

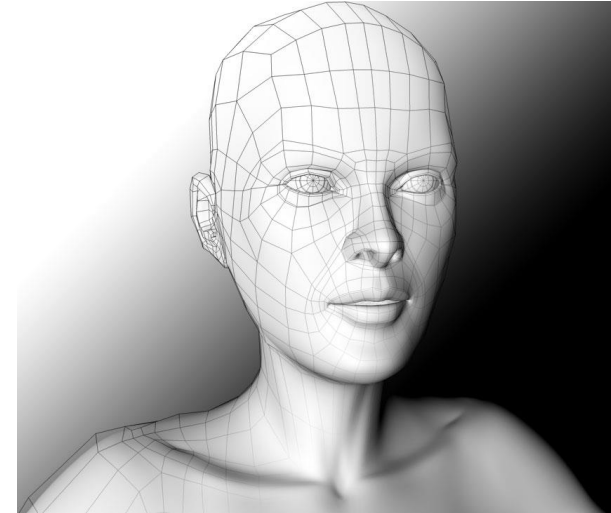
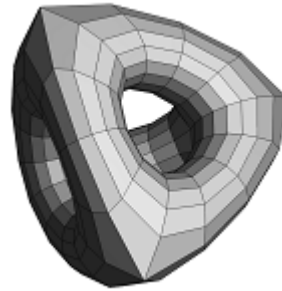
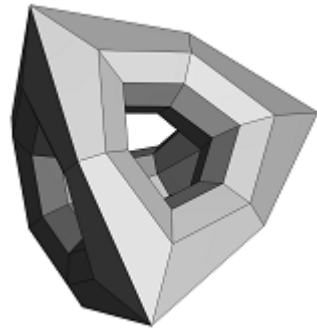
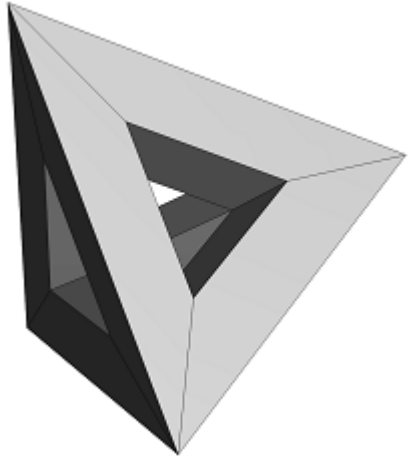
# Introduction to Computer Graphics

## – Modeling (2) –

April 21, 2016

Kenshi Takayama

# Subdivision surfaces

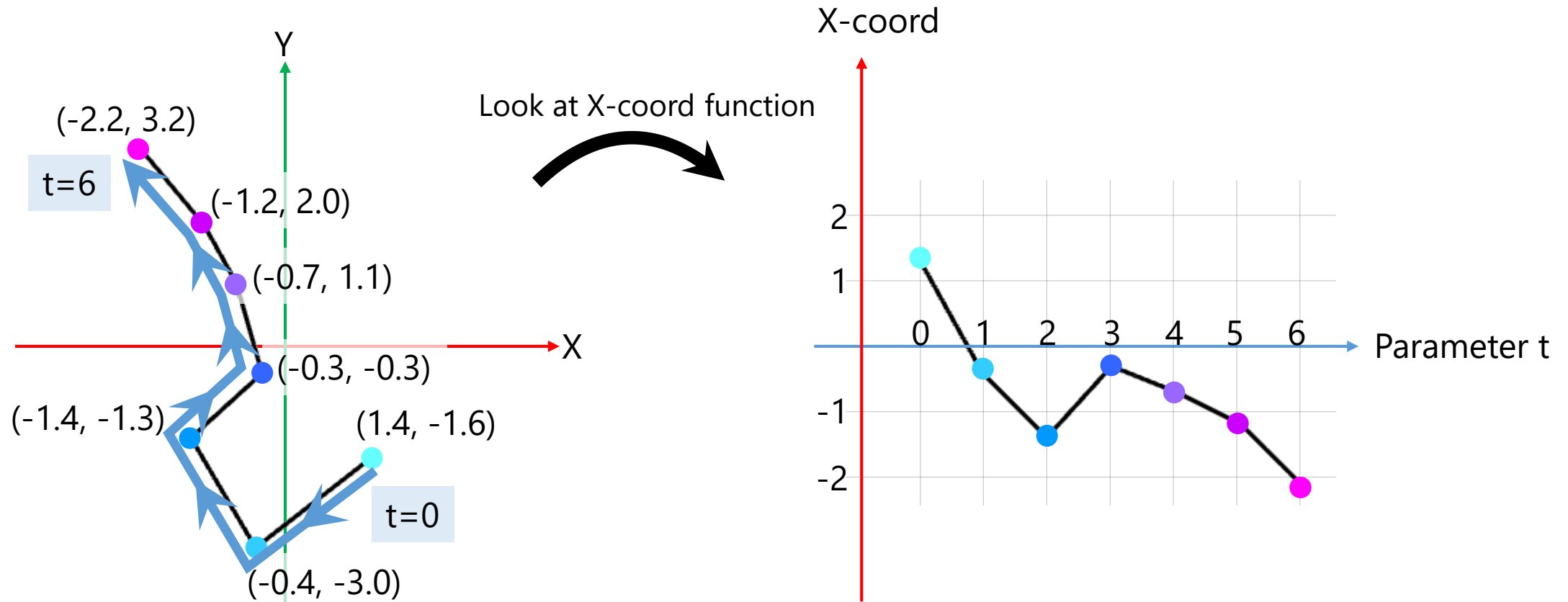


First, we'll look at its theoretical basis:

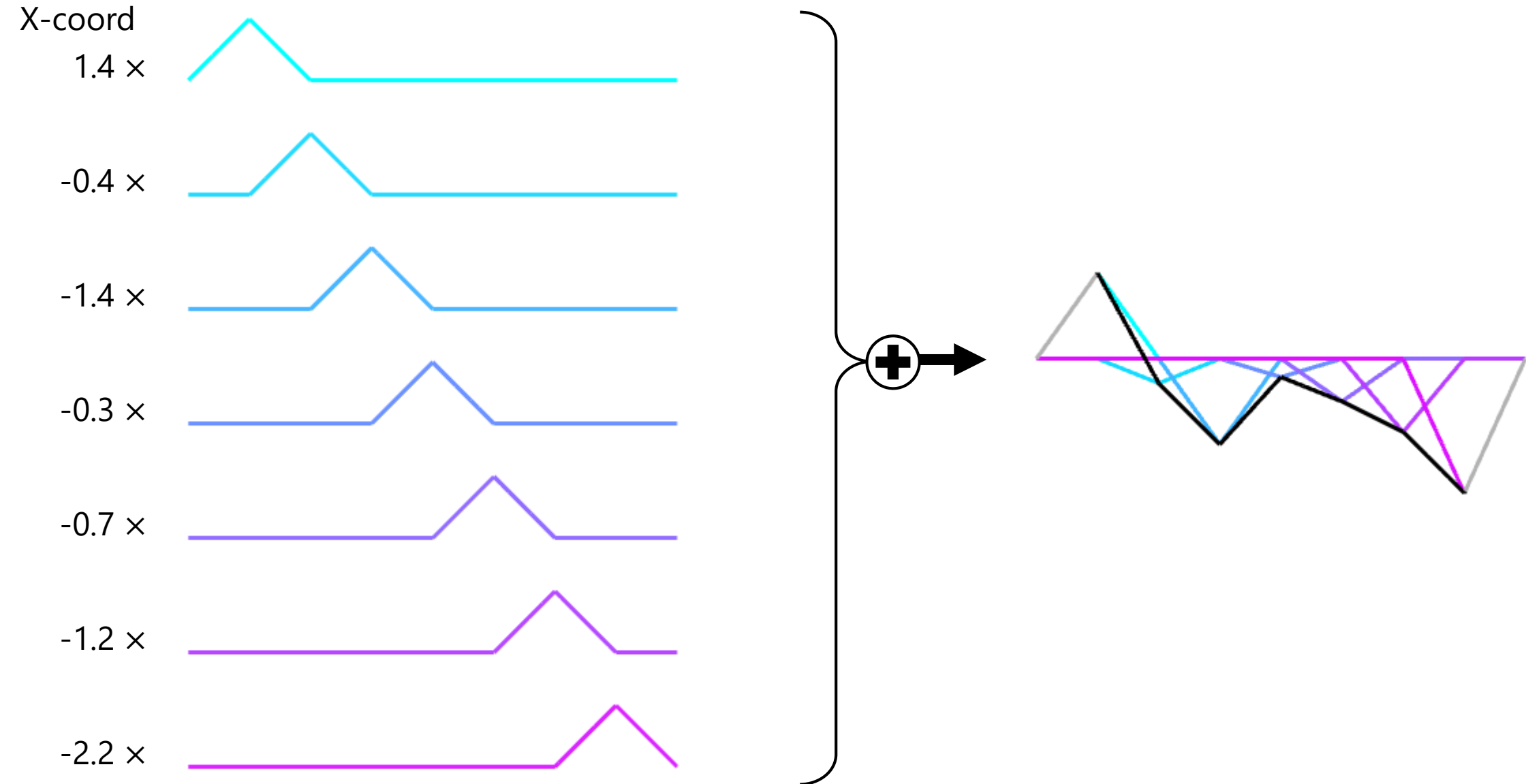
## B-Spline curves

Basis

# Example: 2D polyline represented as function



# Representing polyline using linear **basis**



# de Boor's n-th order basis

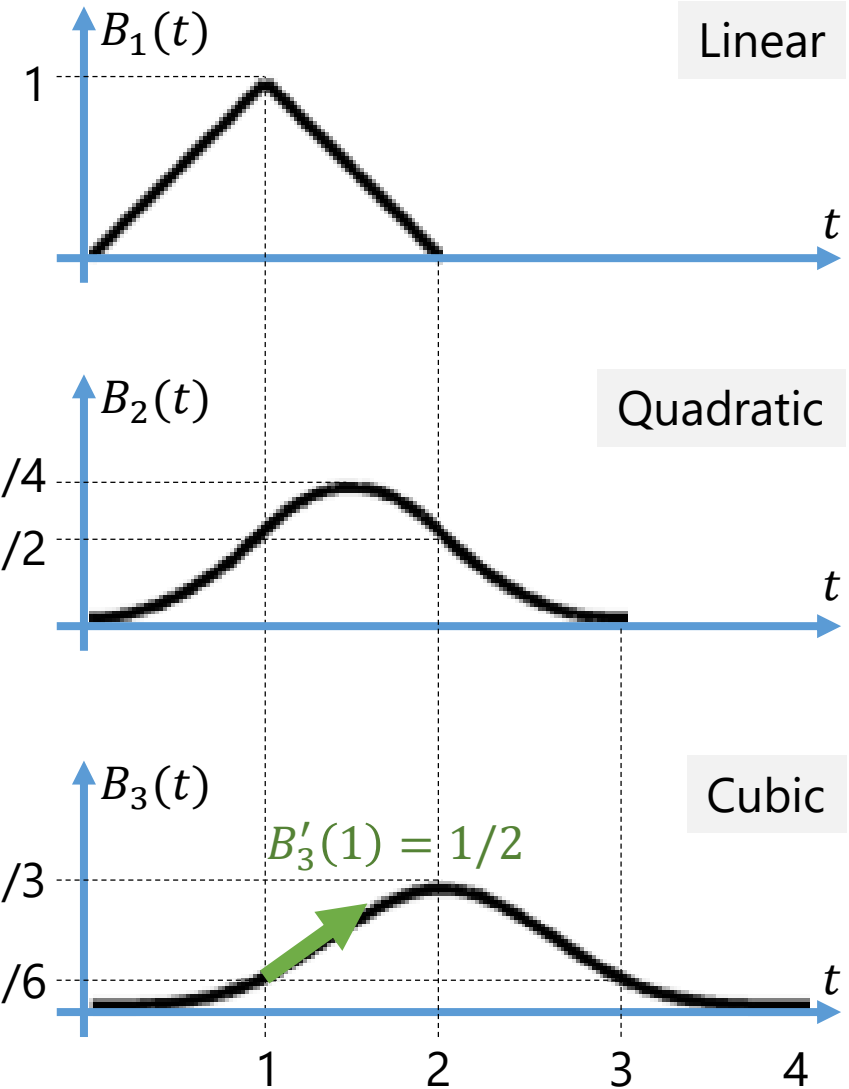
- Recursively defined:

- $B_0(t) = \begin{cases} 1, & 0 \leq t < 1 \\ 0, & \text{otherwise} \end{cases}$

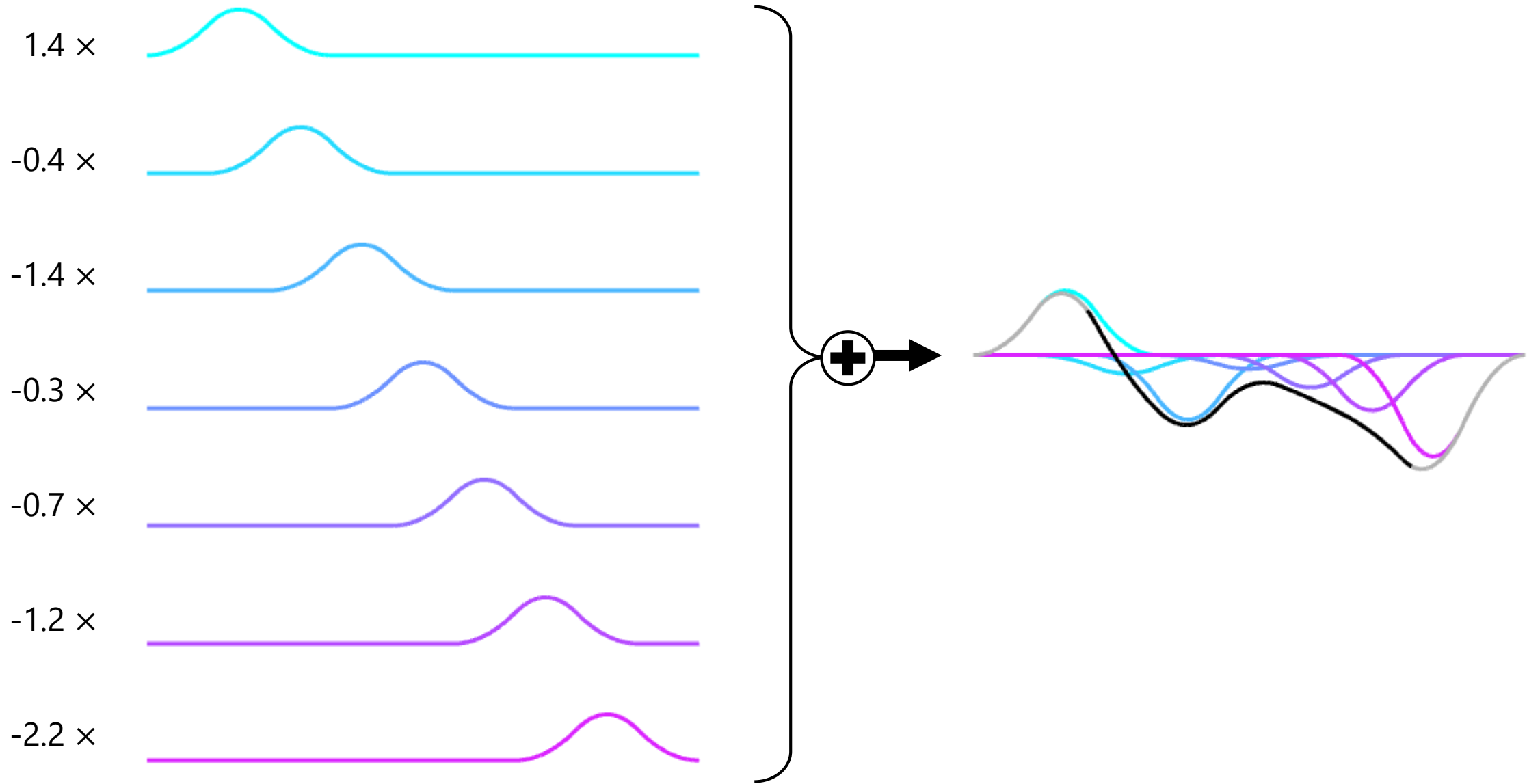
- $B_n(t) = \frac{t}{n} B_{n-1}(t) + \frac{n+1-t}{n} B_{n-1}(t-1)$

- Properties:

