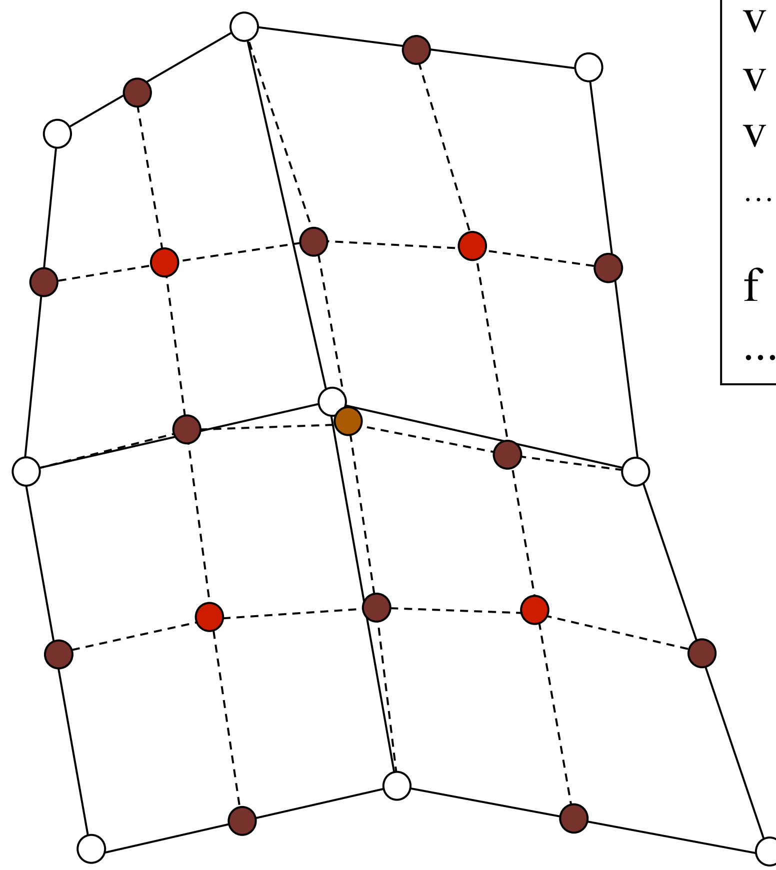
For each face
add vertex at its centroid

Edge vertex

How do we find each edge?

