Curves and Surfaces 2

Computer Graphics

Lecture 17

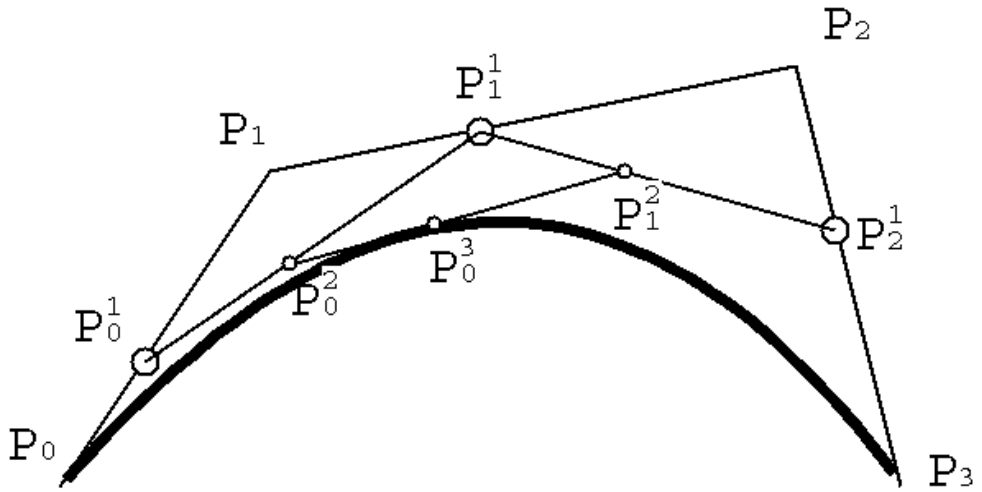
Taku Komura

Today

- **More about Bezier and Bsplines**
  - **de Casteljau's algorithm**
  - BSpline : General form
  - de Boor's algorithm
  - Knot insertion
- NURBS
- Subdivision Surface

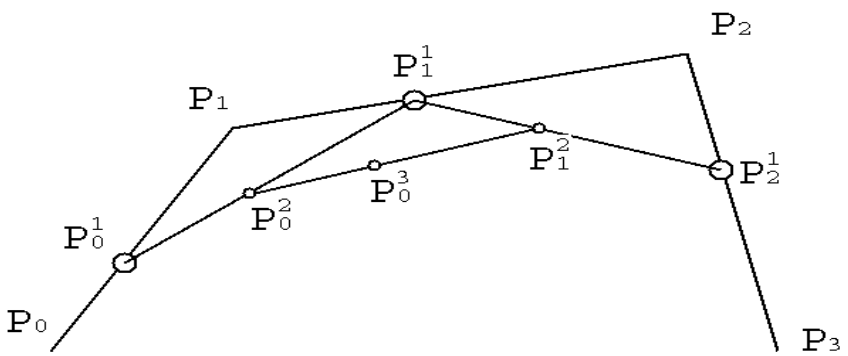
De Casteljau's Algorithm

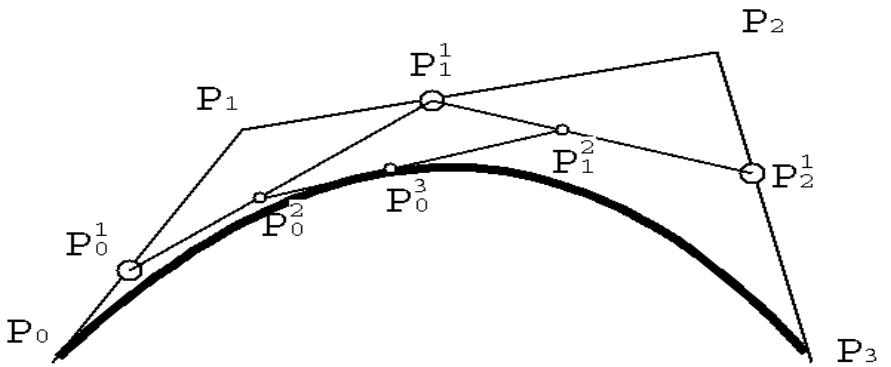
- A method to evaluate (sample points in) or draw the Bezier curve
- The Bezier curve of any degree can be handled
- A precise way to evaluate the curves



de Casteljau's Algorithm

- Given the control points $P_1, \ldots, P_n$ and the parameter value $0 \leq t \leq 1$,
- Repeat the following procedure
  - Set $P^r{}_i(t) = (1-t)\ P^{r-1}{}_i(t) + tP^{r-1}{}_{i+1}(t)$
  - $P^0{}_i(t) = P_i$
  - Then, $P^n{}_0(t)$ is the point with parameter value t on the Bezier curve



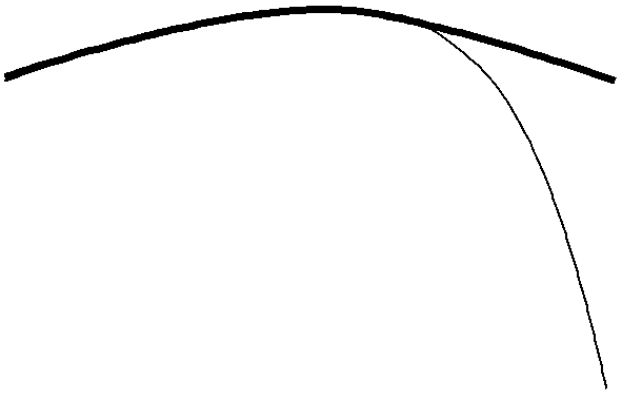


Why does this result in the polynomial?

- Let's think of a quadratic case that there are three points P_0, P_1, P_2.
- $P^1{}_0(t) = (1-t)\,P_0 + tP_1$
- $P^1{}_1(t) = (1-t)\,P_1 + tP_2$
- $P^2{}_0(t) = (1-t)P^1{}_0 + tP^1{}_1$
- By inserting the first two equations into the third one, we obtain
- $P^2{}_0(t) = (1-t)^2P_0 + 2t(1-t)P_1 + t^2P_2$
- Doing this for 4 control points will give the cubic formula I presented last week

Why do we need this?

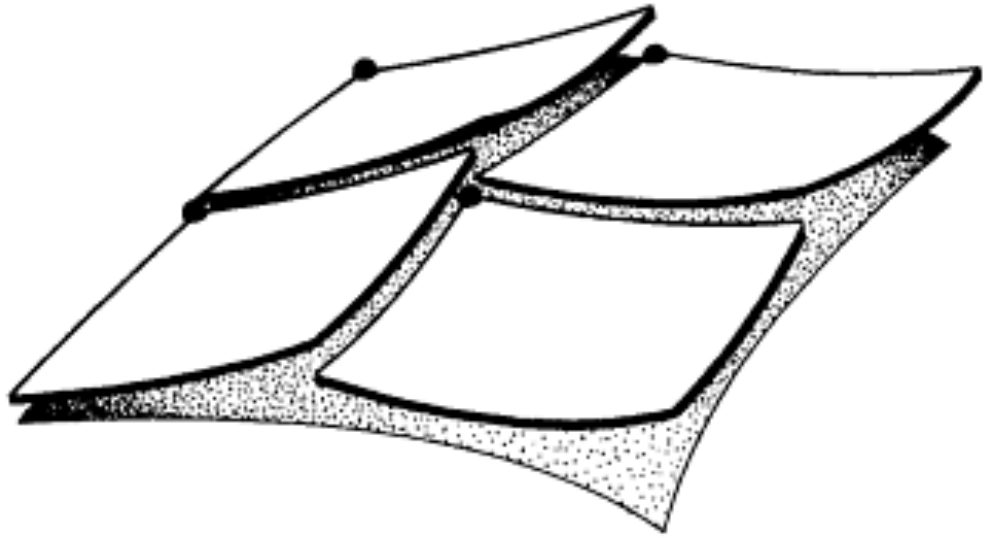
- The explicit representation (monomial form) that I presented last week can result in some instability
- Say the control points are randomly changed for 0.001.
- The curve computed by the de Casteljau's algorithm stays almost the same.
- The curve by the polynomial basis form can deviate from the original curve if the degree is high



Connecting many Bezier Patches in the polynomial form

- The same story applies to surfaces
- The degree of surface can easily go high, as they are the multiplication of two curves
- Bicubic → 6
- The error of 16

control points will be accumulated



Today

- More about Bezier and Bsplines
  - de Casteljau's algorithm
  - **BSpline : General form**
  - de Boor's algorithm
  - Knot insertion
- NURBS
- Subdivision Surface