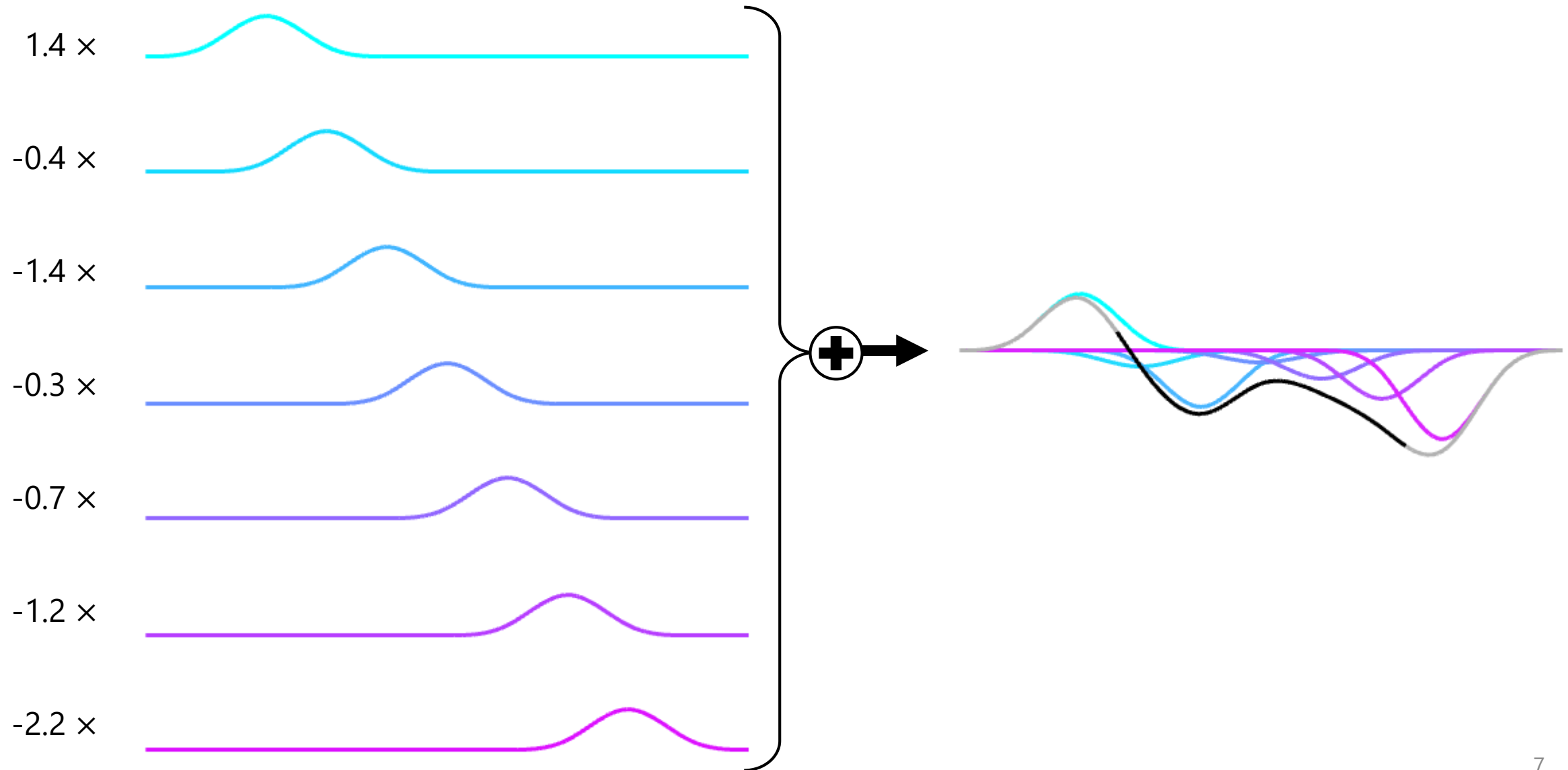
- n-th order piecewise polynomial
  - Zero outside  $[0, n+1]$  (local support)
  - $C^{n-1}$  continuous



# Using quadratic basis $\Rightarrow$ quadratic B-spline



# Using cubic basis $\rightarrow$ cubic B-spline



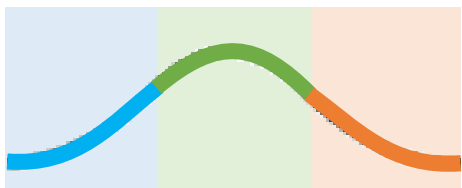
# Important property of basis: partition-of-unity

- X-coord of B-spline:  $x(t) = \sum_i x_i B_n(t - i)$
- Consider moving all control points  $x_i$  by the same amount  $c$  :

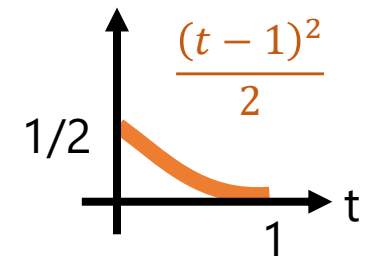
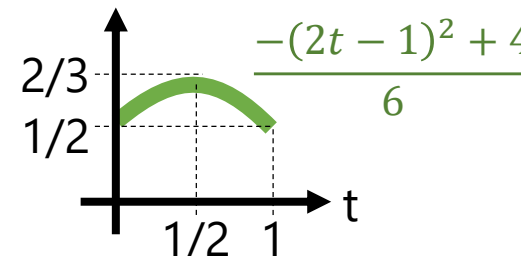
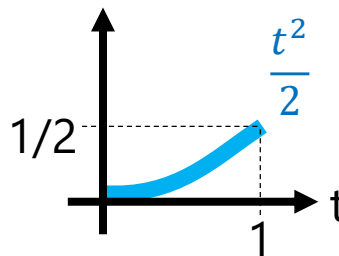
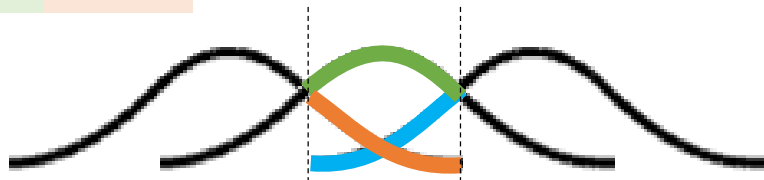
- $x(t) = \sum_i (x_i + c) B_n(t - i)$

$$= \sum_i x_i B_n(t - i) + c \underbrace{\sum_i B_n(t - i)}_1$$

- Partition-of-unity ensures that the entire curve is also moved by  $c$

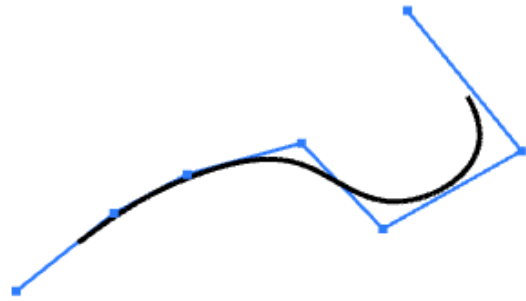


Quadratic basis





# Cubic B-spline vs. Cubic Catmull-Rom spline



Representation

Piecewise cubic

Defined as

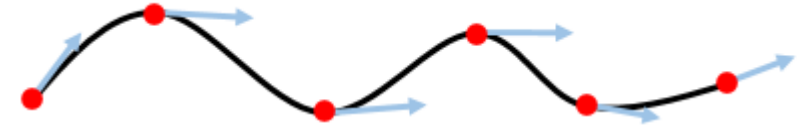
Linear combination of  
cubic bases

Passes through CPs?

No

Continuity across  
intervals

$C^2$ -continuous



Piecewise cubic

Given coordinate value at each knot  $t = t_k$ ,  
compute derivative at each knot

Cubic Hermite interpolation for each interval

Yes

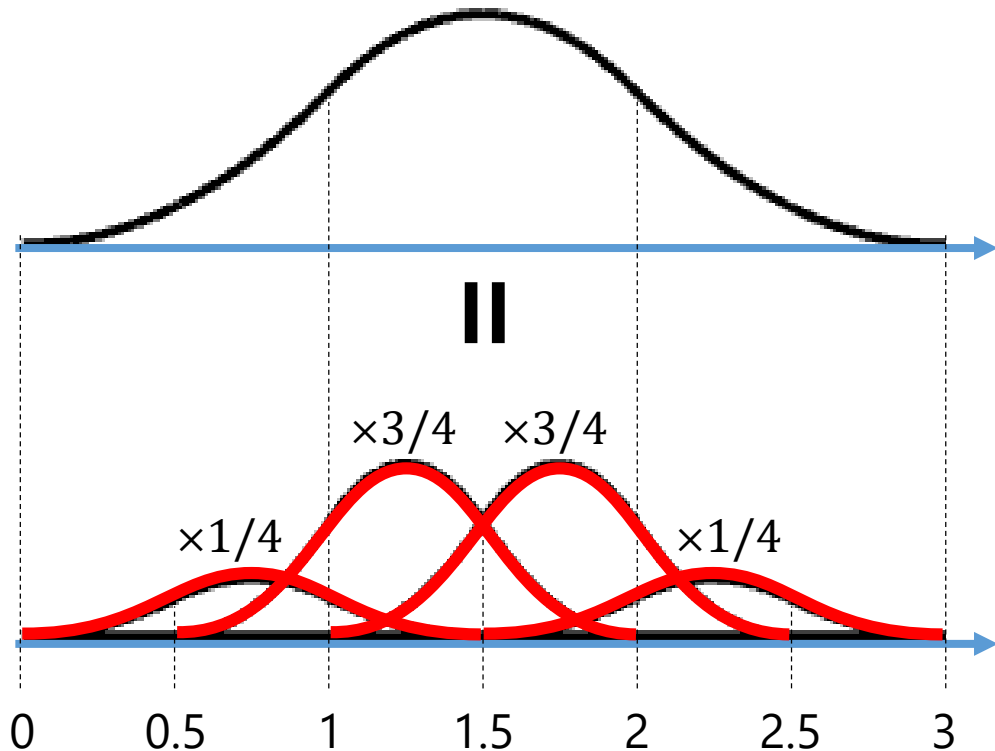
$C^1$ -continuous

# From B-spline to subdivision

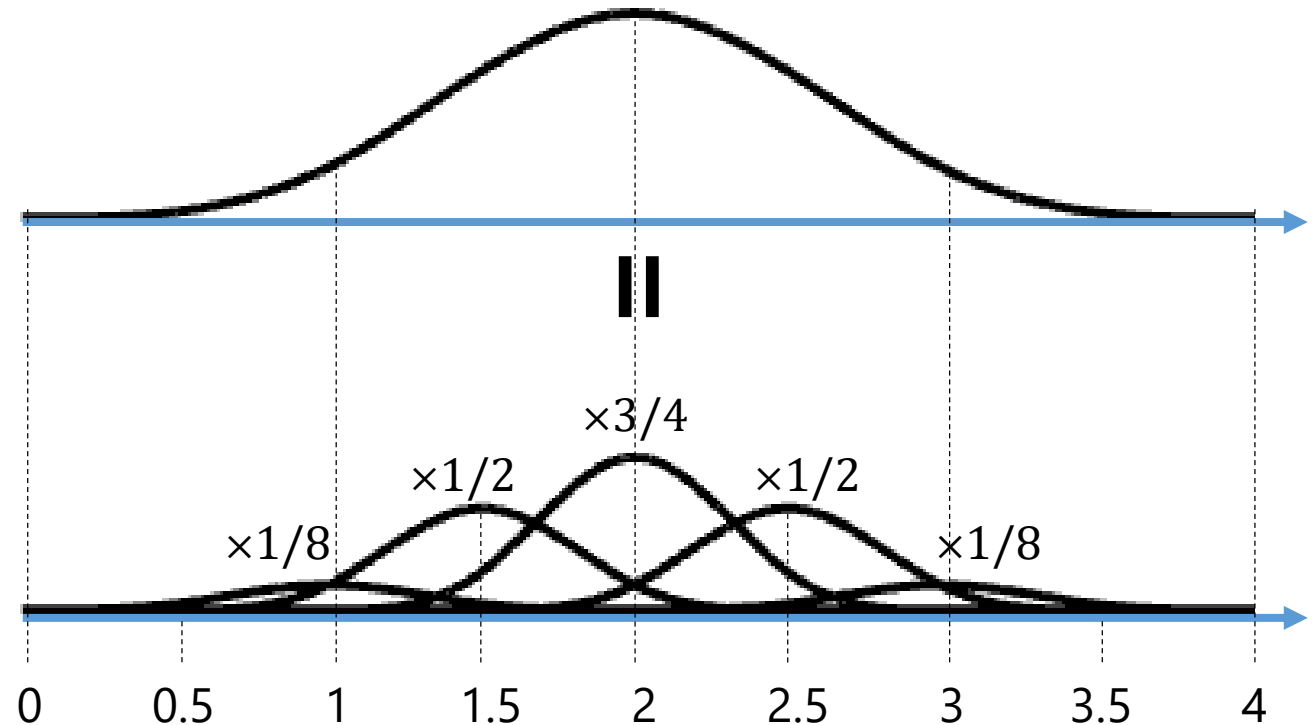
# Another important property of basis function

- Can be decomposed into weighted sum of the same basis functions with halved support

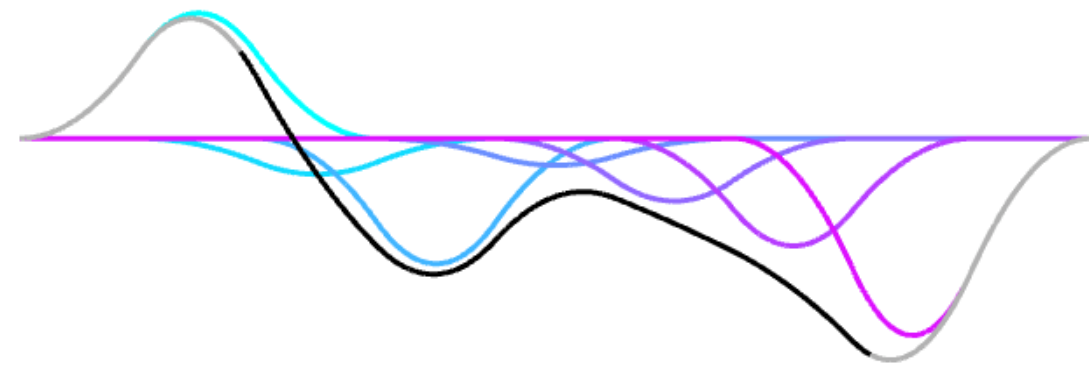
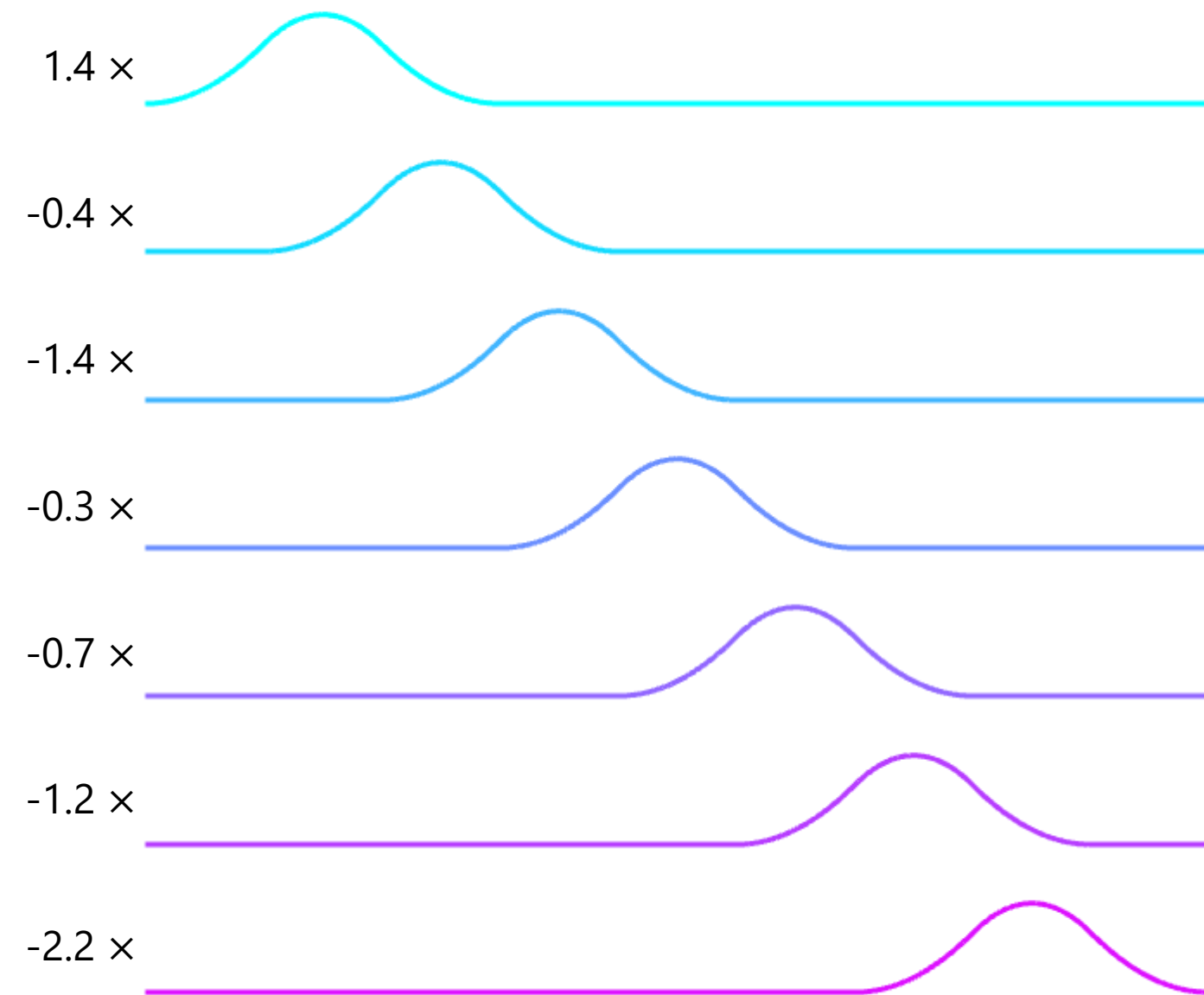
Quadratic