New vertex position

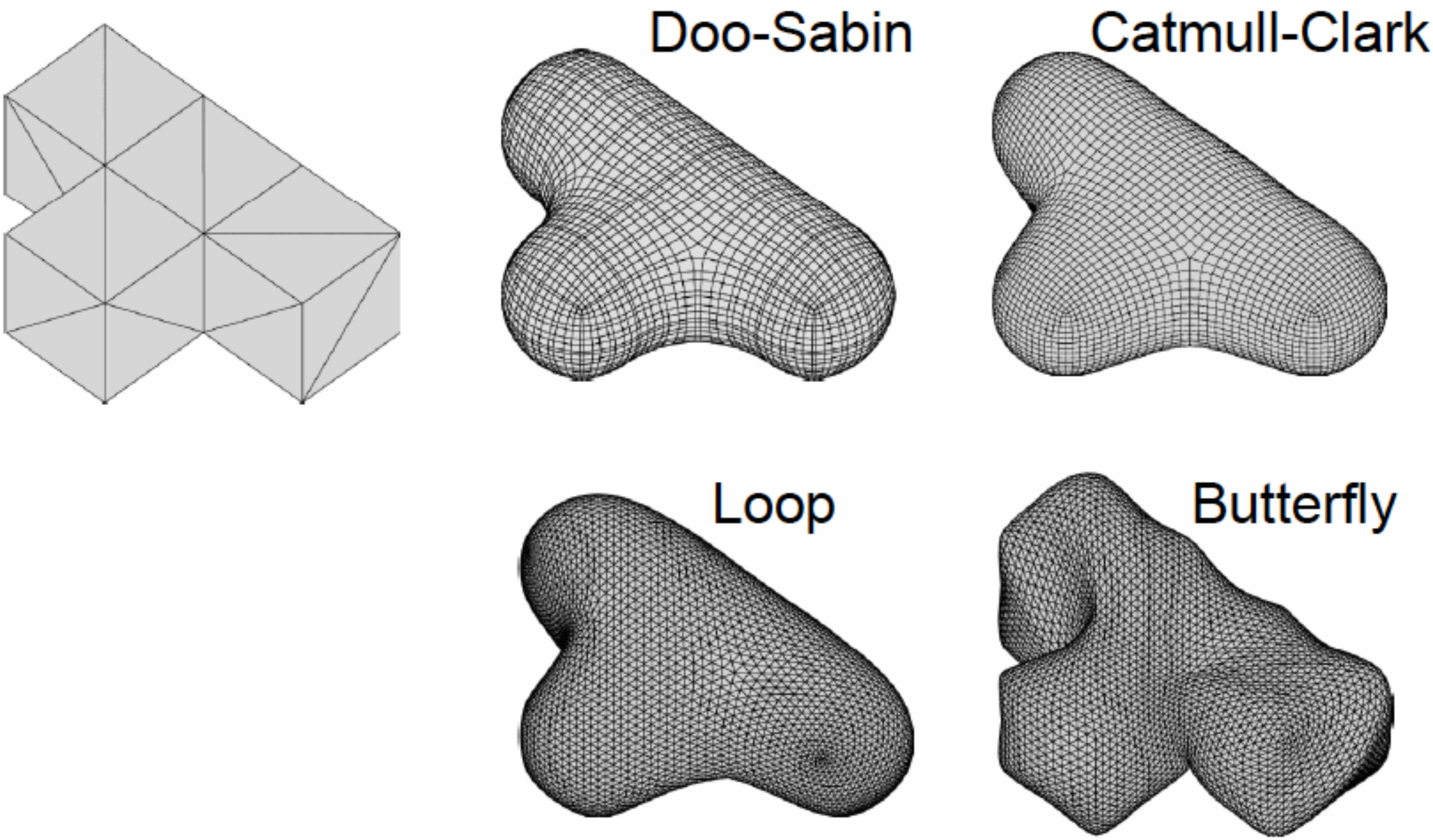
For a given vertex
how do we find neighboring
faces and edges?