B-Spline : A General Form

A Bspline is a parametric curve composed of a linear combination of basis B-splines $B_{i,n}$

$$p(t) = \sum_{i=0}^{m} P_i B_{i,k}(t)$$

P_i ($i=0,\ldots,m$) the control points

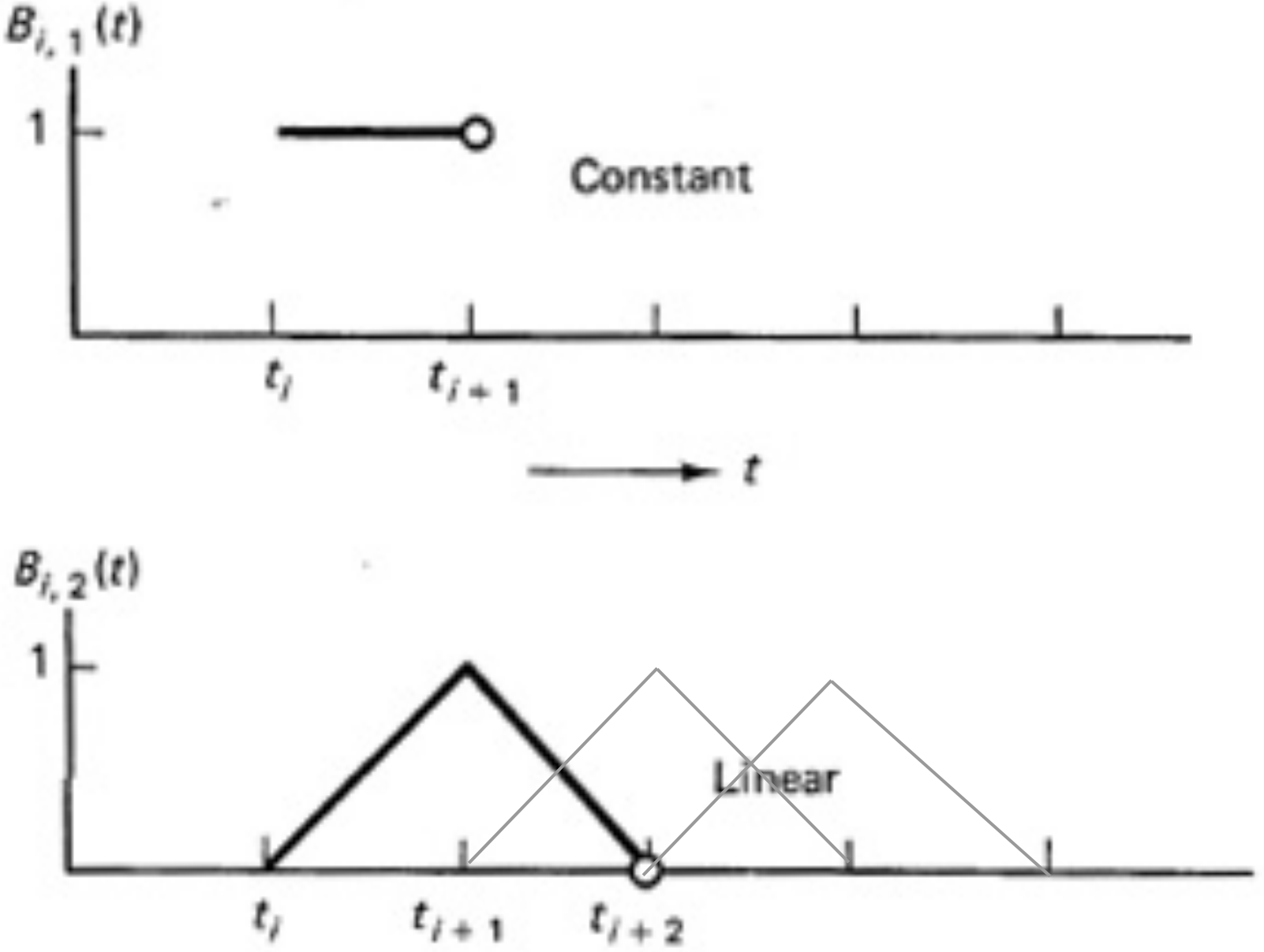
Knots: $t_0 \le t_1 \le \ldots \le t_{k+m}$ - subdivide the domain of the B-spline curve into a set of knot spans $[t_i, t_{i+1})$

The B-splines can be defined by

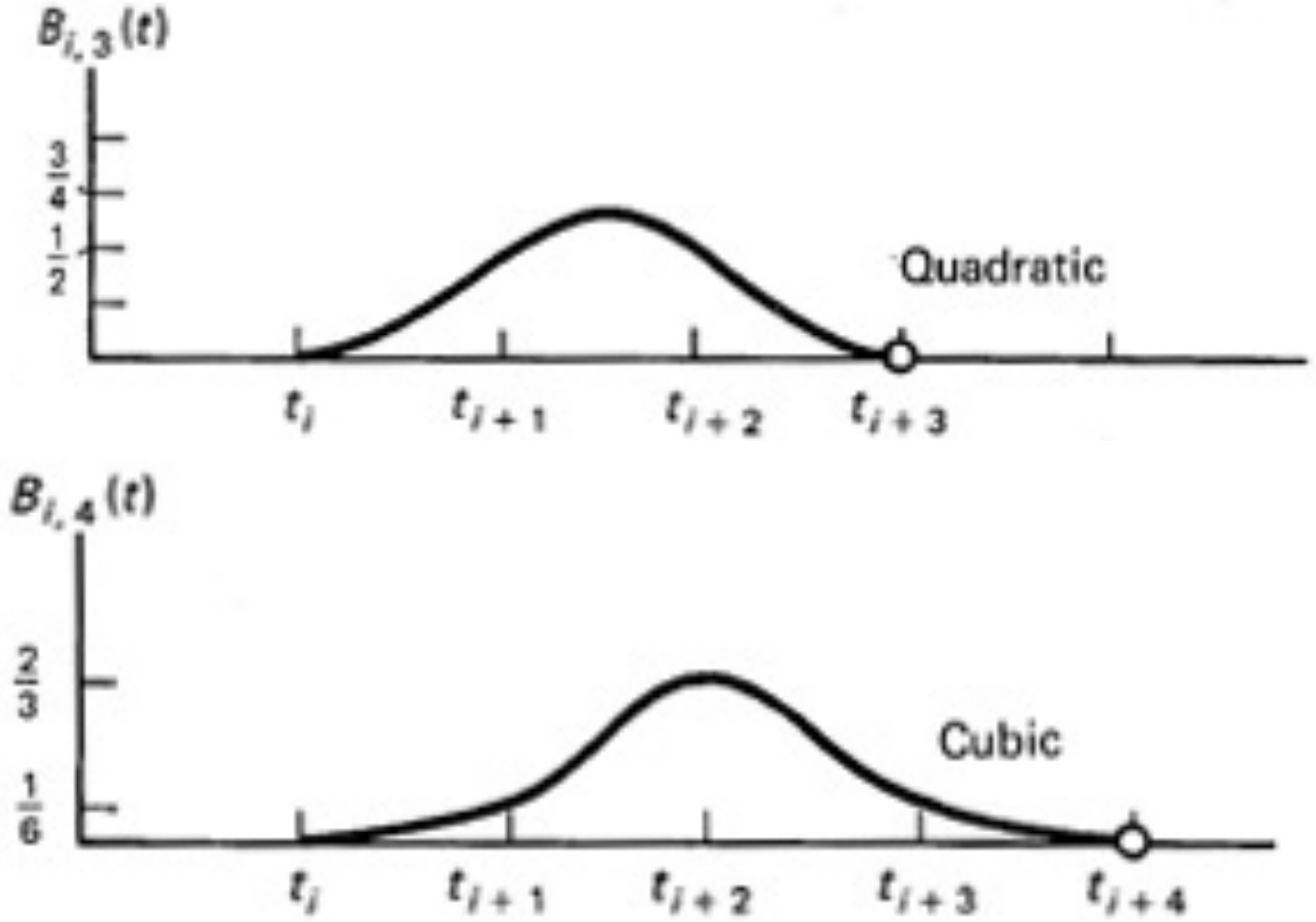
$$B_{i,1}(t) = \begin{cases} 1, & t_i \le t < t_{i+1} \\ 0, & \text{otherwise} \end{cases}$$

$$B_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k-1} - t_i} B_{i+1,k-1}(t)$$