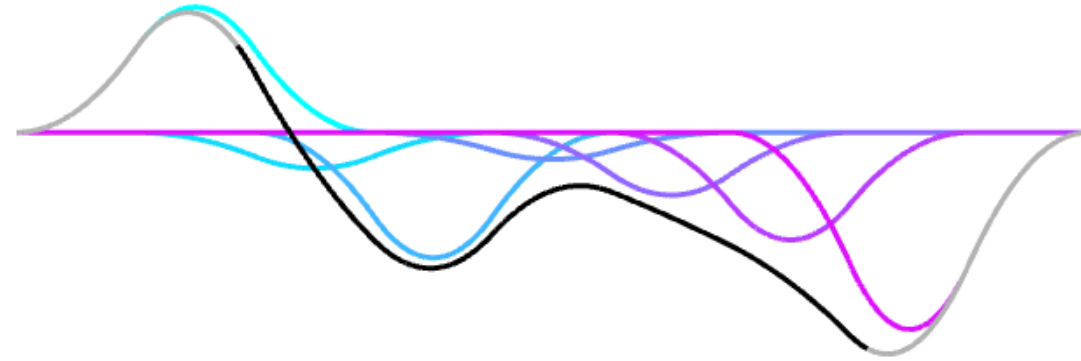
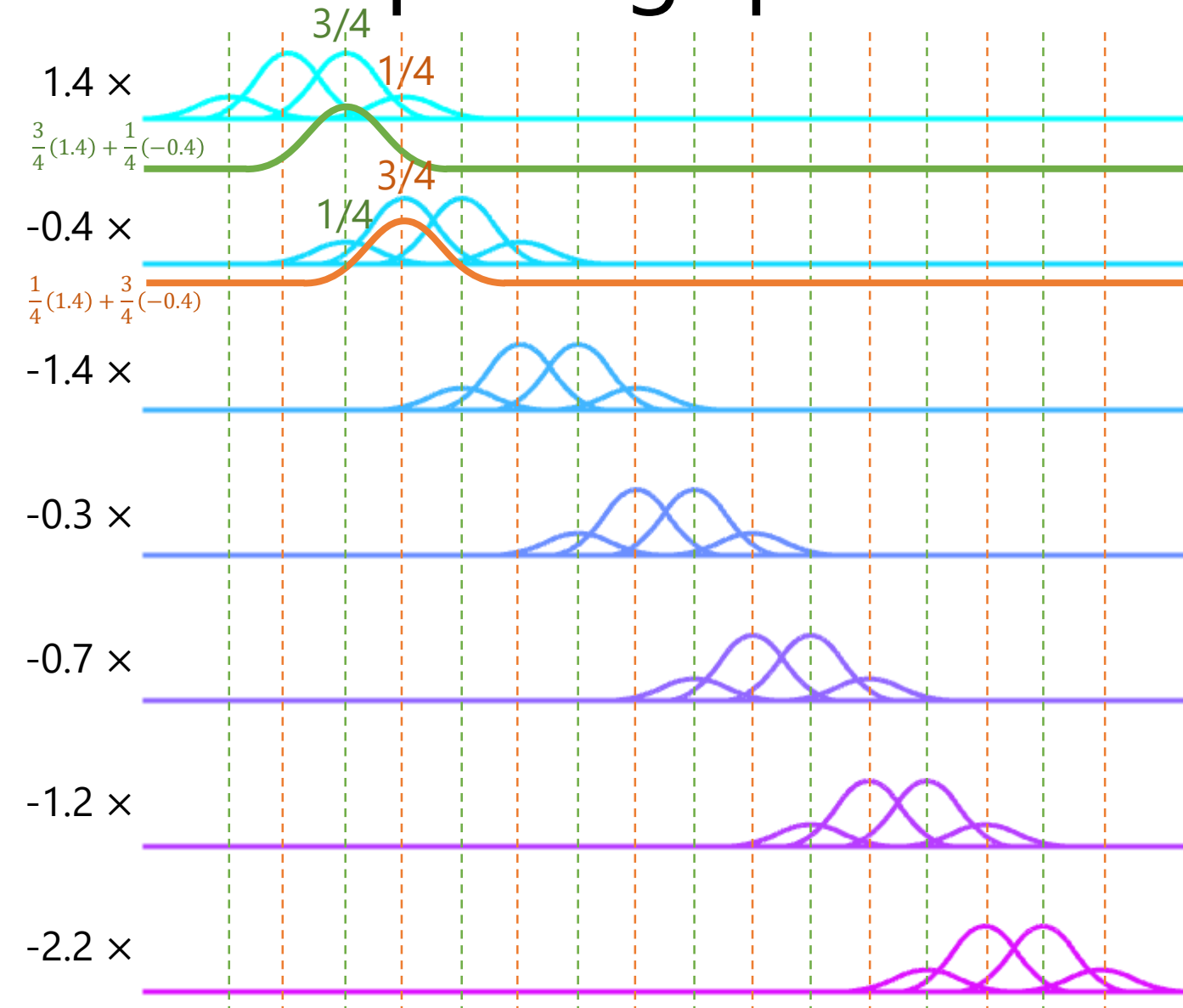
Cubic



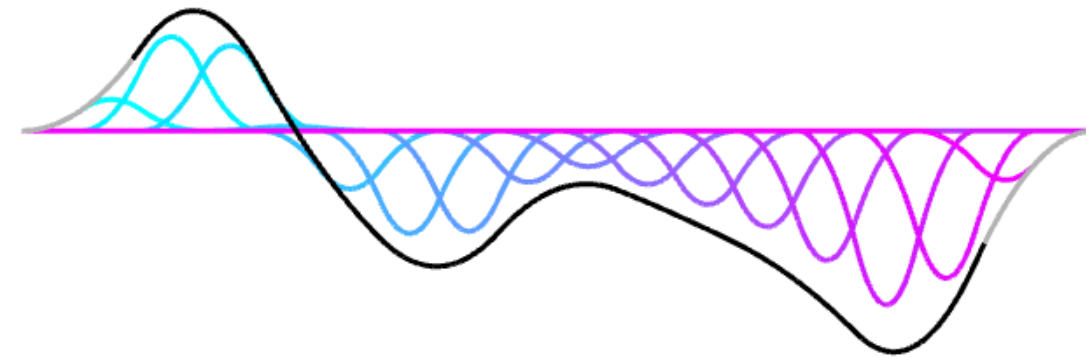
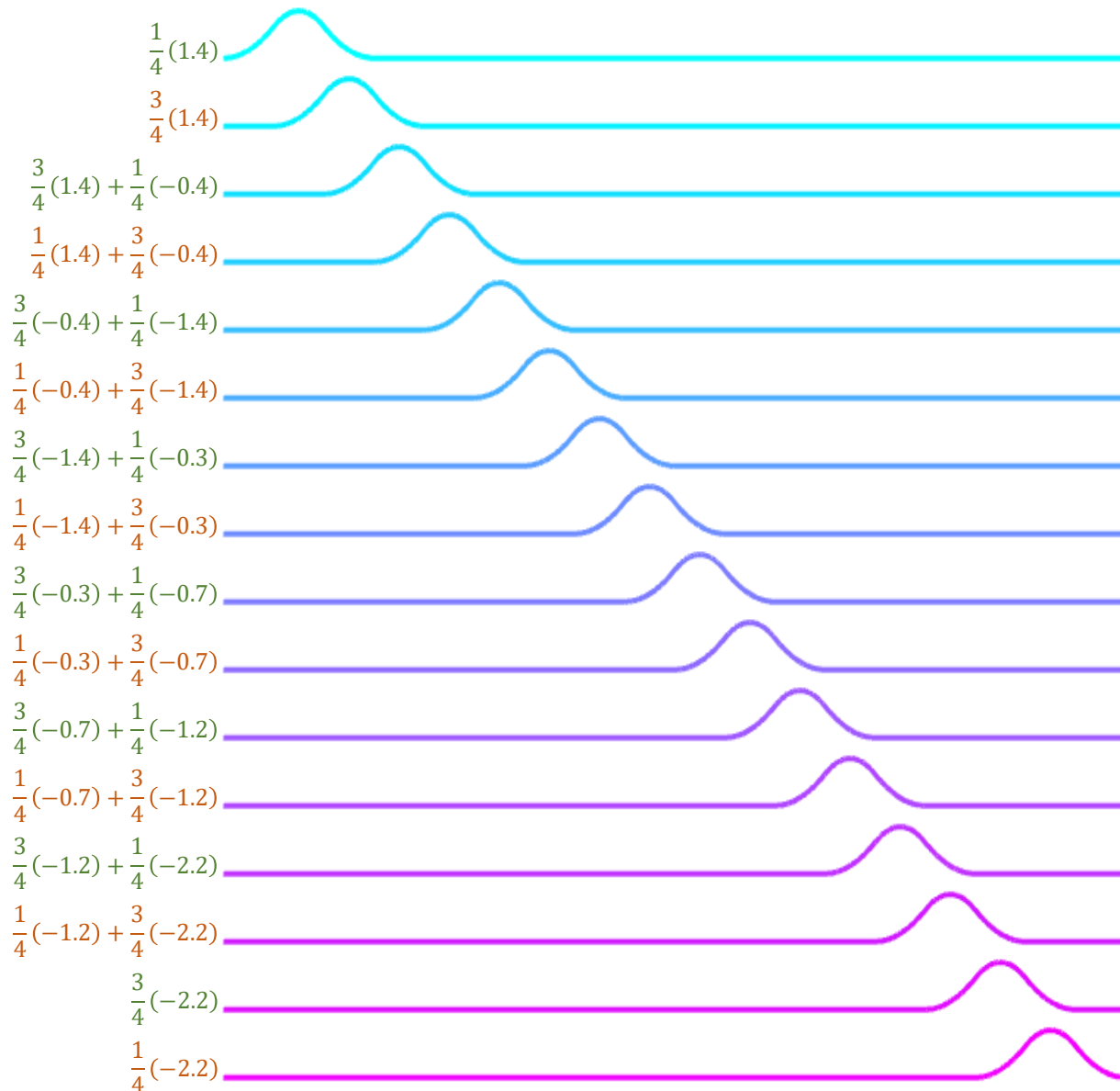
# Decomposing quadratic B-spline



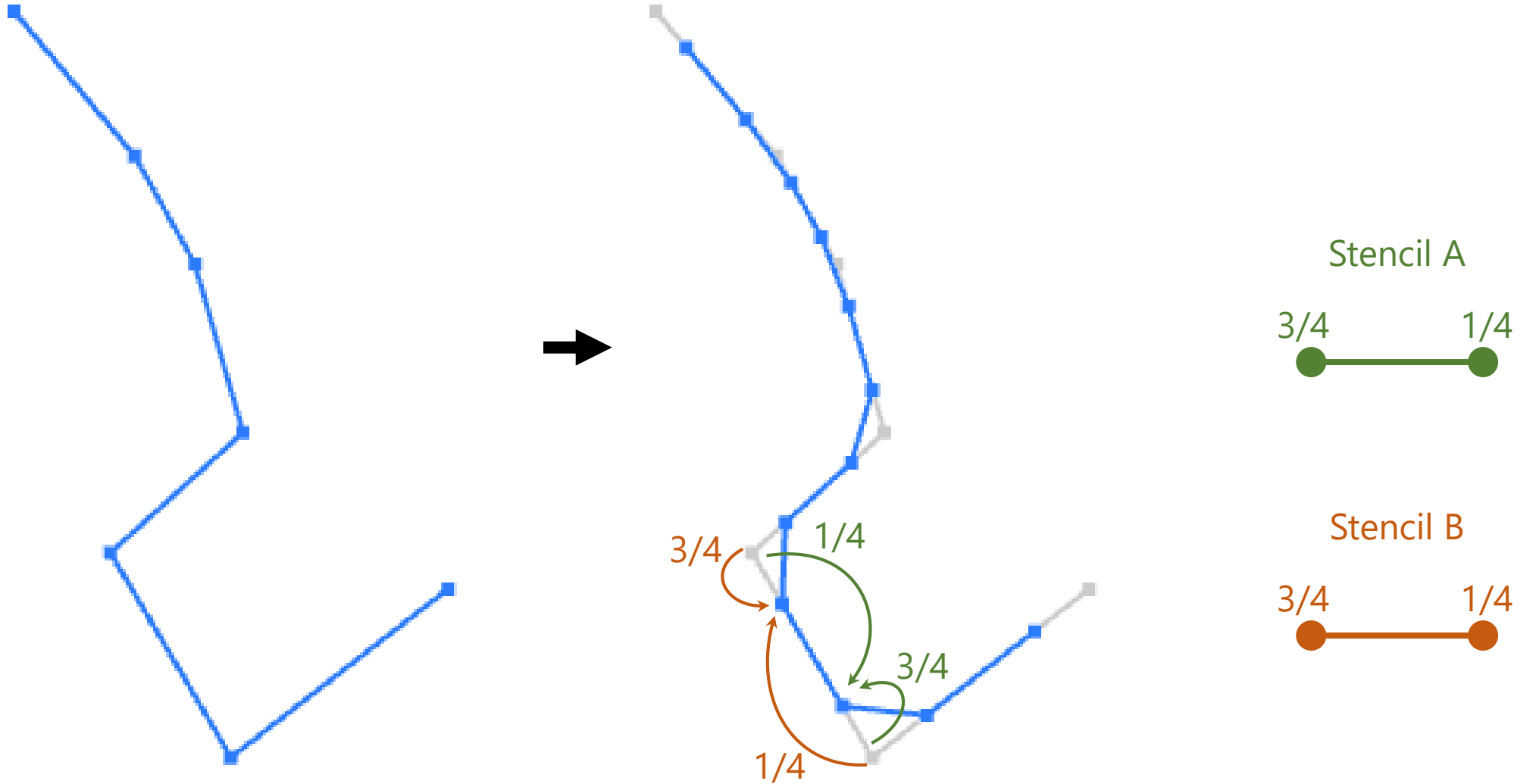
# Decomposing quadratic B-spline



# Decomposing quadratic B-spline

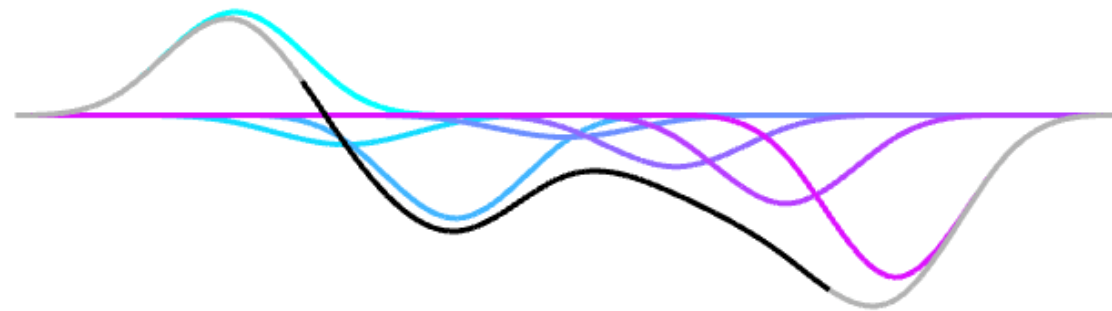
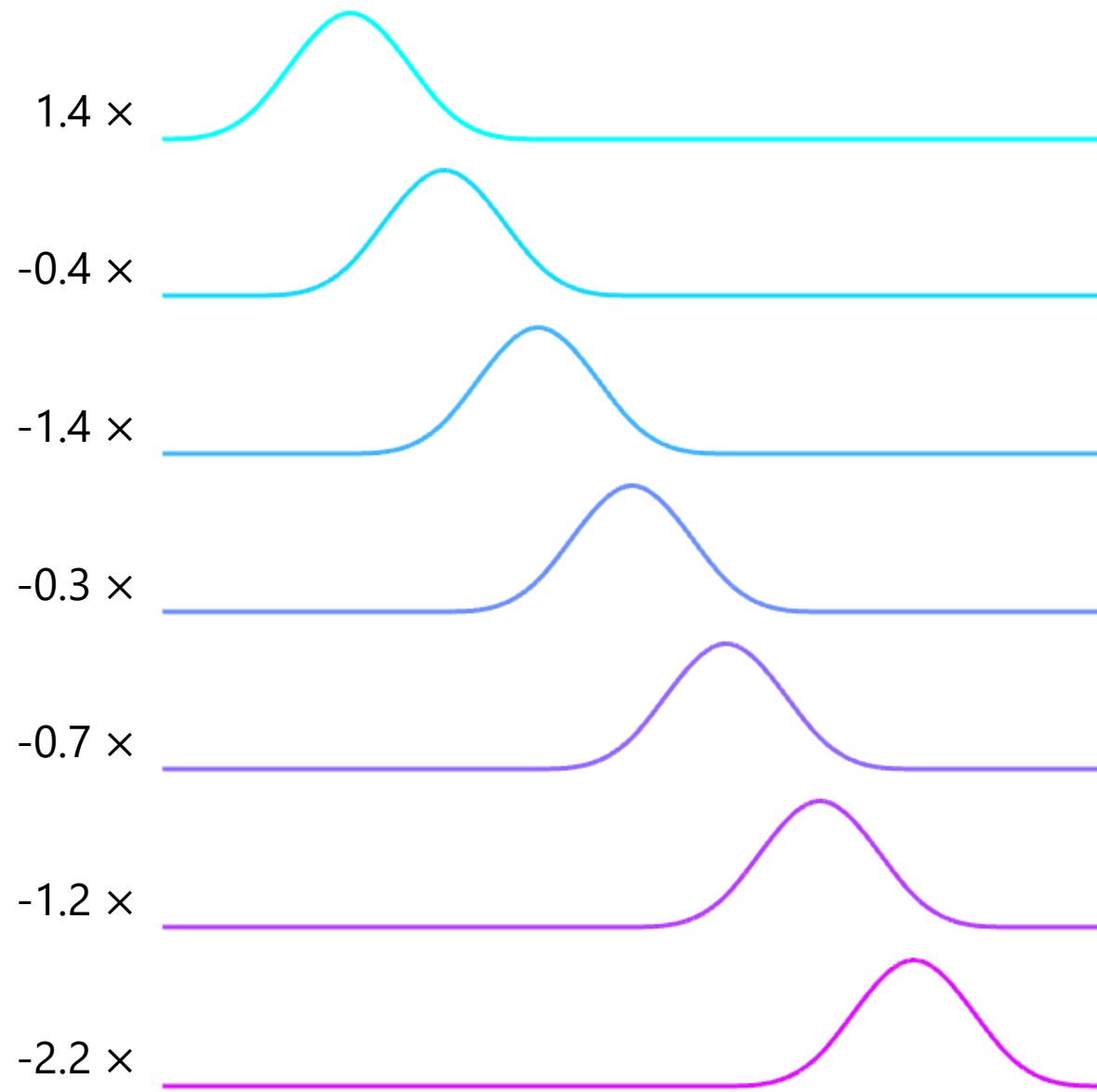