v x_1 y_1 z_1
v x_2 y_2 z_2
v x_3 y_3 z_3
v x_4 y_4 z_4
…

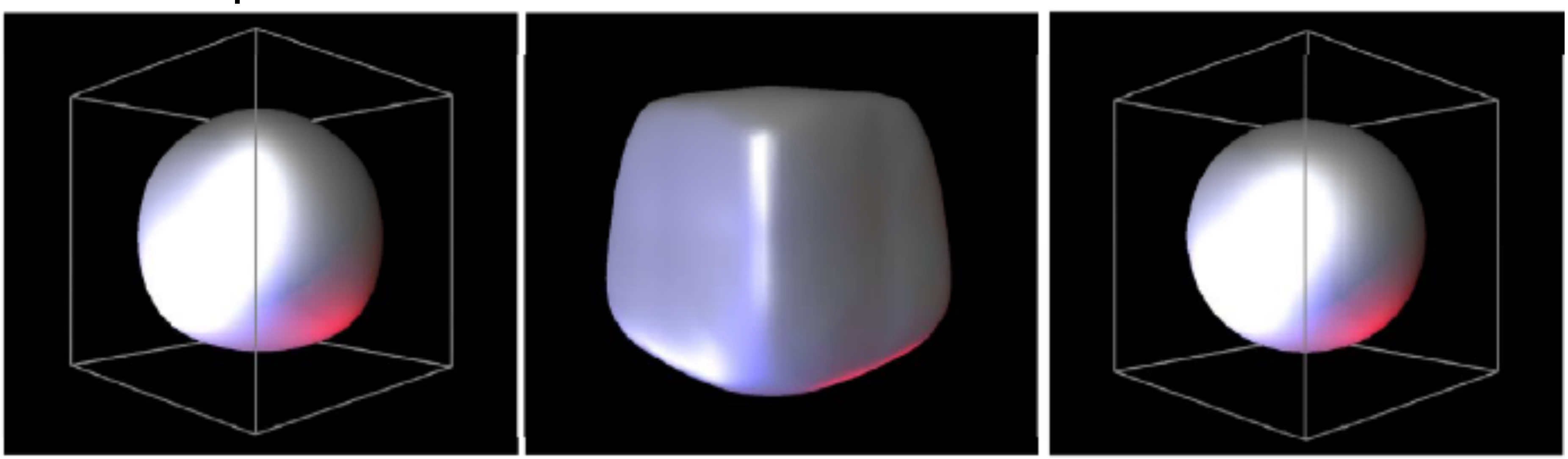
f 1 2 3 4
…

Comparison



Comparison

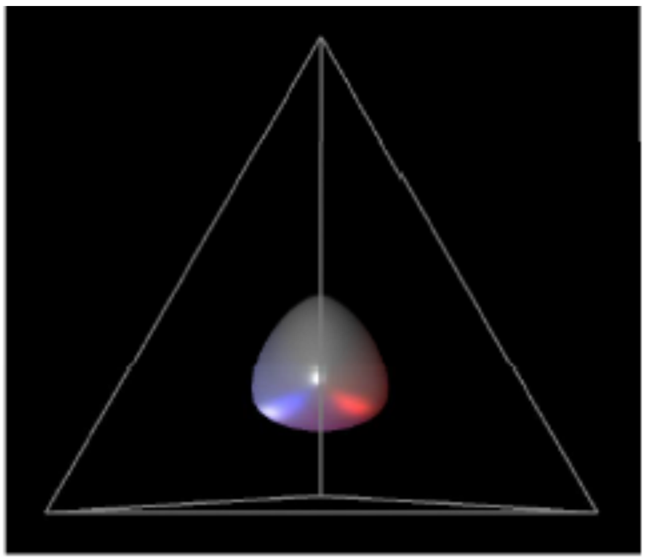
- Subdividing a cube
  - Loop result is assymetric, because cube was triangulated first
  - Both Loop and Catmull-Clark are better then Butterfly (C^2 vs. C^1)
  - Interpolation vs. smoothness