Bspline Basis

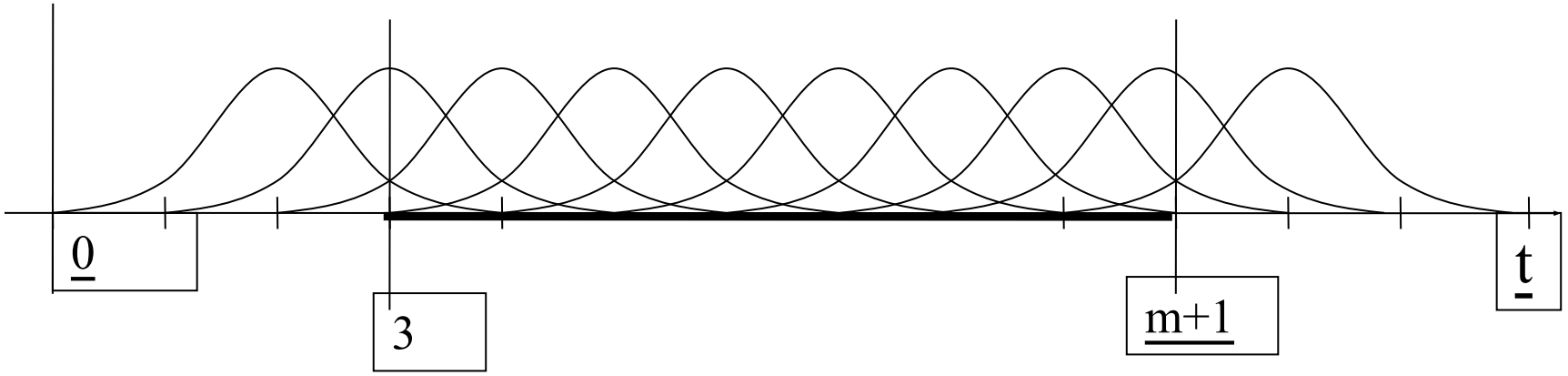


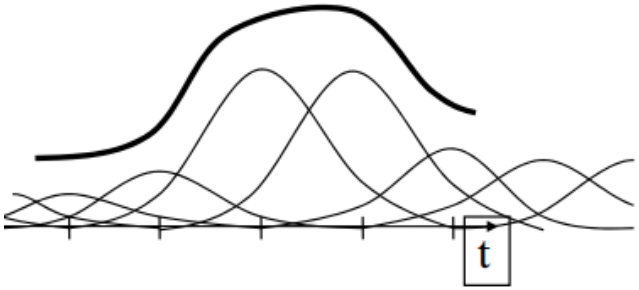
Bspline Basis (2)



Producing Curves by B-Splines

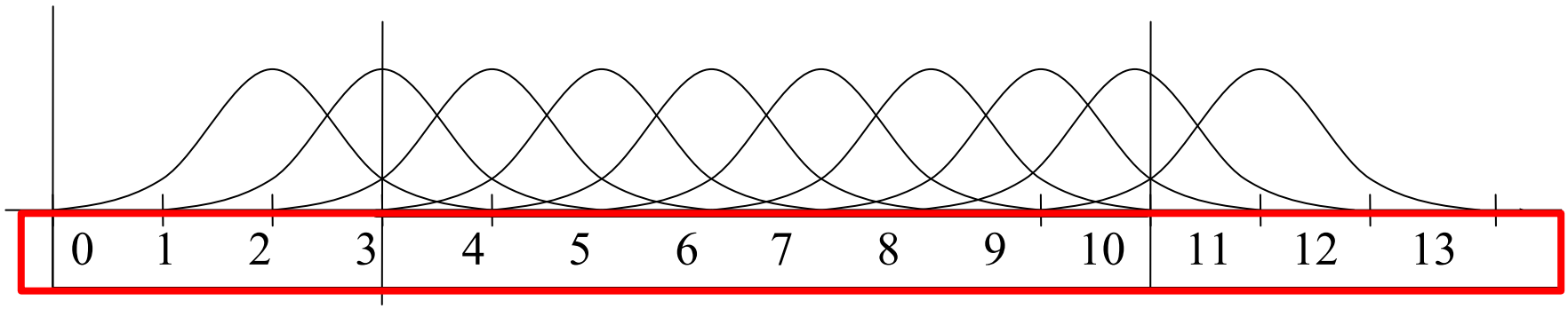
- The basis functions are multiplied to the control points and to define arbitrary curves





Knots

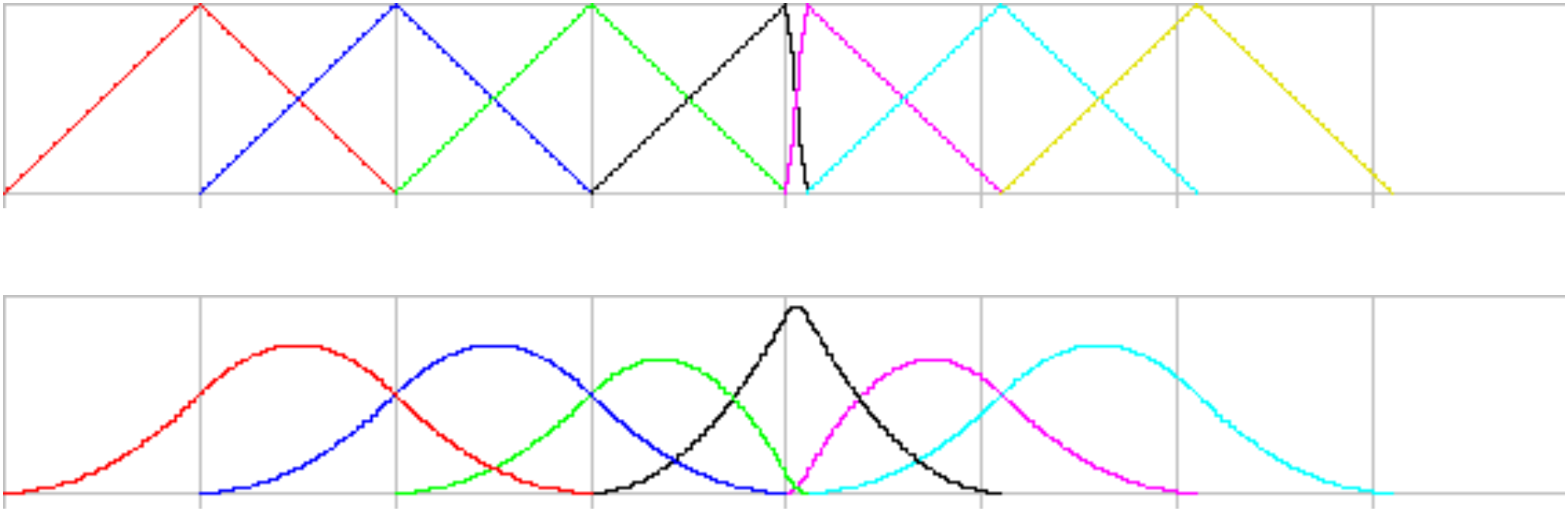
- The knots produce a vector that defines the domain of the curve
- The knots must be in the increasing order
- But not necessarily uniform
- If uniformly sampled and the degree is 3 uniform cubic bspline



knots

Knots

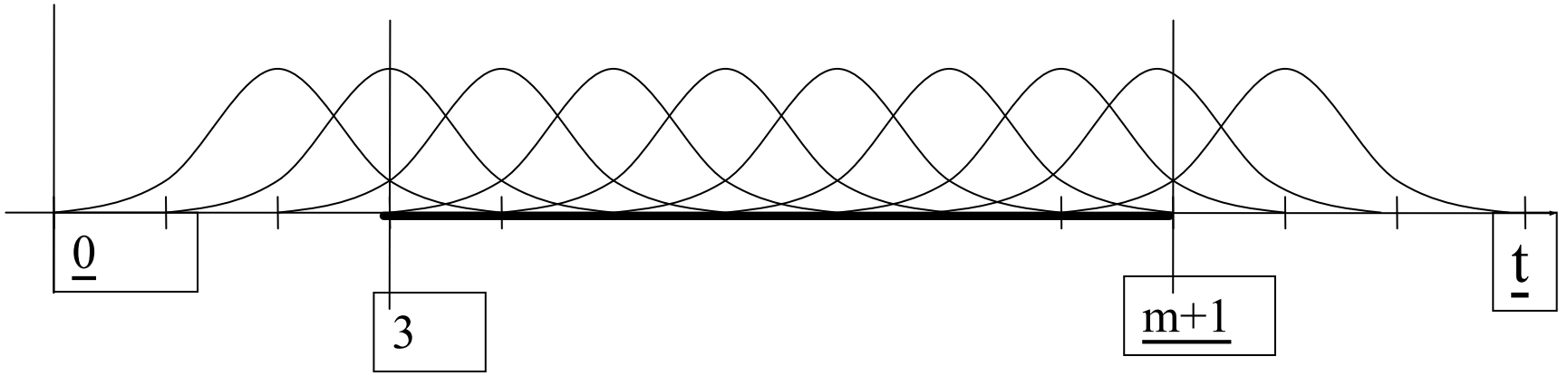
Here is an example of non-uniform knots



http://i33www.ira.uka.de/applets/mocca/html/noplugin/BSplineBasis/AppBSplineBasis/index.html