# Generating quadratic curves via subdivision



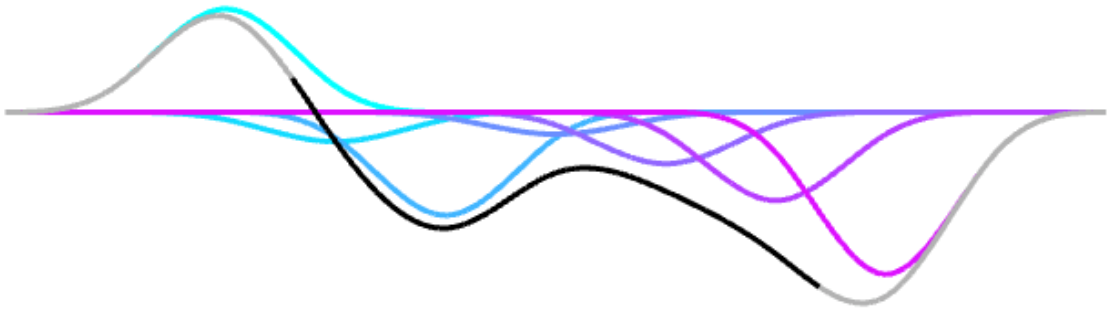
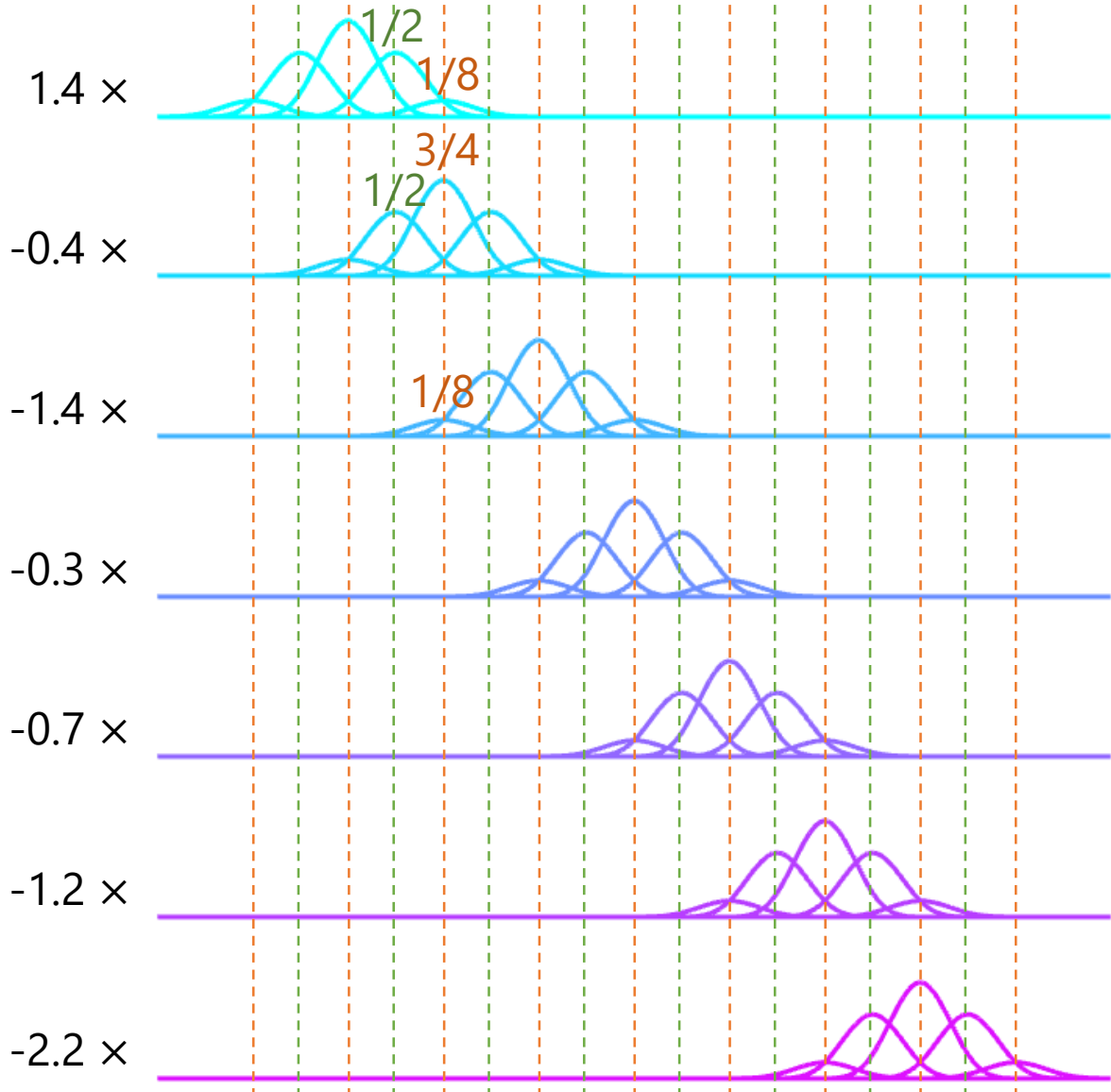
- Split each vertex into 2 new vertices  
(= For each edge, generate 2 new vertices)

# Decomposing cubic B-spline

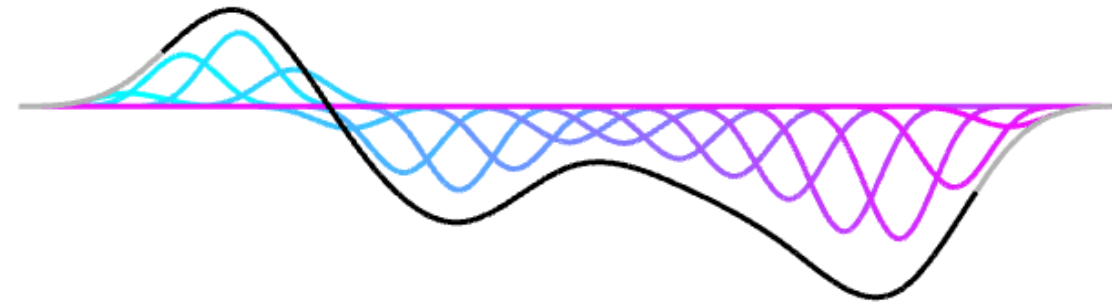
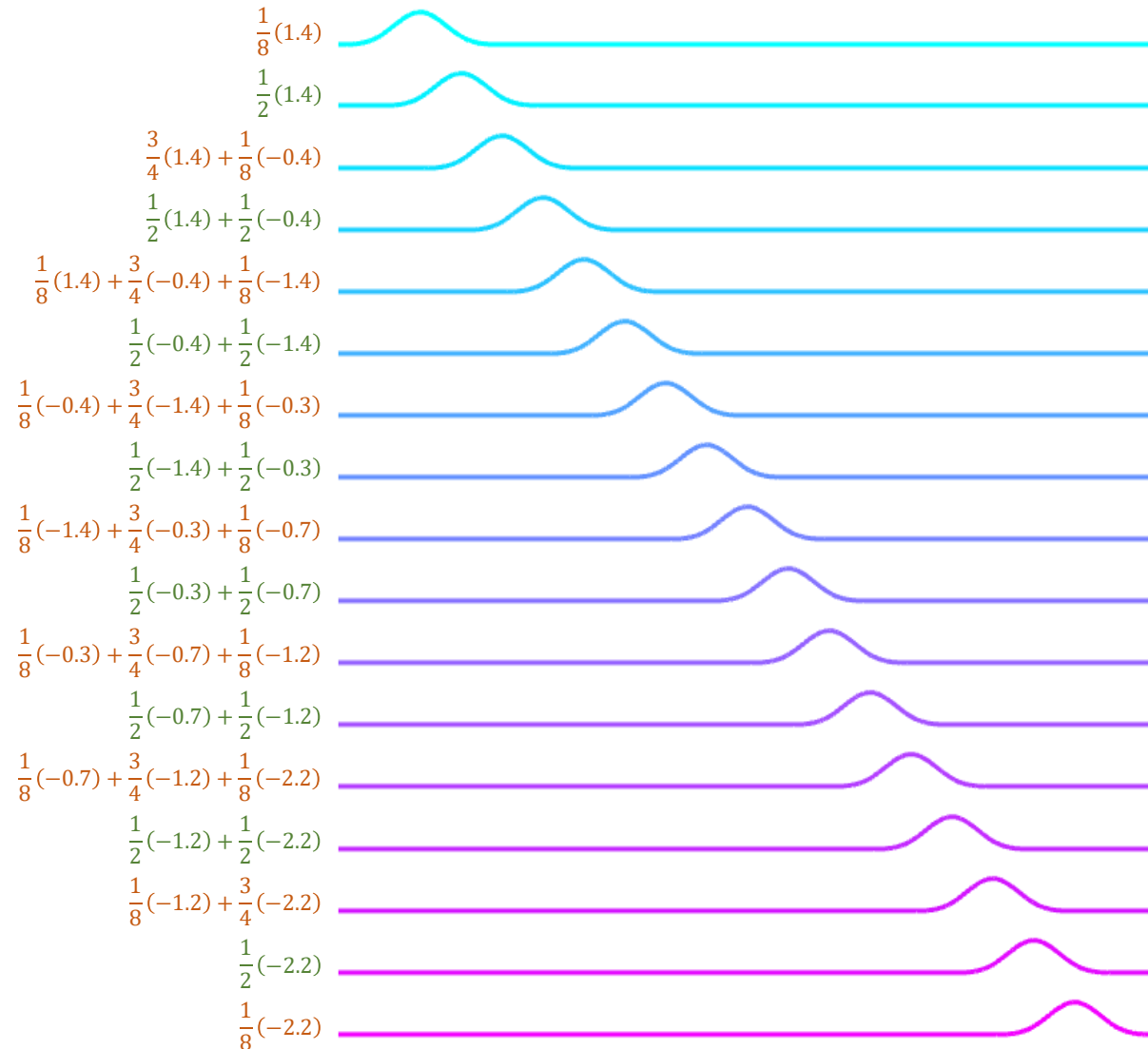