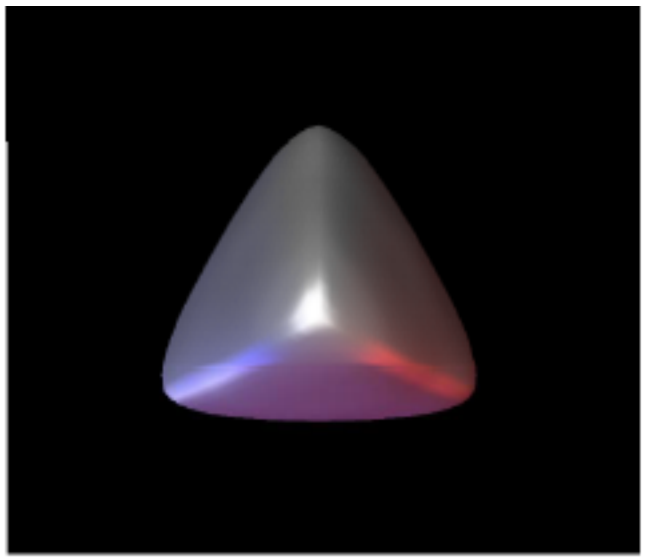
Loop *Butterfly* *Catmull-Clark*

Comparison

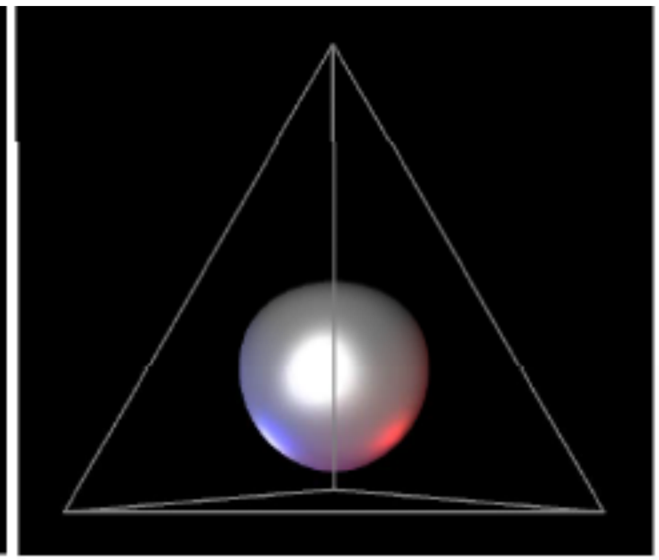
- Subdividing a tetrahedron
  - Same insights
  - Severe shrinking for approximating schemes



Loop



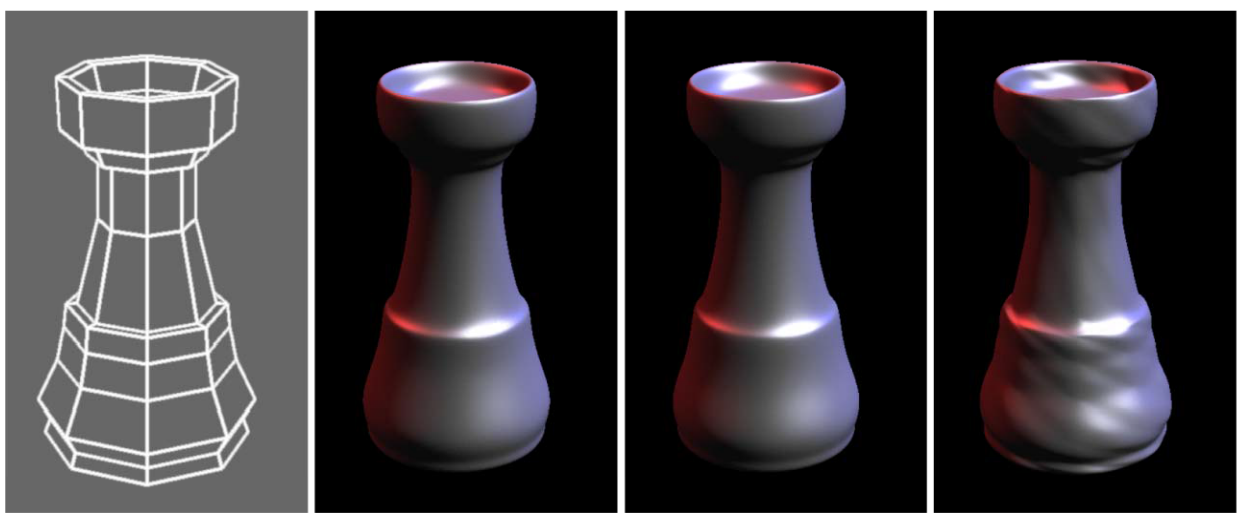
Butterfly



Catmull-Clark

So Who Wins?

- Loop and Catmull-Clark best when interpolation is not required
- Loop best for triangular meshes
- Catmull-Clark best for quad meshes
  - Don’t triangulate and then use Catmull-Clark



Initial mesh *Loop* *Catmull-Clark* *Catmull-Clark,after triangulation*