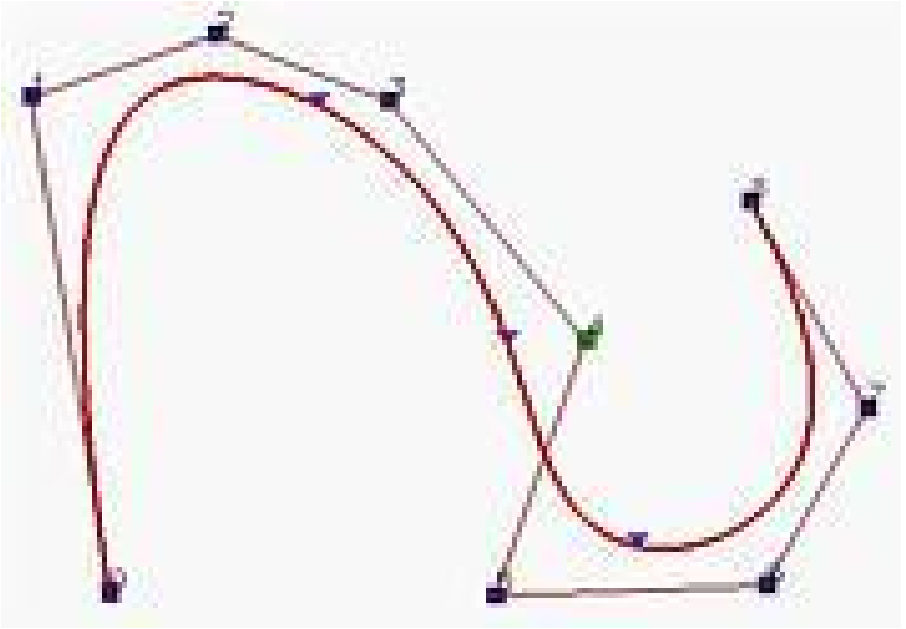
Some Terms

- Order *k: the number of control points that affect the sampled value*
- Degree *k-1 (the basis functions are polynomials of degree k-1)*
- Control points ***Pi*** **(***i=0,...,m***)**
- Knots : *tj*, (*j=0,..., n*)
- An important rule : n – m = k
- The domain of function $t_{k-1} \leqq t \leqq t_{m+1}$
  - Below, k = 4, m = 9, domain, $t_3 \leqq t \leqq t_{10}$



Clamped Bsplines

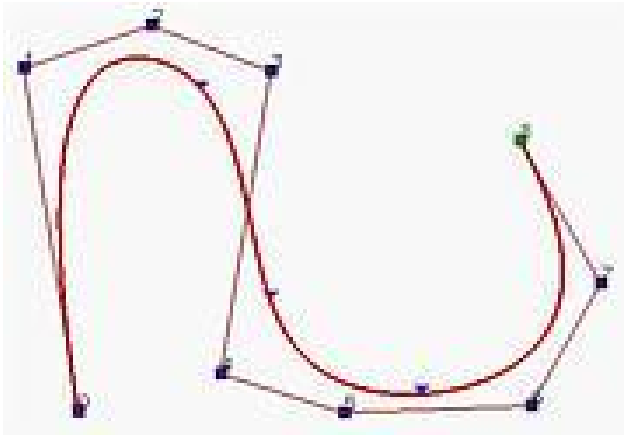
- The first and last knot values are repeated with multiplicity equal to the order (= degree + 1)
- The end points pass the control point
- For cubic Bsplines, the multiplicity of the first / last knots must be 4 (repeated four times)

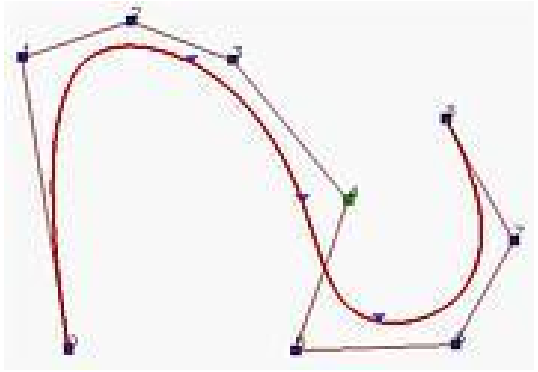


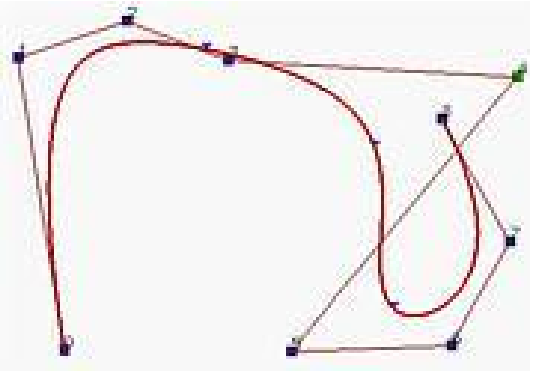
Controlling the shape of B-splines

- Moving the control points is the most obvious way to control B-spline curves
- Changing the position of control point **P***i* only affects the local region

http://www.ibiblio.org/e-notes/Splines/basis.html







Today

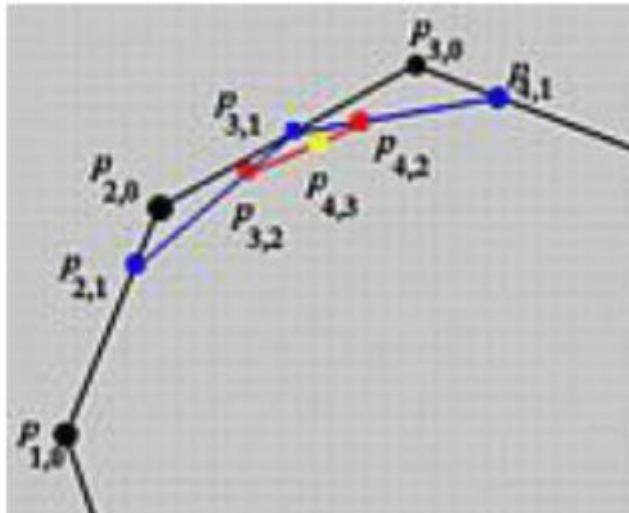
- **More about Bezier and Bsplines**
  - de Casteljau's algorithm
  - BSpline : General form
  - **de Boor's algorithm**
  - **Knot insertion**
- NURBS
- Subdivision Surface

De Boor's Algorithm

- A B-spline version of the de Casteljau's algo
- A precise method to evaluate the curve
- Starting from the control points $P_0, \ldots, P_n$ and parameter value t, recursively solve the following problem

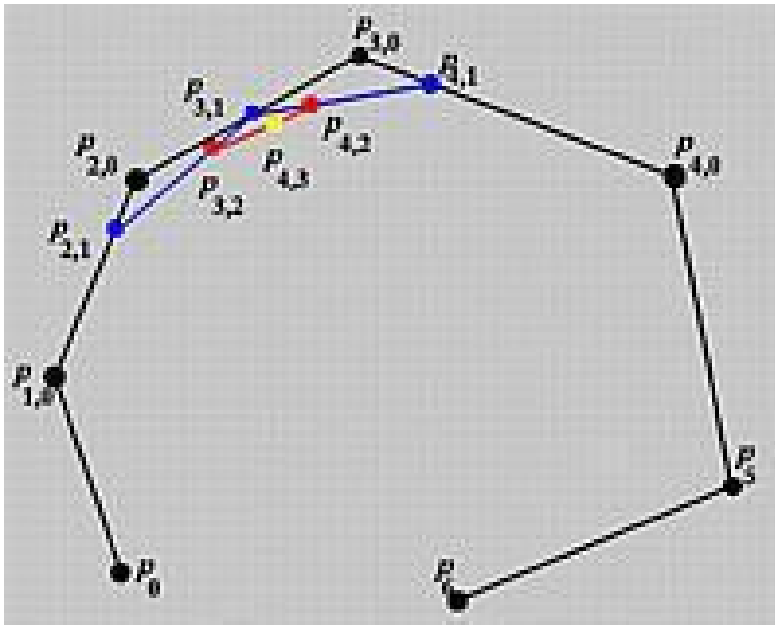
$$\mathbf{P}_i^r = (1 - a_{i,r})\mathbf{P}_{i-1}^{r-1} + a_{i,r}\mathbf{P}_i^{r-1}$$