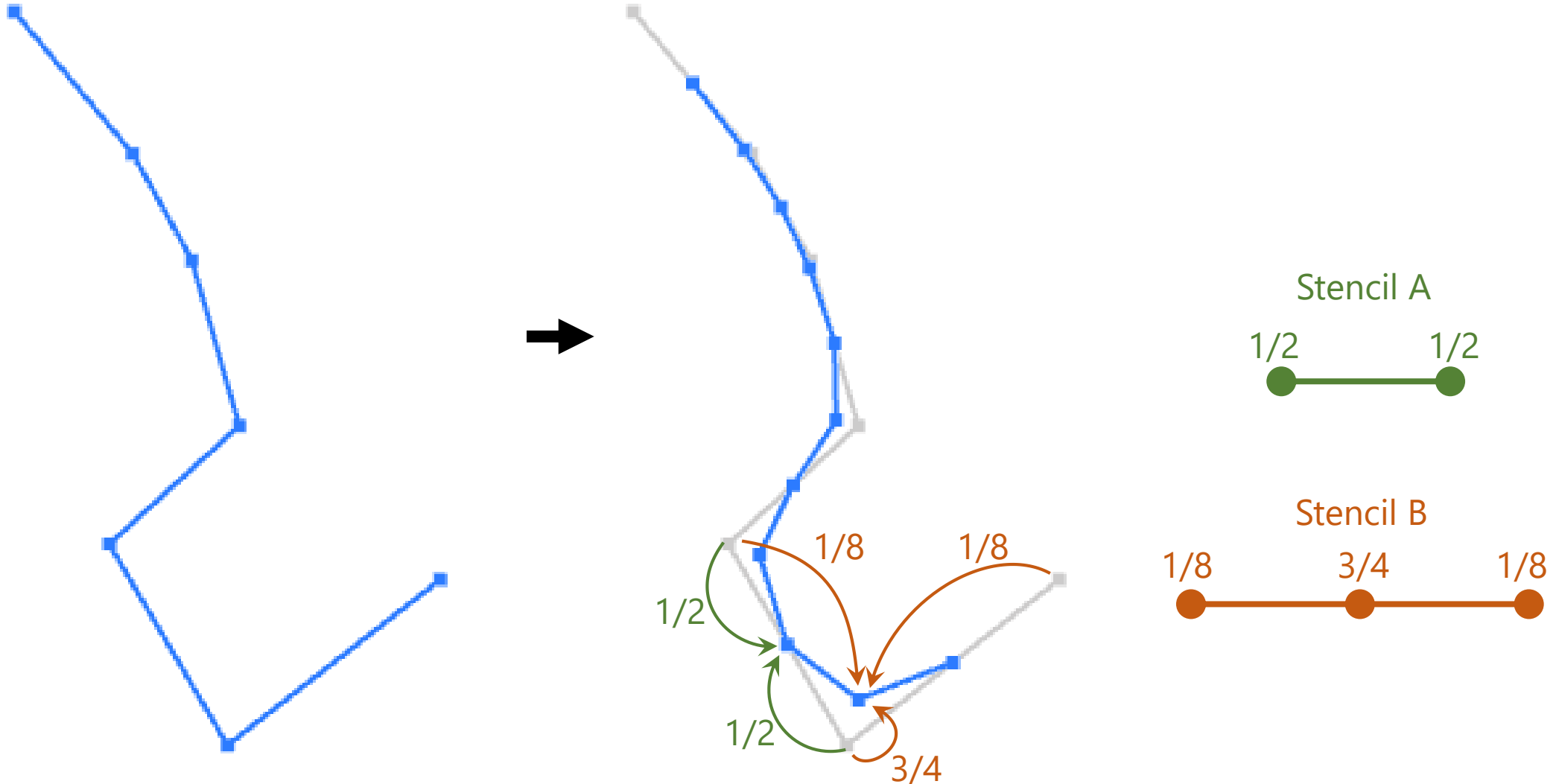
# Decomposing cubic B-spline



# Decomposing cubic B-spline

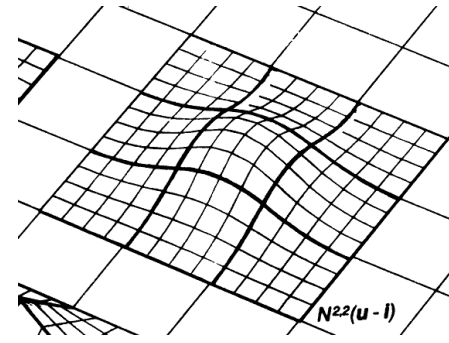


# Generating cubic curves via subdivision

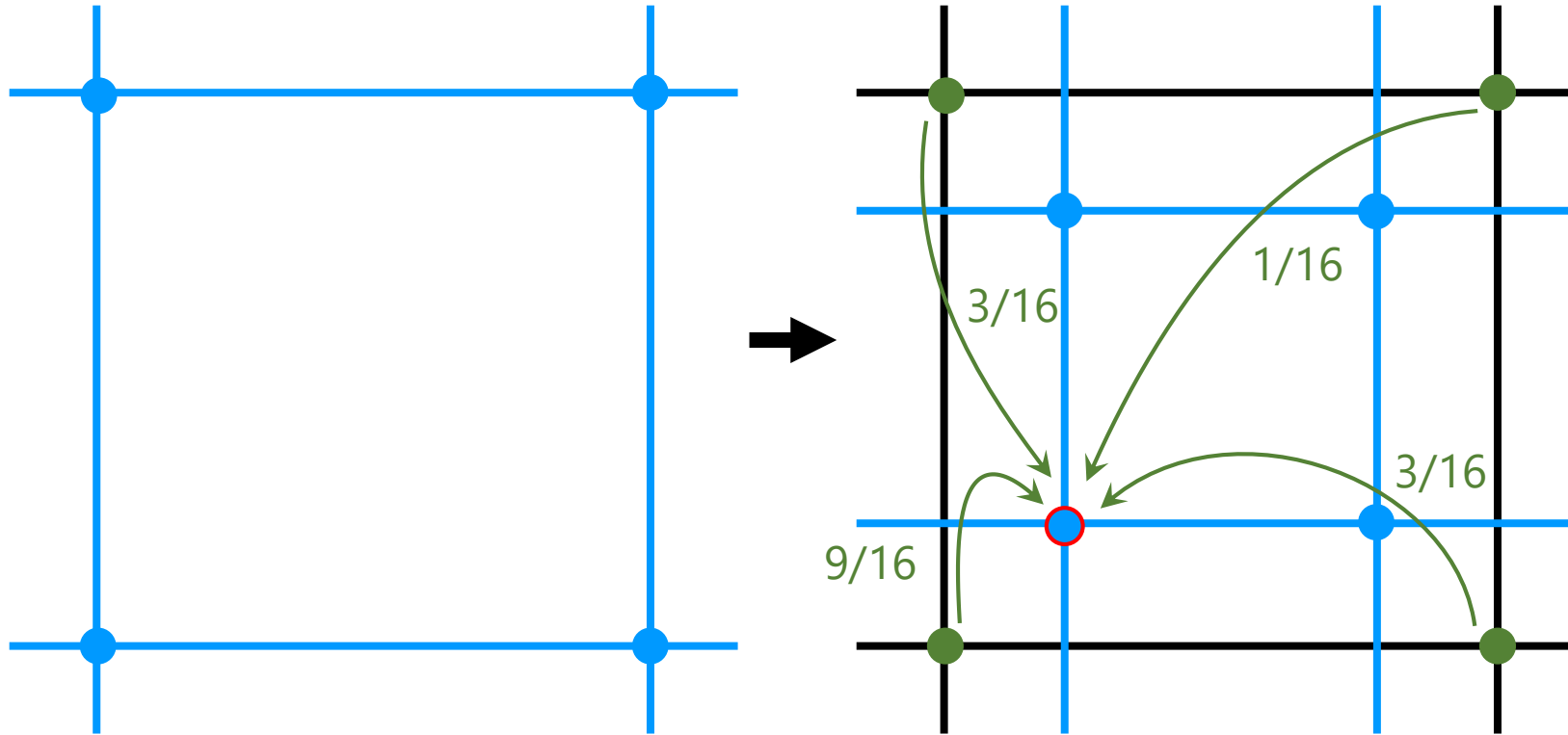


- For each edge, generate a new vertex at its midpoint
- Move each vertex to weighted average of its neighbors

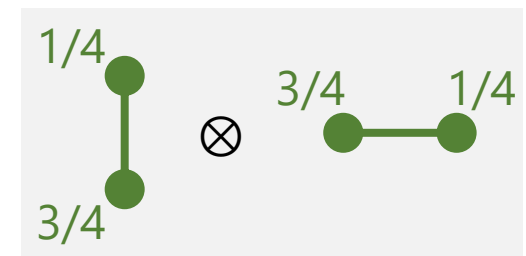
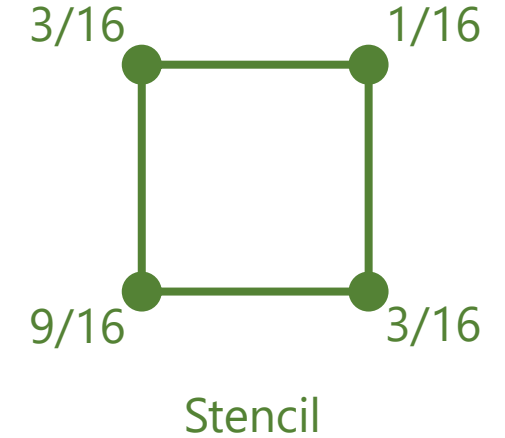
# Generating quadratic surfaces via subdivision



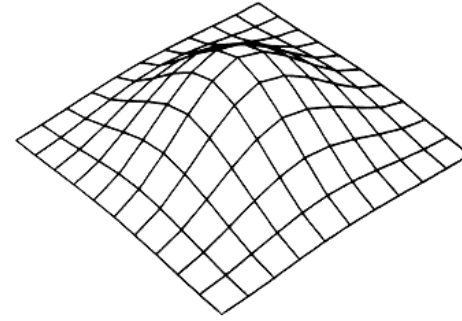
Bi-quadratic basis:  
 $B_{2,2}(s, t) = B_2(s) B_2(t)$



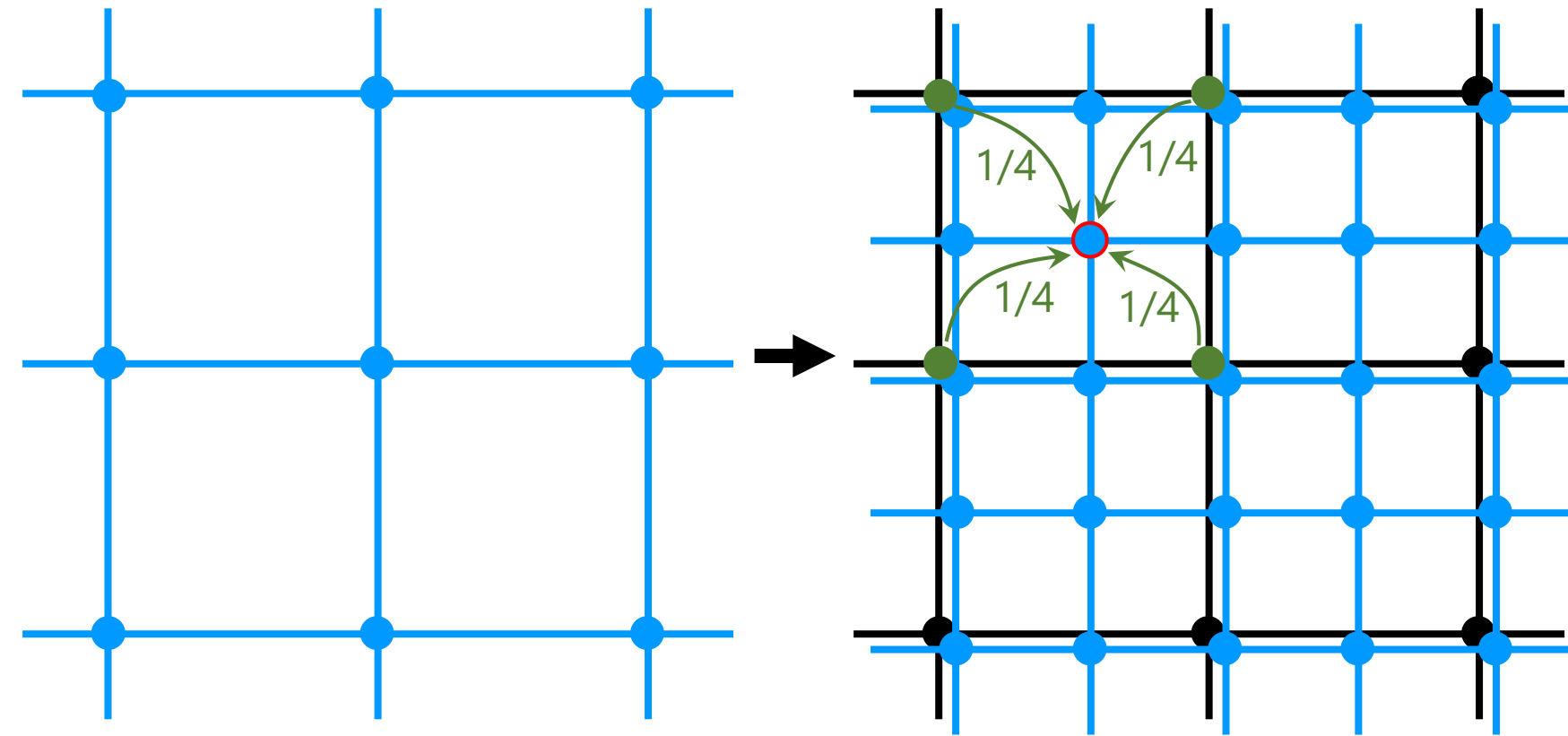
- Split each vertex into 4 new vertices  
 (= For each face, generate 4 new vertices)



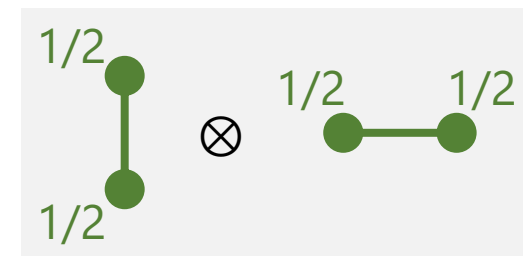
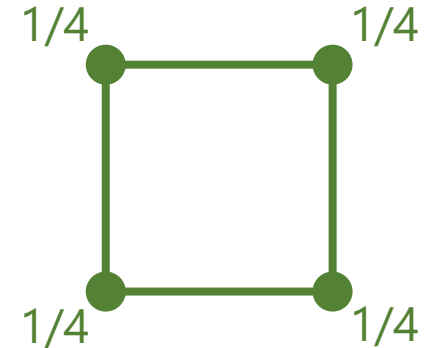
# Generating cubic surfaces via subdivision



Bi-cubic basis:  
 $B_{3,3}(s, t) = B_3(s) B_3(t)$

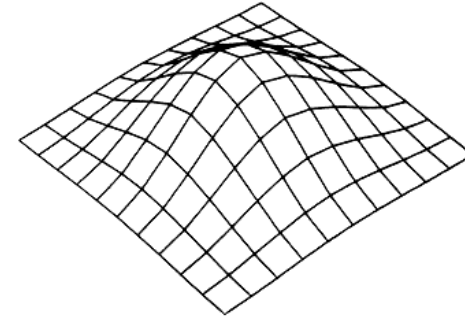


"face point" stencil

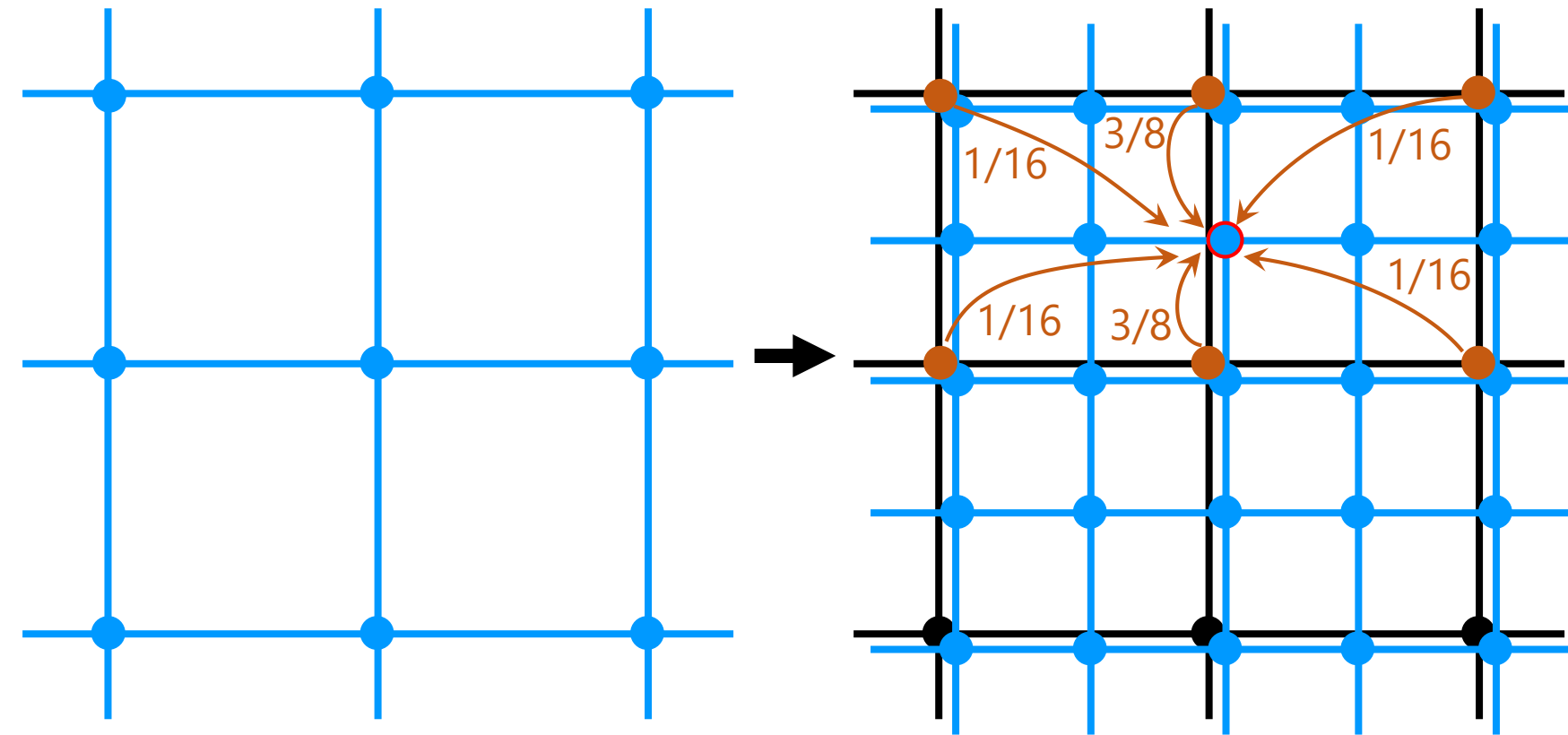


- For each face, generate a new vertex at its barycenter

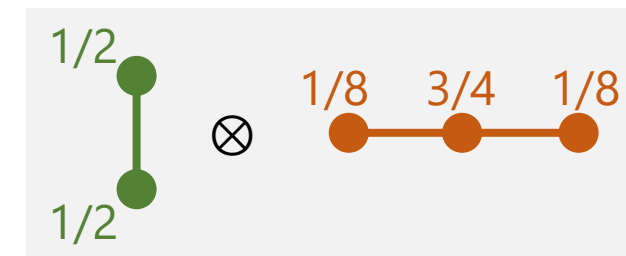
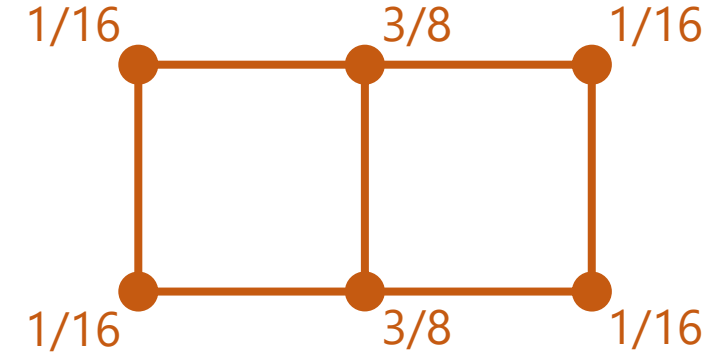
# Generating cubic surfaces via subdivision