$$a_{i,r} = \frac{t - t_i}{t_{i+k-1-r} - t_i}$$

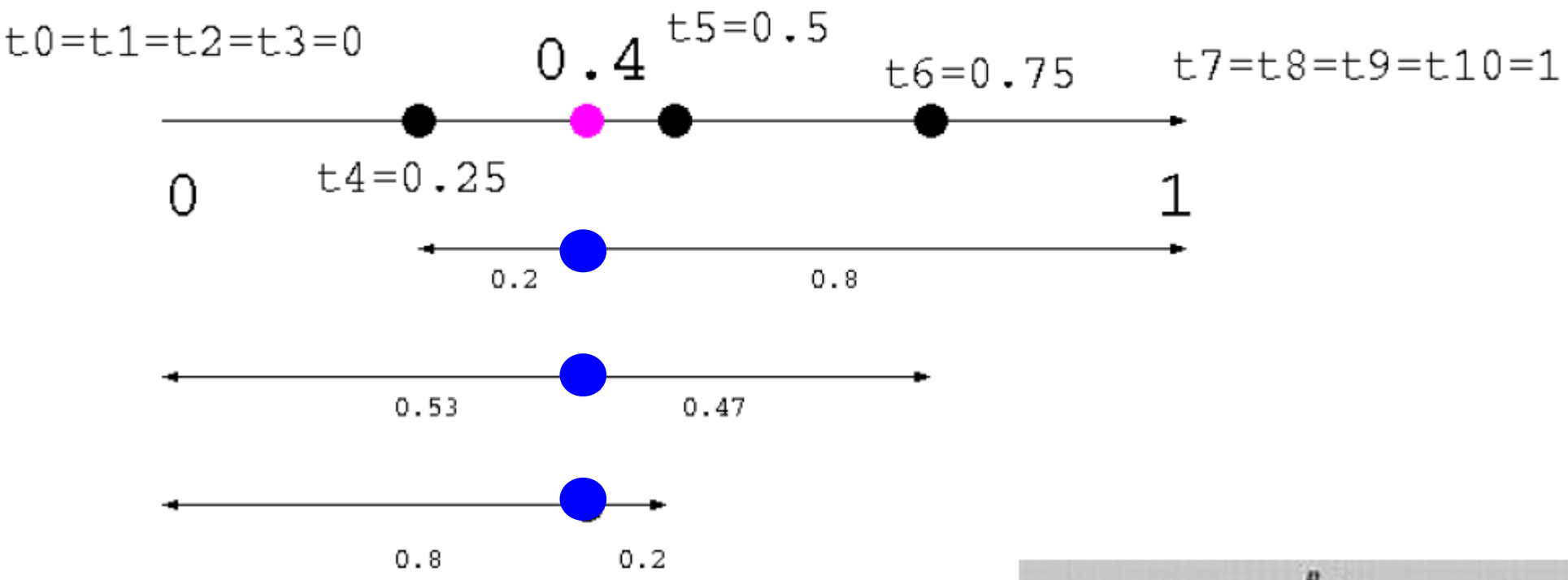


Example

- Assume we have a cubic B-spline whose knot vector is {0, 0, 0, 0, 0.25, 0.5, 0.75, 1, 1, 1, 1}
- Let's compute a point at t = 0.4
- Then, $t_4 < t < t_5$, and the control points that affect the final position are P4, P3, P2, P1



Example



$$a_{4,1} = (t - t_4)/(t_{4+3} - t_4) = 0.2$$

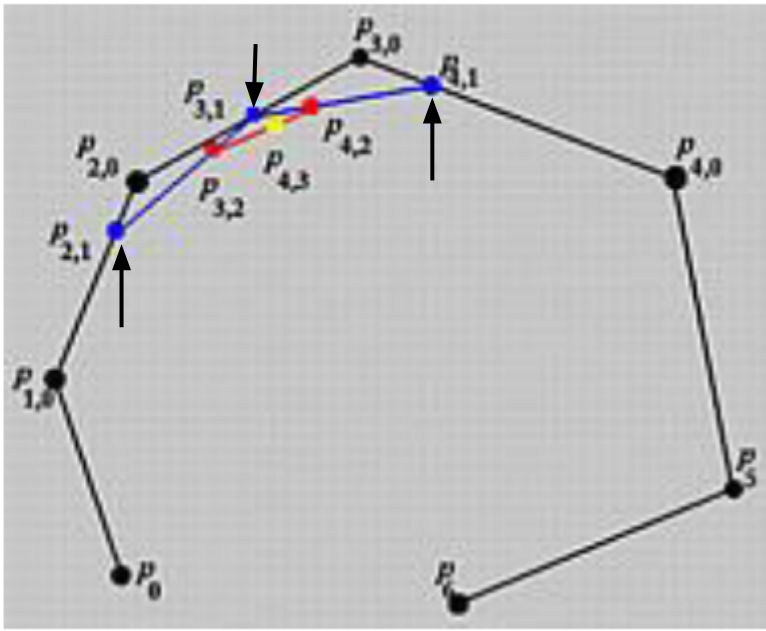
$$a_{3,1} = (t - t_3)/(t_{3+3} - t_3) = 0.53$$

$$a_{2,1} = (t - t_2)/(t_{2+3} - t_2) = 0.8$$

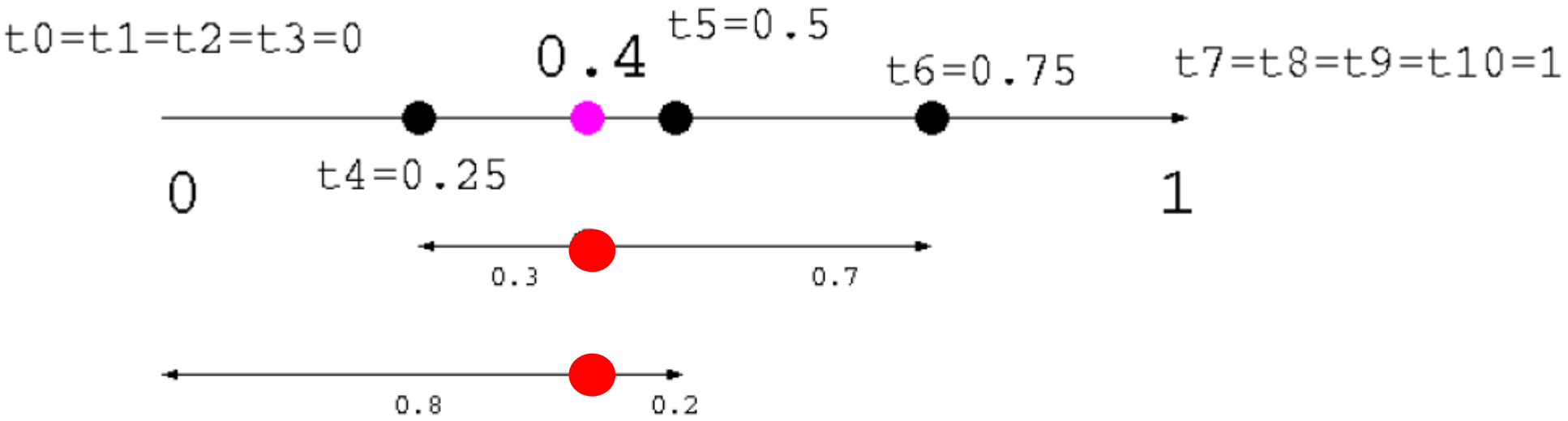
$$\mathbf{P}_{4,1} = (1 - a_{4,1})\mathbf{P}_{3,0} + a_{4,1}\mathbf{P}_{4,0}$$

$$\mathbf{P}_{3,1} = (1 - a_{3,1})\mathbf{P}_{2,0} + a_{3,1}\mathbf{P}_{3,0}$$

$$\mathbf{P}_{2,1} = (1 - a_{4,1})\mathbf{P}_{1,0} + a_{2,1}\mathbf{P}_{2,0}$$



Example

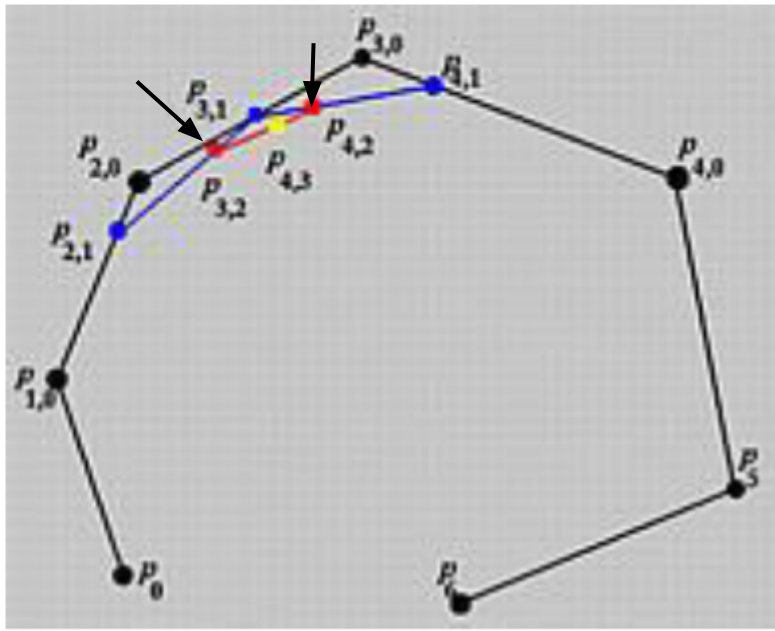


$$a_{4,2} = (t - t_4) / (t_{4+3-1} - t_4) = 0.3$$

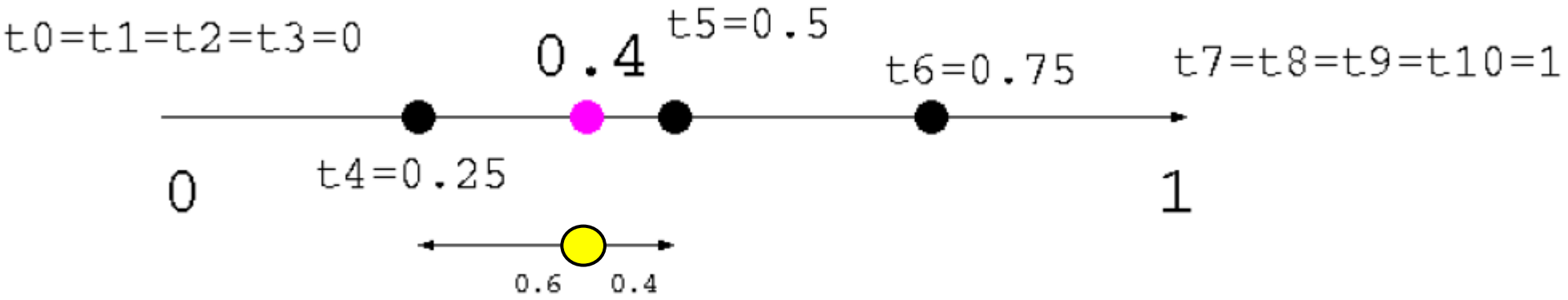
$$a_{3,2} = (t - t_3) / (t_{3+3-1} - t_3) = 0.8$$