Bi-cubic basis:  
 $B_{3,3}(s, t) = B_3(s) B_3(t)$

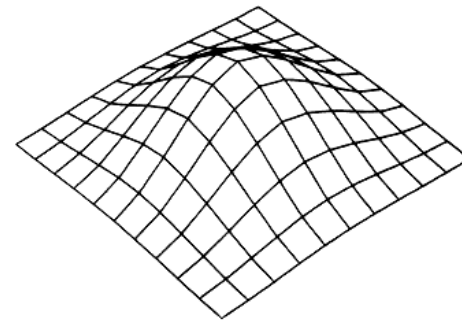


"edge point" stencil



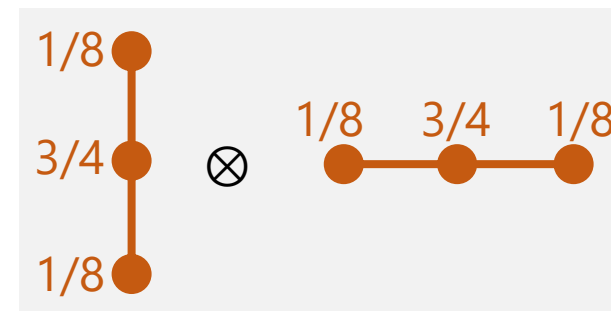
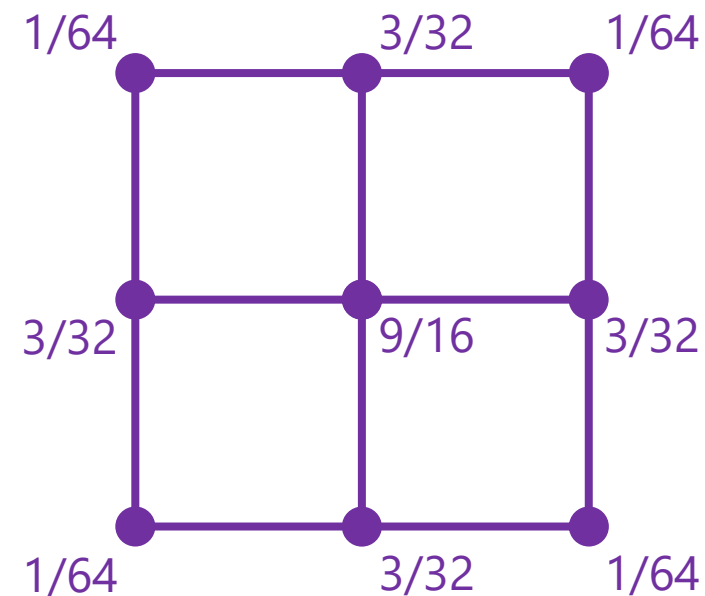
- For each edge, generate a new vertex at weighted average of its neighbors

# Generating cubic surfaces via subdivision



Bi-cubic basis:  
 $B_{3,3}(s, t) = B_3(s) B_3(t)$

"vertex point" stencil



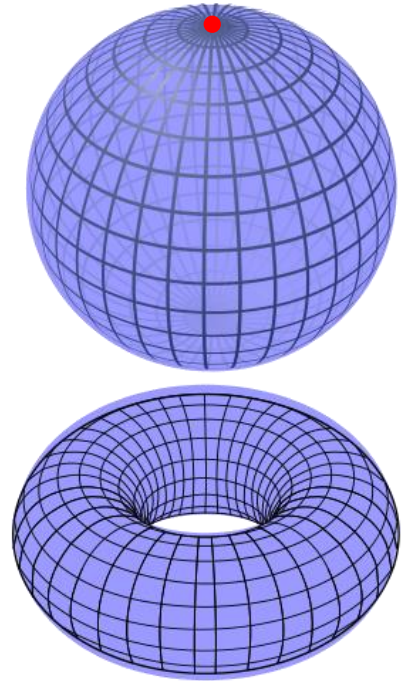
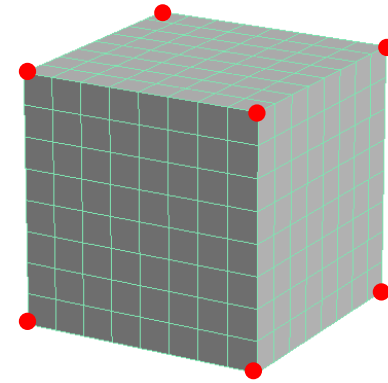
- Move each vertex to weighted average of its neighbors

# Generalizing subdivision scheme

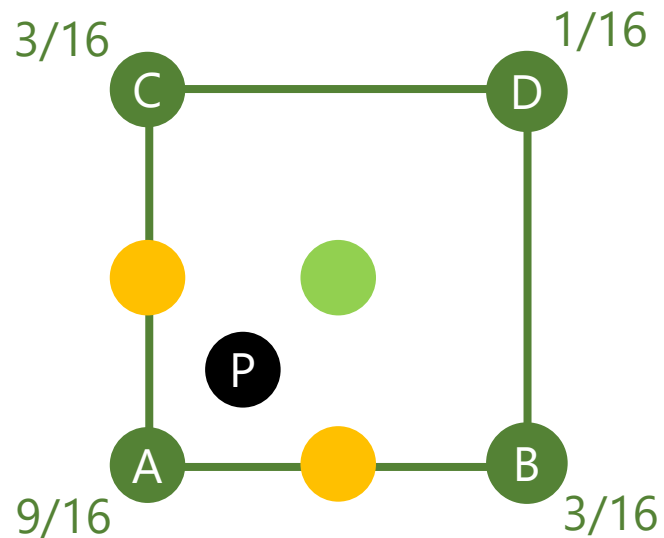


# Fundamental limitation of B-splines

- Need “clean” quadrilateral decomposition of the region
  - “Clean” vertex: # of neighboring faces (**valence**) is 4
    - If valence is not 4 → **singularity**
  - Theoretically impossible to obtain in general
    - Special case: donut (torus)
- What you can do:  
Create B-spline patches around singularities,  
try to maintain continuity across patches by manual work  
→ Very hard, tedious ☹
- Benefit of generalized subdivision scheme: can handle singularities 😊
  - Generalize B-spline stencils through geometric interpretations



# Generalizing quadratic stencil (Doo-Sabin)



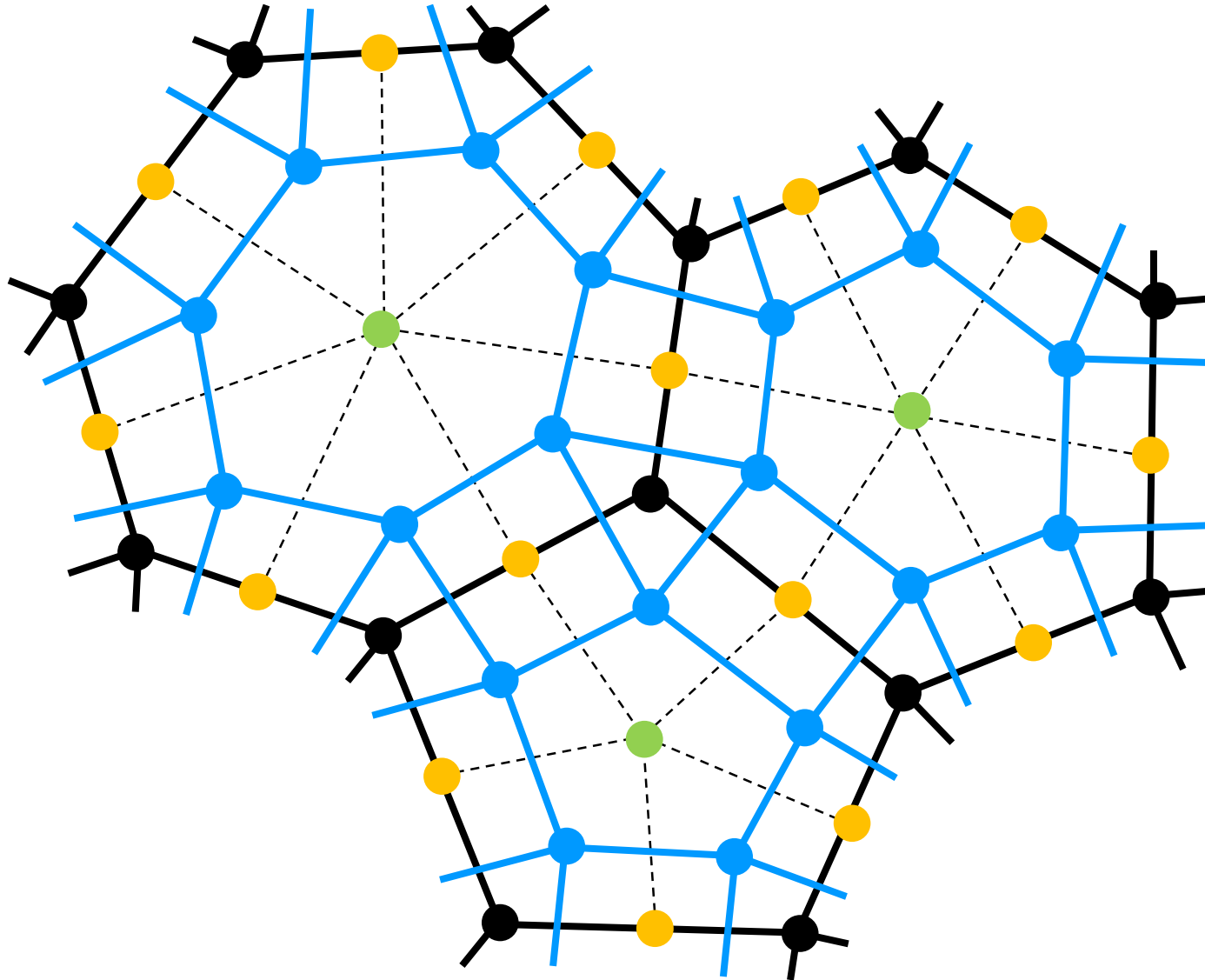
$$P = \frac{1}{16} (9 A + 3 B + 3 C + D)$$
$$= \frac{\text{Barycenter}}{4} + \frac{\text{Midpoint}}{4} + \frac{\text{Midpoint}}{4} + A$$

The diagram shows the derivation of the Doo-Sabin stencil formula. The first line is the formula for the new vertex P. The second line shows the formula broken down into its components: the barycenter (A+B+C+D)/4, the midpoint of edge AB (A+B)/2, the midpoint of edge AC (A+C)/2, and the vertex A. The entire expression is then divided by 4 to match the first line.

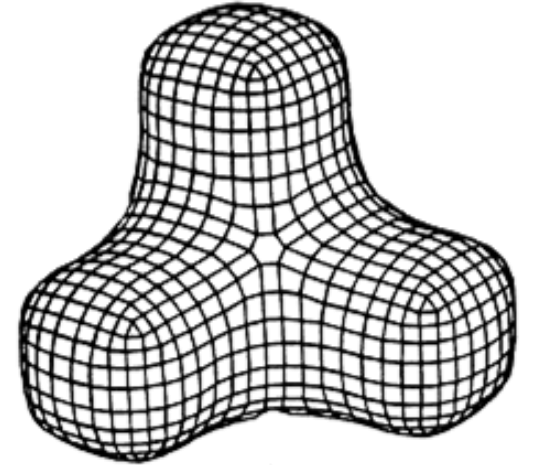
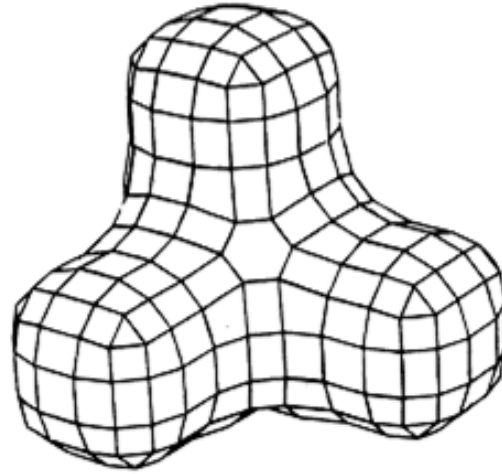
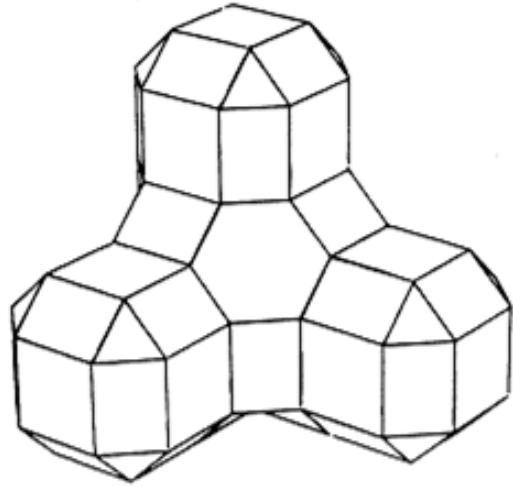
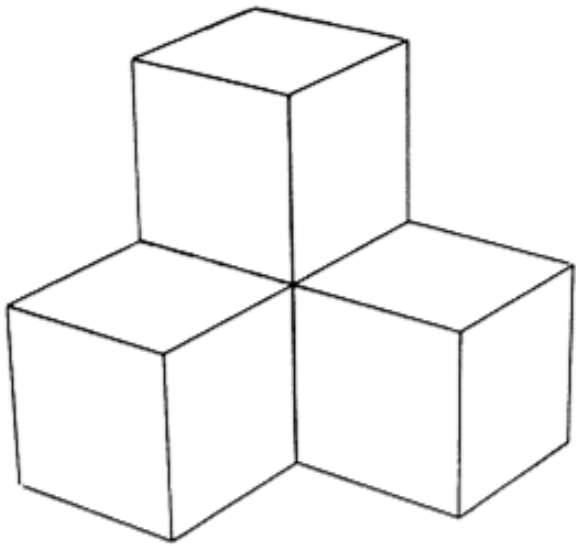
For each polygon's each vertex, generate a new vertex at the average of the polygon's barycenter, its adjacent edges' midpoints, and itself

➔ Applicable to general polygon mesh

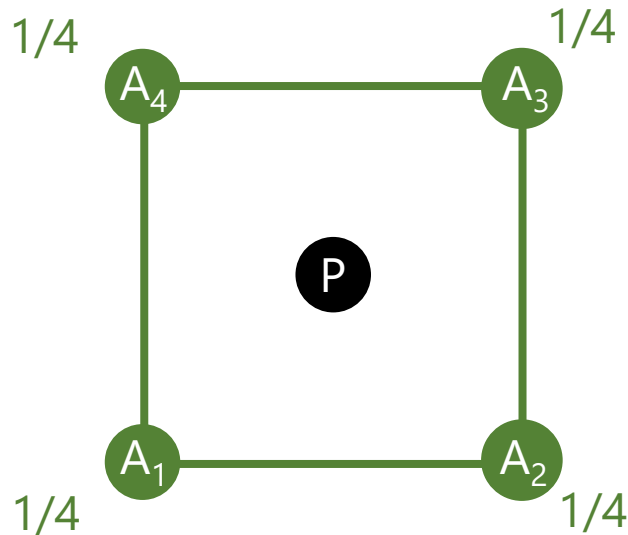
# Examples of Doo-Sabin



# Examples of Doo-Sabin



# Generalizing cubic stencils (Catmull-Clark)

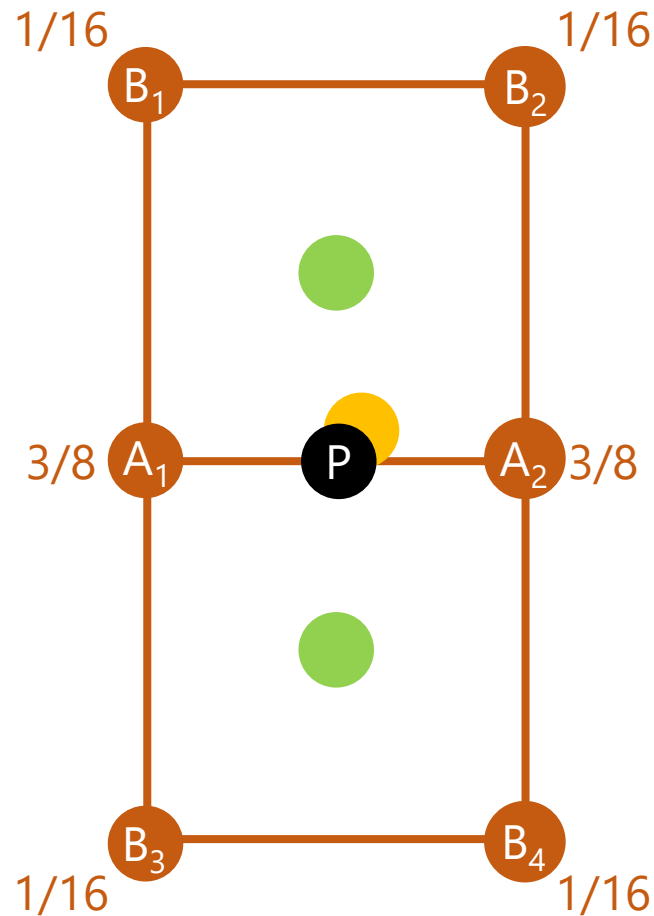


$$P = \frac{A_1 + A_2 + A_3 + A_4}{4}$$

For each polygon, generate a new vertex at its barycenter

➔ Applicable to general polygon mesh

# Generalizing cubic stencils (Catmull-Clark)



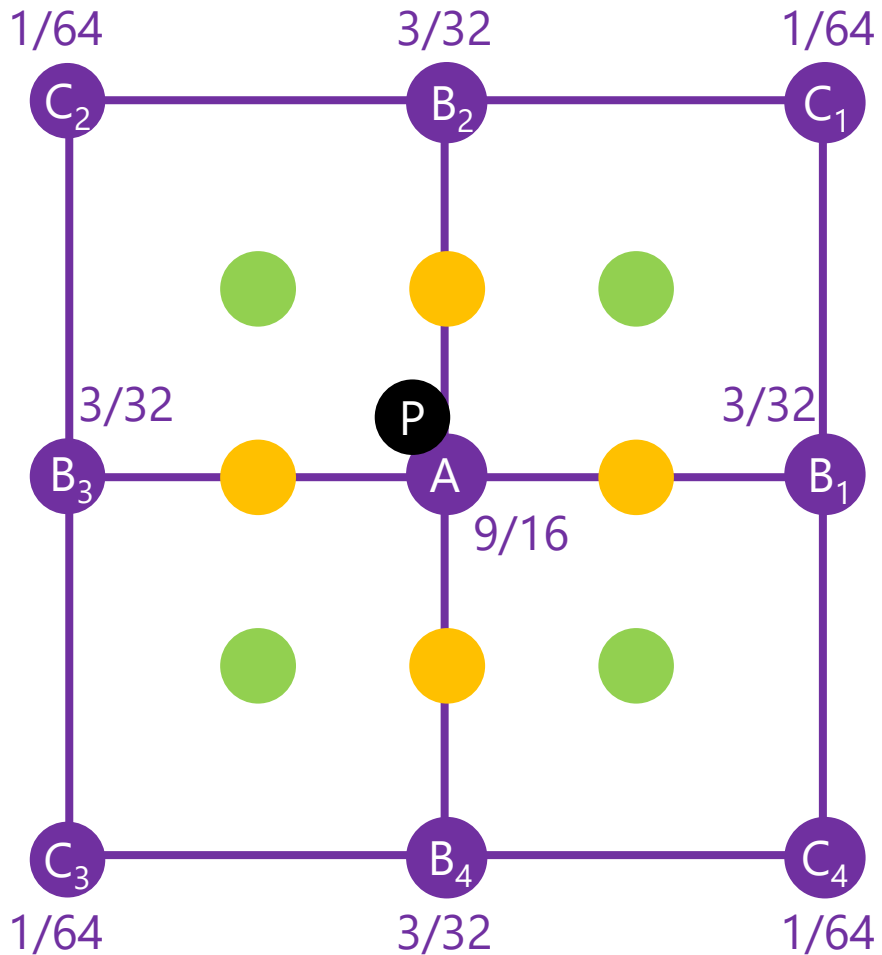
$$P = \frac{3}{8} (A_1 + A_2) + \frac{1}{16} (B_1 + B_2 + B_3 + B_4)$$

$$\begin{aligned} & \frac{\text{Barycenter}}{A_1 + A_2 + B_1 + B_2} + \frac{\text{Barycenter}}{A_1 + A_2 + B_3 + B_4} + \frac{\text{Midpoint}}{A_1 + A_2} \\ &= \frac{\frac{A_1 + A_2 + B_1 + B_2}{4} + \frac{A_1 + A_2 + B_3 + B_4}{4}}{2} + \frac{\frac{A_1 + A_2}{2}}{2} \end{aligned}$$