$$\mathbf{P}_{4,2} = (1 - a_{4,2})\mathbf{P}_{3,1} + a_{4,2}\mathbf{P}_{4,1}$$

$$\mathbf{P}_{3,2} = (1 - a_{3,2})\mathbf{P}_{2,1} + a_{3,2}\mathbf{P}_{3,1}$$



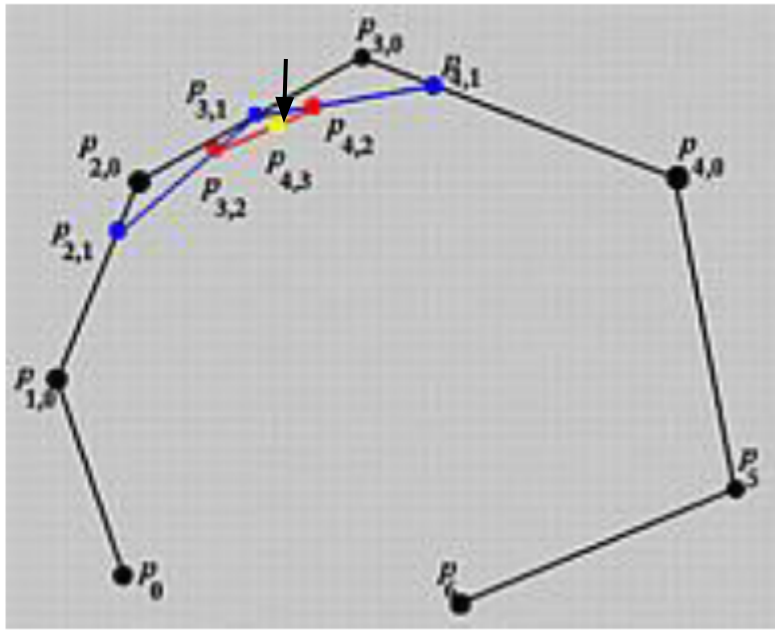
Example



$$a_{4,3} = (u - u_4)/(u_{4+3-2} - u_4) = 0.6$$

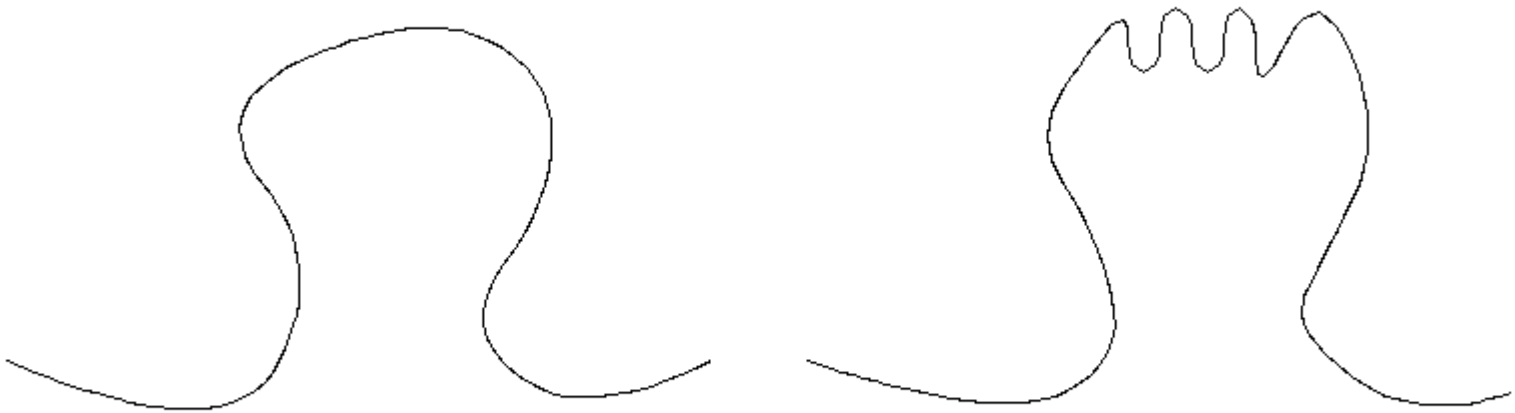
$$\mathbf{P}_{4,3} = (1 - a_{4,3})\mathbf{P}_{3,2} + a_{4,2}\mathbf{P}_{4,2}$$

$$= 0.4\mathbf{P}_{3,2} + 0.6\mathbf{P}_{4,2}$$



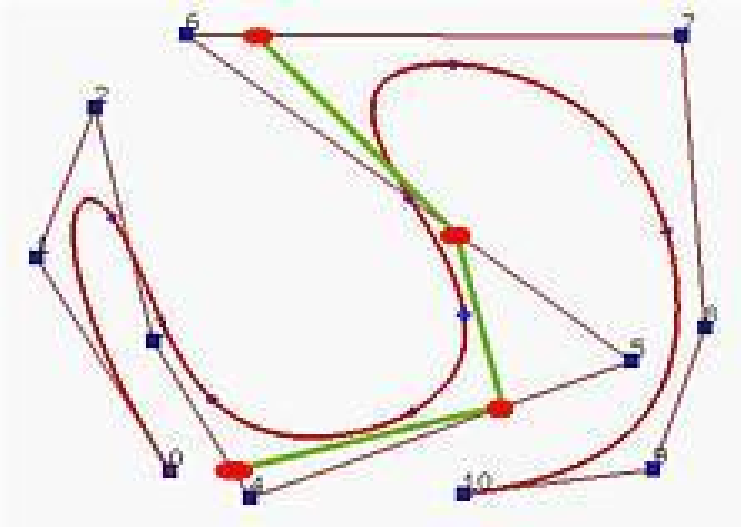
What if you want to edit some details?

- You might want to add some high resolution details at a particular area while keeping the rest the same



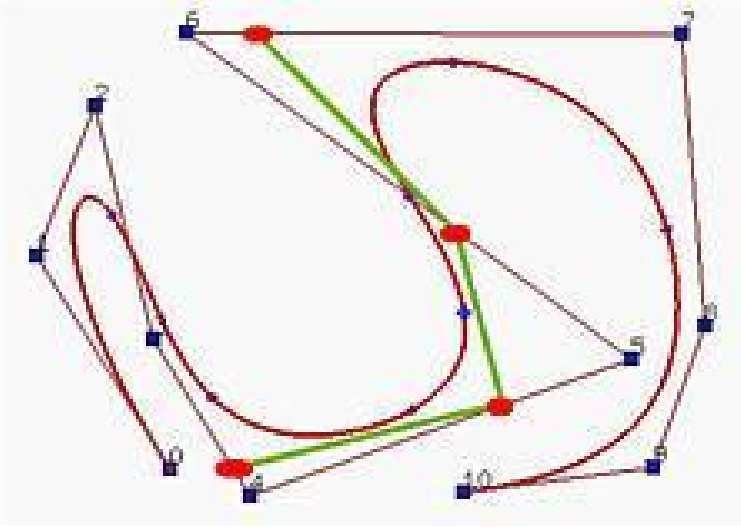
Knot insertion

- We can do this by knot Insertion
  - increase the resolution for some area
- New knots and control points can be added without changing the shape of the curve
- Both the knots and control points are going to increase



Knot insertion

- For a curve of degree f, we remove f-1 points and add f points
- i.e., for a cubic B-spline, remove 2 points and add 3 points



Knot insertion

- If the new knot t is inserted into the span $[t_j, t_{j+1})$,

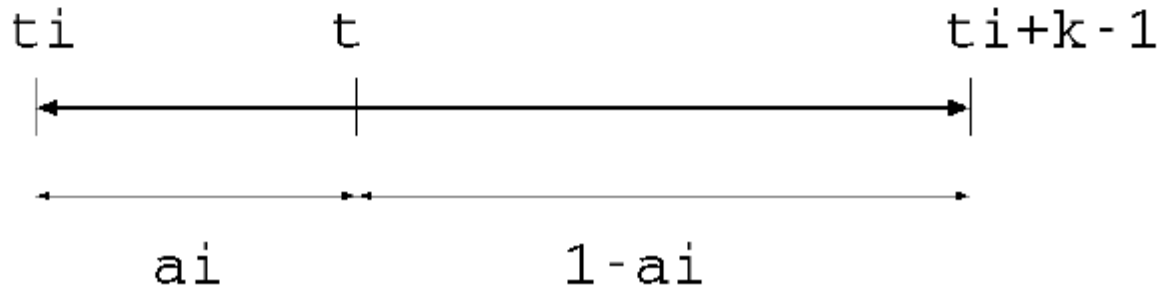
the new control points can be computed by

$$\mathbf{Q_i} = (1 - a_i)\mathbf{P_{i-1}} + a_i\mathbf{P_i}$$

where **Qi** is the new control point and a_i is computed by

$$a_i = \frac{t - t_i}{t_{i+k-1} - t_i} \quad \text{for } j\text{-}k + 2 \le i \le j$$

The polyline $\mathbf{P}_{j-k+1}$, $\mathbf{P}_{j-k+2}$, ..., $\mathbf{P}_{j-1}$, $\mathbf{P}_j$ is replaced with $\mathbf{P}_{j-k+1}$, $\mathbf{Q}_{j-k+2}$, ..., $\mathbf{Q}_{j-1}$, $\mathbf{Q}_j$, $\mathbf{P}_j$.

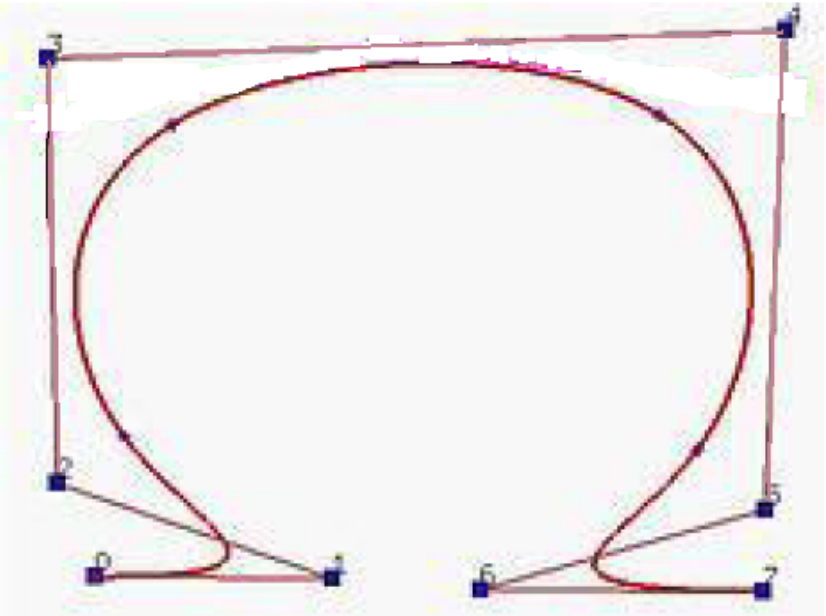


Example

- A Bspline curve of degree 3 (k=4) having the following knots
- t=0.5 inserted

t_0 to t_3	t_4	t_5	t_6	t_7	t_8 to t_{11}
0	0.2	0.4	0.6	0.8	1

t_0 to t_3	t_4	t_5	t_6	t_7	t_8	t_9 to t_{12}
0	0.2	0.4	**0.5**	0.6	0.8	1



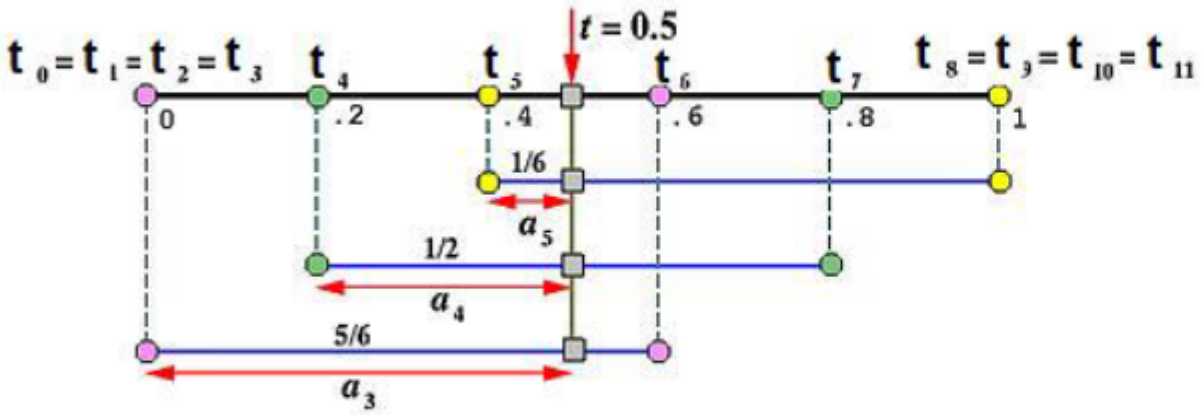
http://i33www.ira.uka.de/applets/mocca/html/noplugin/curves.html

Example

- A bspline curve of degree 3 (k=4) having the following knots
- t=0.5 inserted

t_0 to t_3	t_4	t_5	t_6	t_7	t_8 to t_{11}
0	0.2	0.4	0.6	0.8	1

t_0 to t_3	t_4	t_5	t_6	t_7	t_8	t_9 to t_{12}
0	0.2	0.4	**0.5**	0.6	0.8	1



$$a_5 = \frac{t - t_5}{t_8 - t_5} = \frac{0.5 - 0.4}{1 - 0.4} = \frac{1}{6}$$

$$a_4 = \frac{t - t_4}{t_7 - t_4} = \frac{0.5 - 0.2}{0.8 - 0.2} = \frac{1}{2}$$

$$a_3 = \frac{t - t_3}{t_6 - t_3} = \frac{0.5 - 0}{0.6 - 0} = \frac{5}{6}$$

Example

http://geom.ibds.kit.edu/applets/mocca/html/noplugin/IntBSpline/AppInsertion/index.html

- A bspline curve of degree 3 (k=4) having the following knots
- t=0.5 inserted

t_0 to t_3	t_4	t_5	t_6	t_7	t_8 to t_{11}
0	0.2	0.4	0.6	0.8	1

t_0 to t_3	t_4	t_5	t_6	t_7	t_8	t_9 to t_{12}
0	0.2	0.4	**0.5**	0.6	0.8	1

$$a_5 = \frac{t - t_5}{t_8 - t_5} = \frac{0.5 - 0.4}{1 - 0.4} = \frac{1}{6}$$

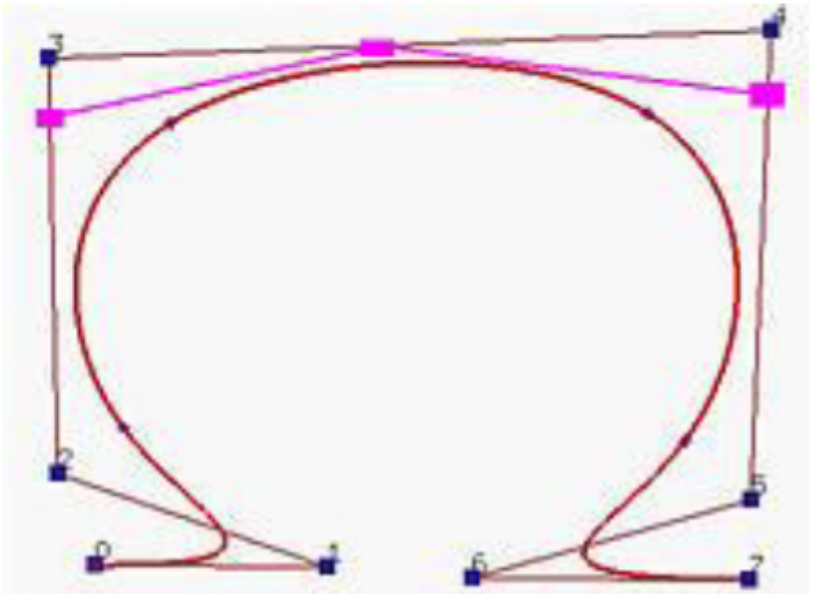
$$a_4 = \frac{t - t_4}{t_7 - t_4} = \frac{0.5 - 0.2}{0.8 - 0.2} = \frac{1}{2}$$

$$a_3 = \frac{t - t_3}{t_6 - t_3} = \frac{0.5 - 0}{0.6 - 0} = \frac{5}{6}$$

$$\mathbf{Q}_5 = \left(1 - \frac{1}{6}\mathbf{P}_4\right) + \frac{1}{6}\mathbf{P}_5$$

$$\mathbf{Q}_4 = \left(1 - \frac{1}{2}\mathbf{P}_3\right) + \frac{1}{2}\mathbf{P}_4$$

$$\mathbf{Q}_3 = \left(1 - \frac{5}{6}\mathbf{P}_2\right) + \frac{5}{6}\mathbf{P}_3$$



http://i33www.ira.uka.de/applets/mocca/html/noplugin/curves.html

Summary of B-splines

- Knot vector defines the domain
- Evaluation by de Boor's algorithm
- Controlling the shape by the control points
- Clamping the points by increasing the multiplicity of the knots at the end points
- Increase the resolution by knot insertion

Today

- More about Bezier and Bsplines
  - de Casteljau's algorithm
  - BSpline : General form
  - de Boor's algorithm
  - Knot insertion
- **NURBS**
- Subdivision Surface

NURBS (Non-uniform rational B-spline)

- Standard curves/surface representation in computer aided design

$$C(t) = \frac{\sum_{i=0}^{n} B_{i,k}(t) w_i \mathbf{P}_i}{\sum_{i=0}^{n} B_{i,k}(t) w_i}$$