For each edge, generate a new vertex at the average of the barycenters of its adjacent polygons and its midpoint

→ Applicable to general polygon mesh

# Generalizing cubic stencils (Catmull-Clark)



$$P = \frac{9}{16}A + \frac{3}{32}(B_1 + B_2 + B_3 + B_4) + \frac{1}{64}(C_1 + C_2 + C_3 + C_4)$$

$$= \frac{1}{4} \left\{ \begin{array}{c} \text{Barycenter} \\ \frac{A + B_1 + C_1 + B_2}{4} + \frac{A + B_2 + C_2 + B_3}{4} + \frac{A + B_3 + C_3 + B_4}{4} + \frac{A + B_4 + C_4 + B_1}{4} \end{array} \right\} Q$$

$$+ \frac{2}{4} \left\{ \begin{array}{c} \text{Midpoint} \\ \frac{A + B_1}{2} + \frac{A + B_2}{2} + \frac{A + B_3}{2} + \frac{A + B_4}{2} \end{array} \right\} R$$

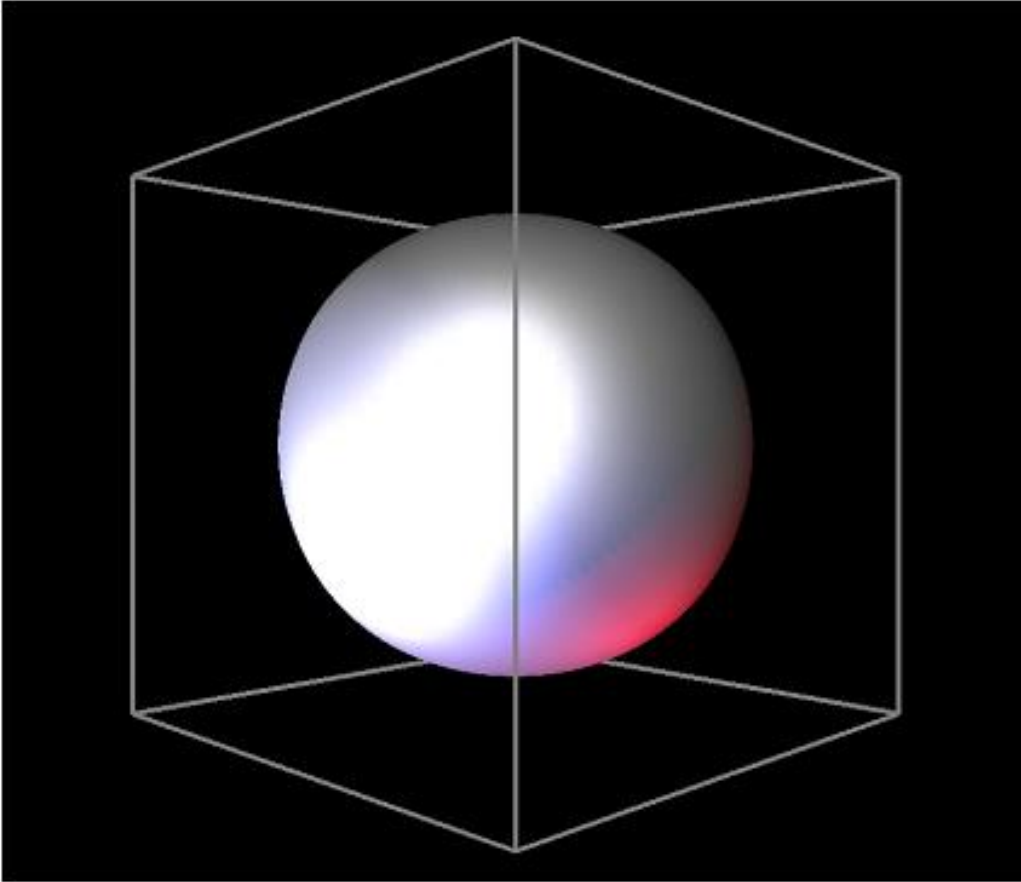
$$+ \frac{1}{4}A$$

When A's valence is n,

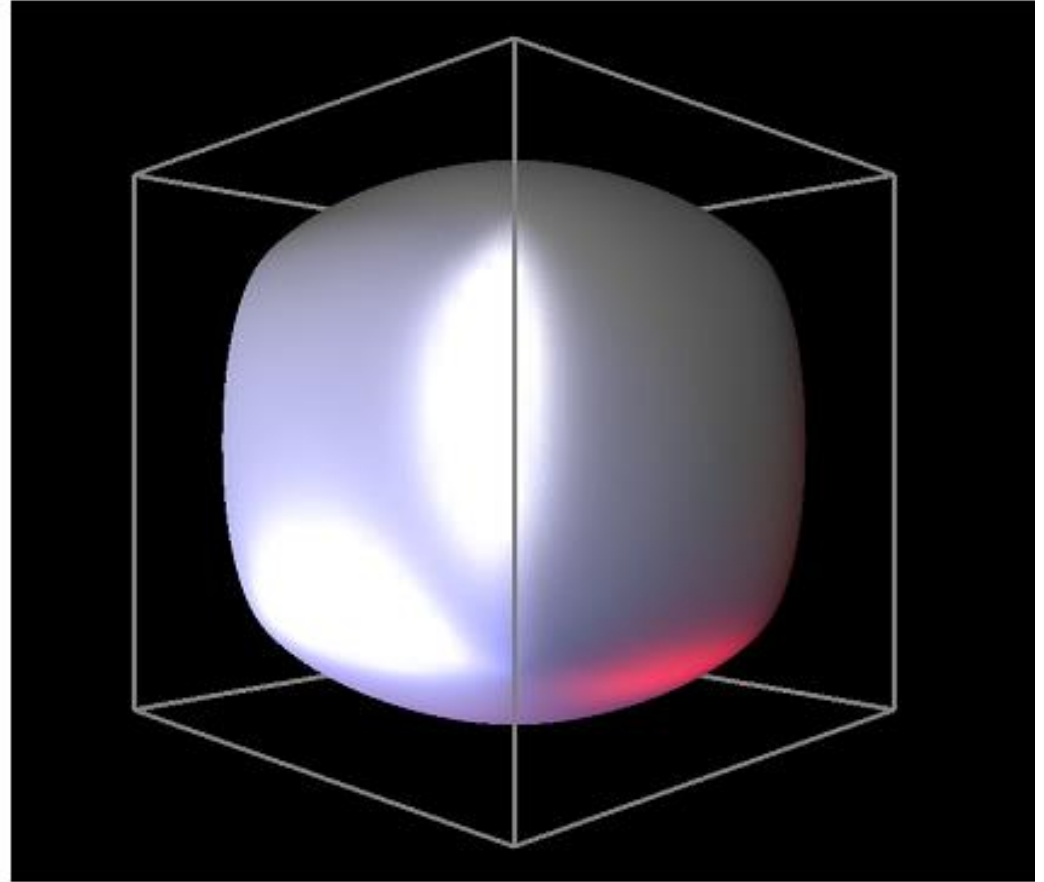
$$P = \frac{1}{n}Q + \frac{2}{n}R + \frac{n-3}{n}A$$

➔ Applicable to general polygon mesh

# Comparison



*Catmull-Clark* = cubic surface

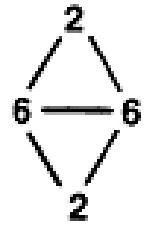
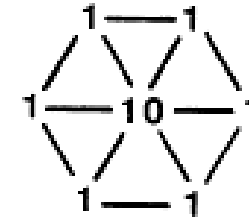
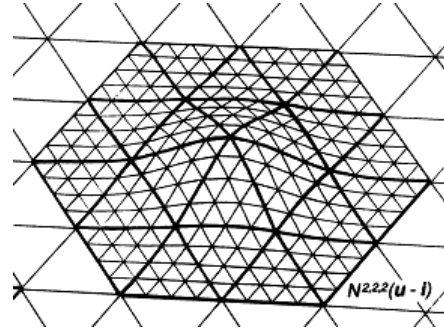


*Doo-Sabin* = quadratic surface

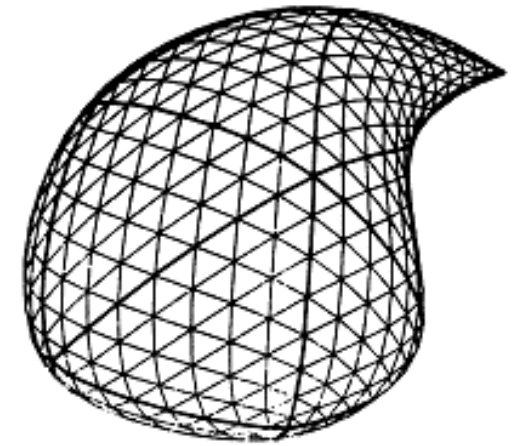
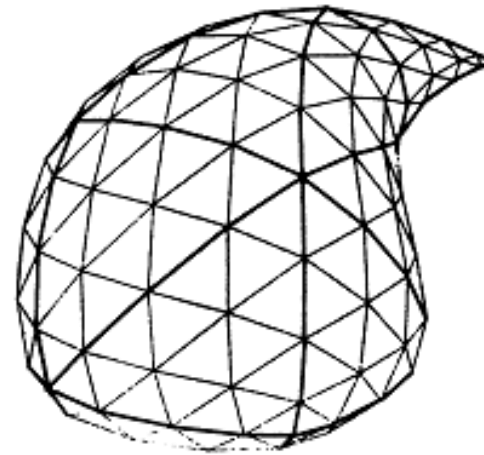
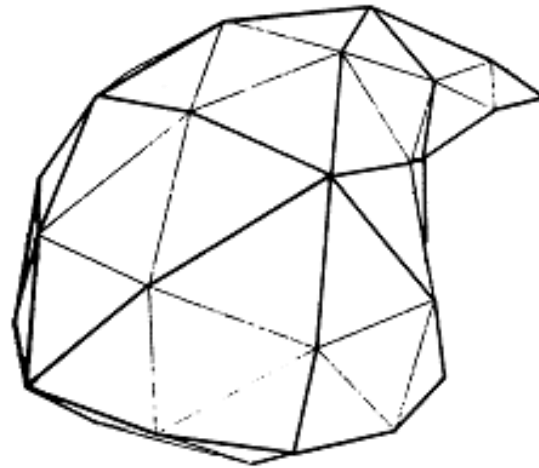
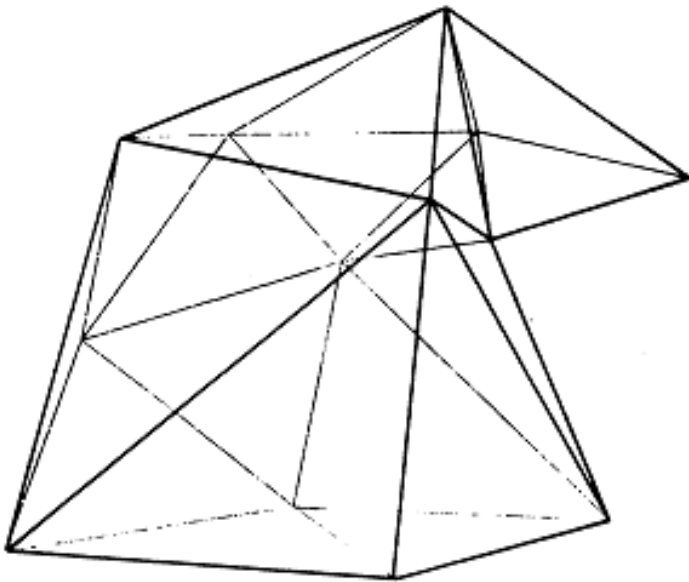


# Subdivision scheme for triangle meshes (Loop)

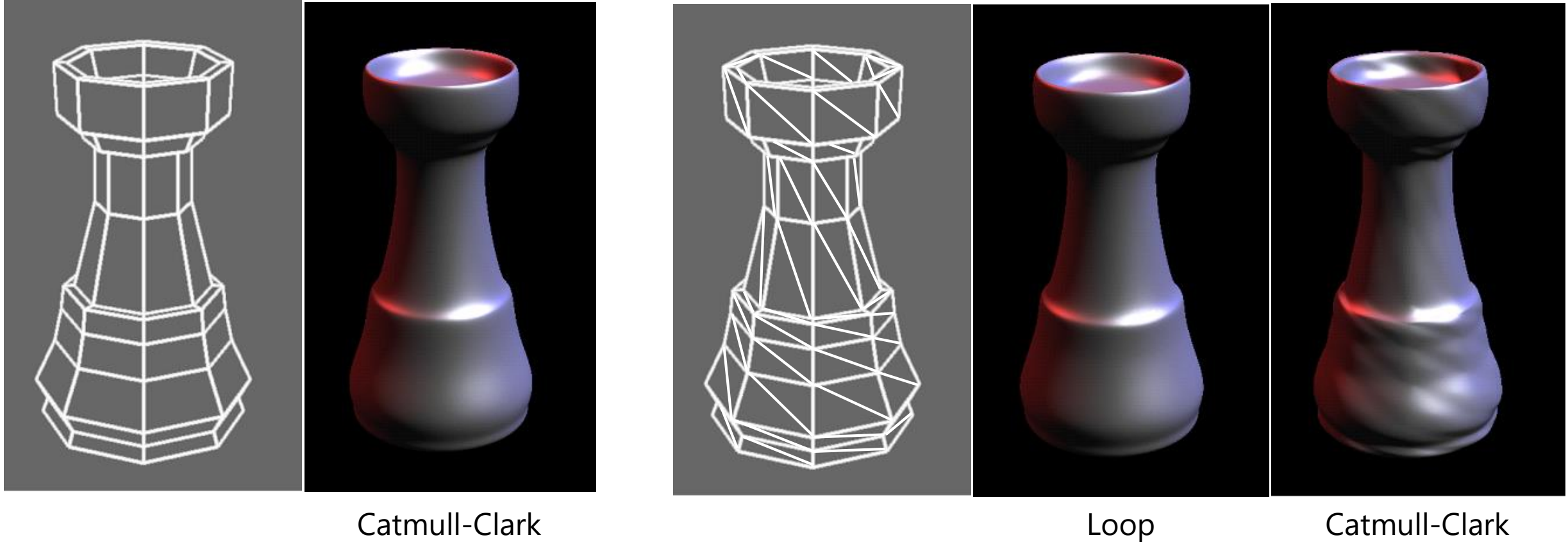
- Based on B-spline basis defined on triangular lattice



- $C^2$ -continuous cubic surface except at singularities



# Comparing Catmull-Clark & Loop



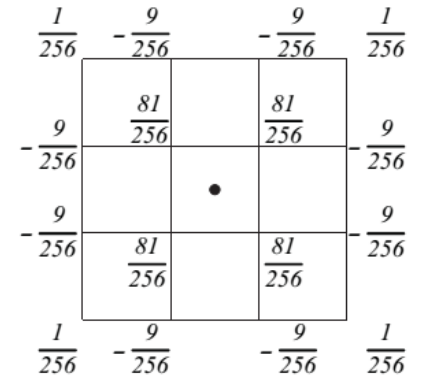
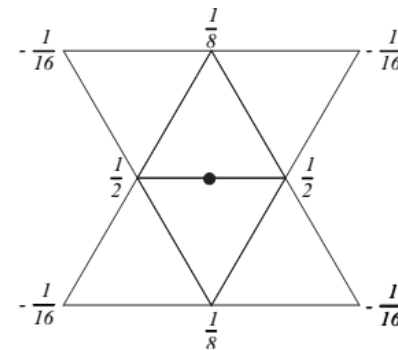
- Catmull-Clark is de facto standard in CG industry
  - Quad meshes can naturally represent two principal curvature directions

# Other subdivision schemes

- four-point method
  - Passes through CPs (interpolating)
    - $\leftrightarrow$  approximating
  - Cannot be represented as polynomials
  - $C^1$ -continuous
  - Surface version: butterfly method

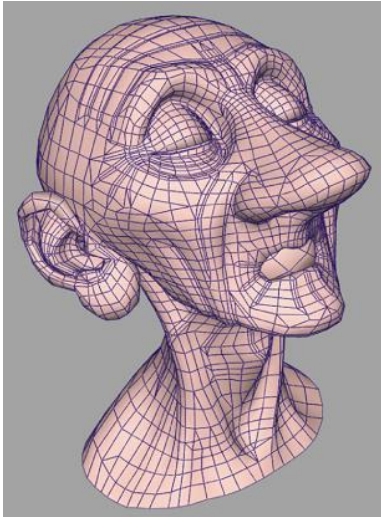


- Many more variants
  - Kobbelt's method
  - $\sqrt{3}$ -method
  - etc...



# Geri's Game (Pixar, 1997)

- First film using subdivision surfaces
- Previously (Toy Story), tedious modeling work using B-splines

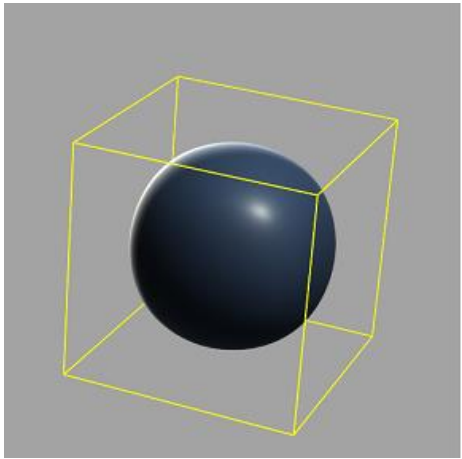


<https://www.youtube.com/watch?v=9IYRC7g2ICg>

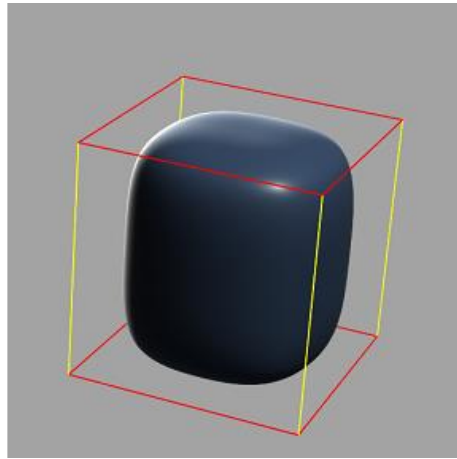


# Controlling smoothness

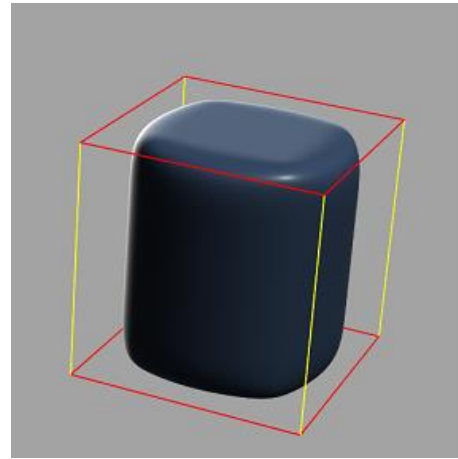
- Can represent sharp edges by altering subdivision rules



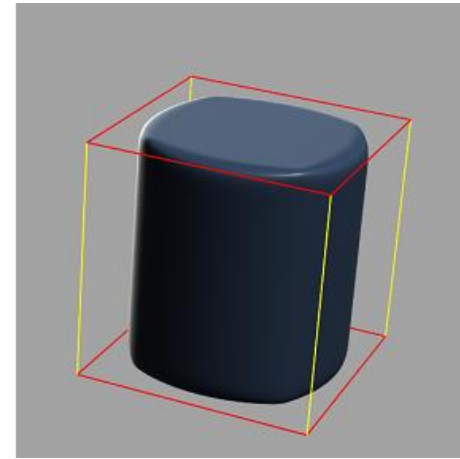
sharpness=0



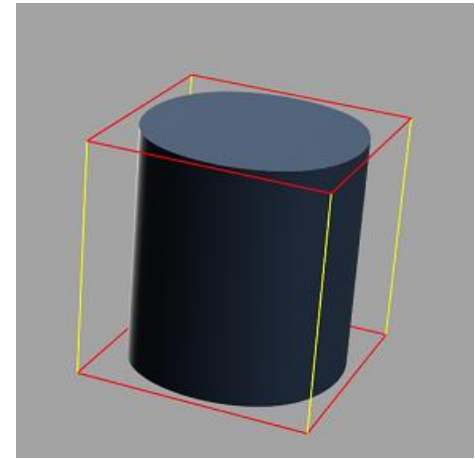
sharpness=1



sharpness=2



sharpness=3



sharpness= $\infty$

# Controlling smoothness

- Can represent sharp edges by altering subdivision rules



# Resources for learning subdivision techniques

- Smooth Subdivision Surfaces Based on Triangles [Loop, **MSc. Thesis** 87]
  - Thorough & visual explanation of literature (Doo-Sabin & Catmull-Clark)
  - Some known errors:  
<http://www.cs.berkeley.edu/~sequin/CS284/TEXT/LoopErrata.txt>
- Subdivision for Modeling and Animation [SIG00 Course]
  - Most famous survey, but a little arcane
  - <http://www.cs.nyu.edu/~dzorin/sig00course/>
- OpenSubdiv from research to industry adoption [SIG13 Course]
  - More recent topics
  - <http://dx.doi.org/10.1145/2504435.2504451>

# Halfedge data structure



# Mesh representation using vertex & face lists

	OFF file format			
Geometry data	OFF			
	8 6 0			
	-0.5 -0.5 0.5			
	0.5 -0.5 0.5			
	-0.5 0.5 0.5			
	0.5 0.5 0.5			
	-0.5 0.5 -0.5			
	0.5 0.5 -0.5			
	-0.5 -0.5 -0.5			
	0.5 -0.5 -0.5			
Topology data	4 0 1 3 2			
	4 2 3 5 4			
	4 4 5 7 6			
	4 6 7 1 0			
	4 1 7 5 3			
	4 6 0 2 4			

← #vertices, #faces

← xyz coord of 0<sup>th</sup> vertex

•

•

•

← xyz coord of 7<sup>th</sup> vertex

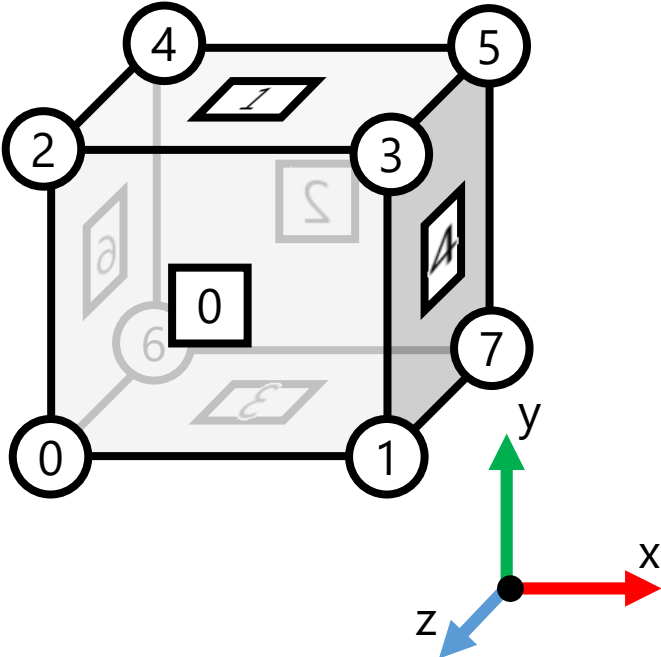
← 0<sup>th</sup> face's #vertices and vertex indices

•

•

•

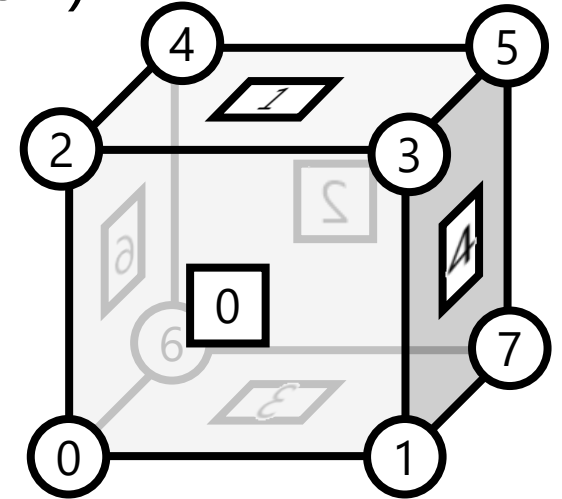
← 6<sup>th</sup> face's #vertices and vertex indices



# Mesh representation using vertex & face lists

- Info needed during mesh processing (e.g. subdivision)

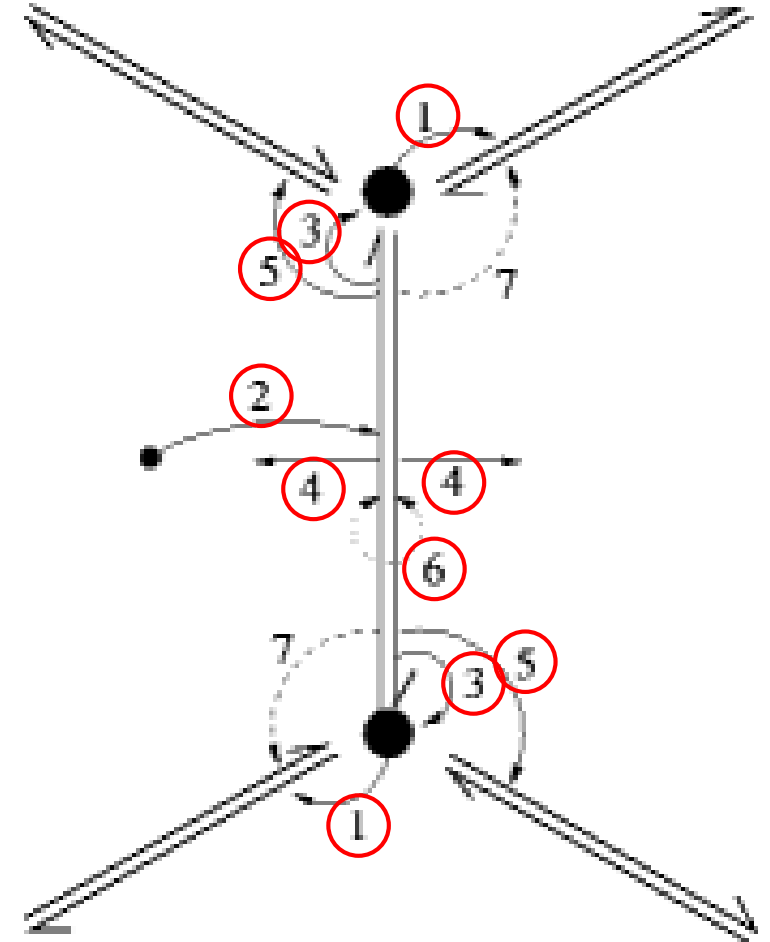
- Set of faces around a vertex
- Set of faces adjacent to a face
- Vertices at an edge's endpoints
- Faces at both sides of an edge
- etc...



- Can be stored as "array or arrays", but consumes more memory☹

# Halfedge data structure

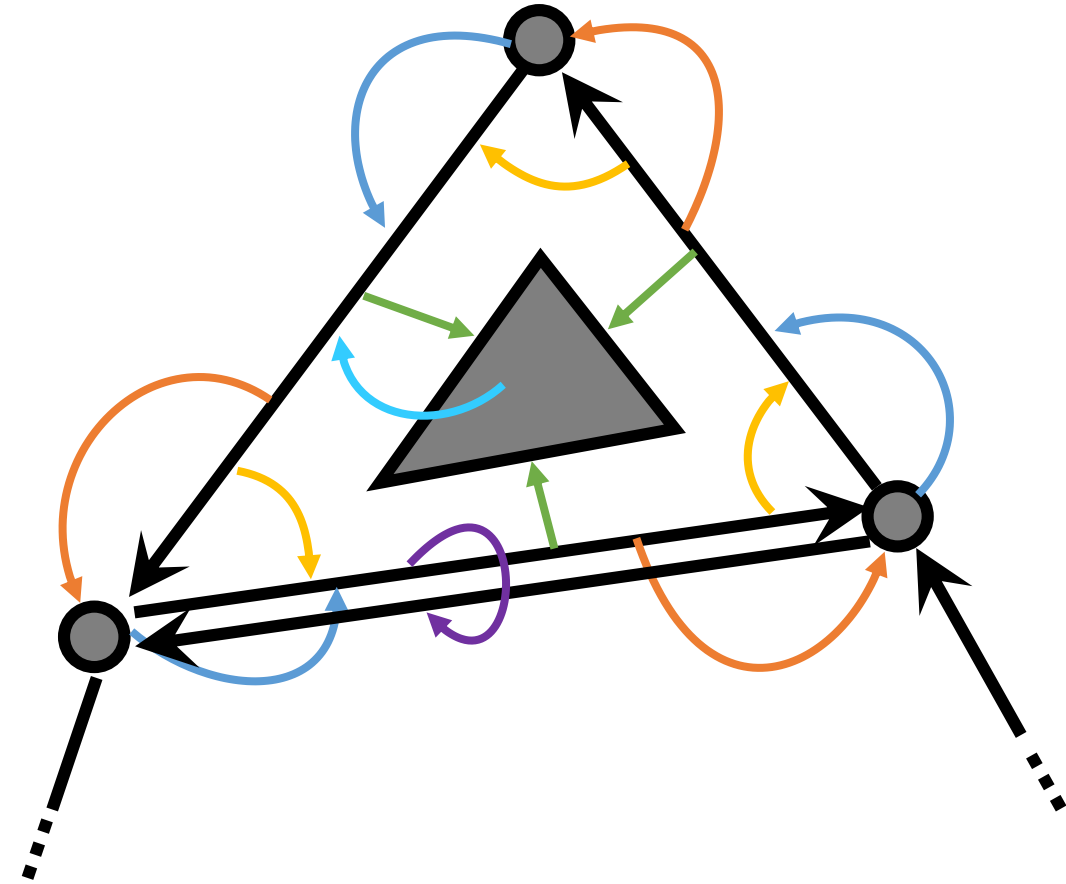
- Store link information:
  - (1) Vertex  $\rightarrow$  One of halfedges emanating from it
  - (2) Face  $\rightarrow$  One of halfedges composing it
  - (3) Halfedge  $\rightarrow$  Vertex that it points to
  - (4) Halfedge  $\rightarrow$  Face that it belongs to
  - (5) Halfedge  $\rightarrow$  Next halfedge
  - (6) Halfedge  $\rightarrow$  Halfedge opposite to it
- Circling around a face:  
(2)  $\rightarrow$  (5)  $\rightarrow$  (5)  $\rightarrow$  ...
- Circling around a vertex:  
(1)  $\rightarrow$  (6)  $\rightarrow$  (5)  $\rightarrow$  (6)  $\rightarrow$  (5)  $\rightarrow$  ...



<http://www.openmesh.org/>

# When a new face is added

- Generate halfedges
- Link vertex to halfedge (1)
- Link halfedge to vertex (3)
- Link halfedge to next halfedge (5)
- Link halfedges to face (4)
- Link face to halfedge (2)
- Link halfedge to its opposite halfedge if such exists (6)



# Papers

- Recursively generated B-spline surfaces on arbitrary topological meshes [Catmull,Clark,CAD78]
- A 4-point interpolatory subdivision scheme for curve design [Dyn,Levin,CAGD87]
- A butterfly subdivision scheme for surface interpolation with tension control [Dyn,Levine,Gregory,TOG90]
- Sqrt(3)-subdivision [Kobbelt,SIGGRAPH00]
- Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values [Stam,SIGGRAPH98]
- Interactive multiresolution mesh editing [Zorin,Schroder,Sweldens,SIGGRAPH97]
- Interpolating subdivision for meshes with arbitrary topology [Zorin,Schroder,Sweldens,SIGGRAPH96]