Pi : control points

$B_{i,k}$: Bspline basis of order k

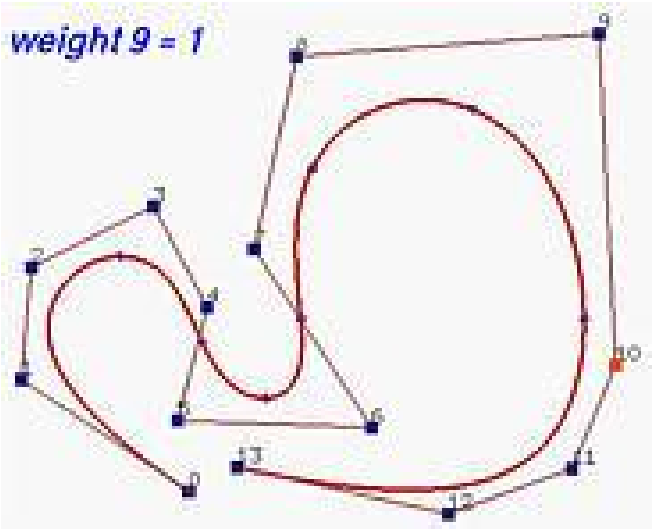
wi : weights

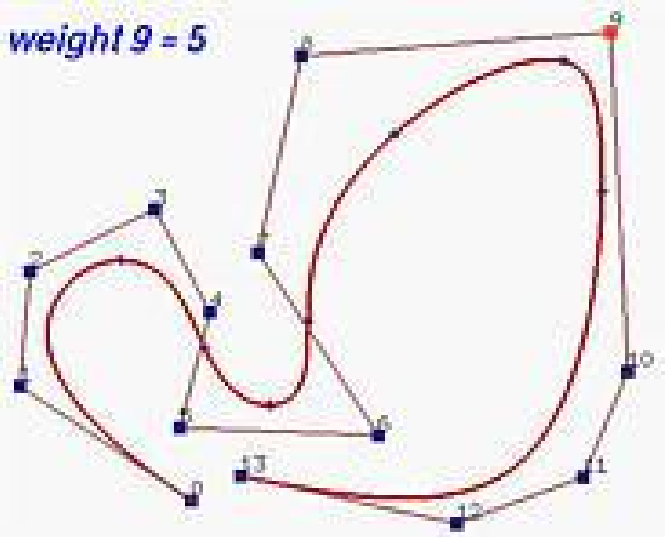
Benefits of using NURBS

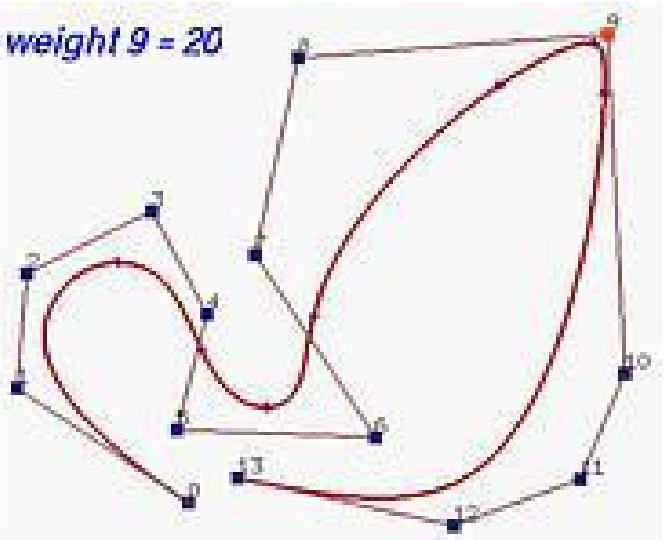
- More degrees of freedom to control the curve (can control the weights)
- Invariant under perspective transformation
  - Can project the control points onto the screen and draw the NURBS on the screen
    - Don’t need to apply the perspective transformation to all the sample-points on the curve
- Can model conic sections such as circles, ellipses and hyperbolas

Example of changing weights

- Increasing the weight will bring the curve closer to the corresponding control point





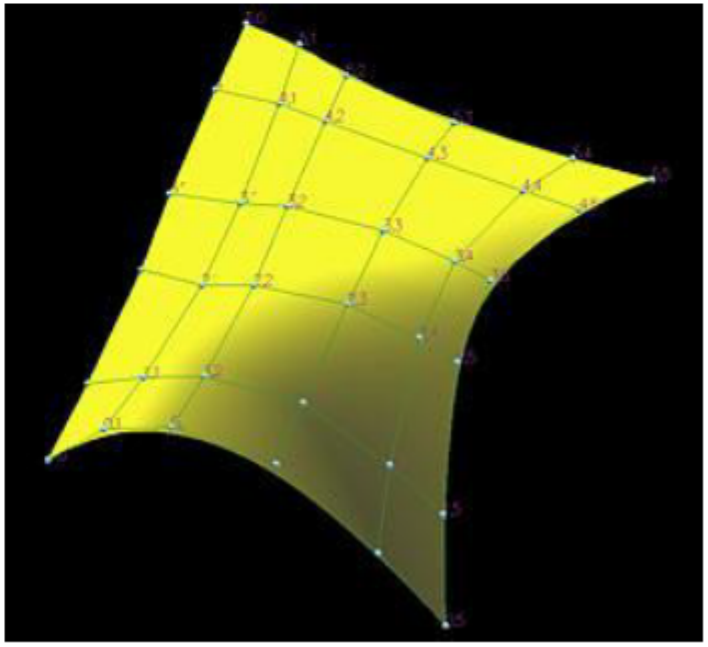


Bspline Surfaces

- Given the following information:
- a set of m+1 rows and n+1 control points $\mathbf{p}i,j$, where 0 <= i <= m and 0 <= j <= n;
- Corresponding knot vectors in the u *and* v direction,

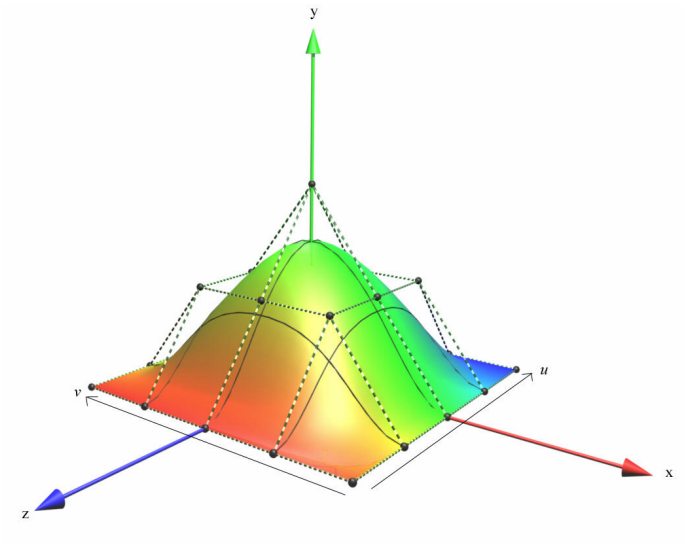
$$p(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} B_{i,p}(u)B_{j,q}(v)\mathbf{P}_{i,j} \text{ : non - rational B - spline}$$

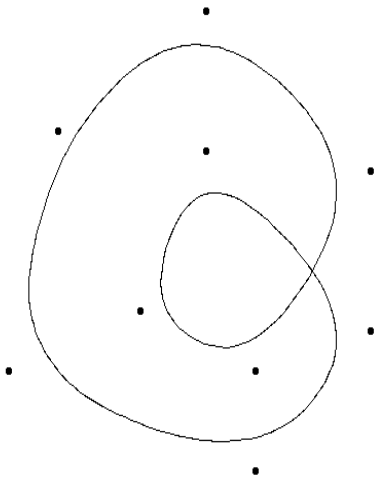
$$p(u,v) = \frac{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{i,j}B_{i,p}(u)B_{j,q}(v)\mathbf{P}_{i,j}}{\sum_{i=0}^{m}\sum_{j=0}^{n} w_{i,j}B_{i,p}(u)B_{j,q}(v)} \text{ : NURBS}$$

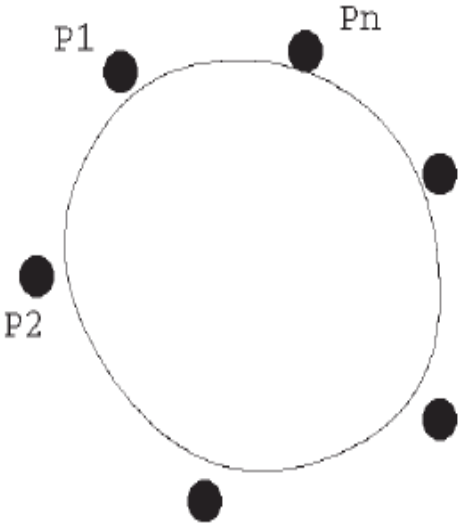


Clamping, Closing B-spline Surfaces

- The B-spline surfaces can be clamped by repeating the same knot values in one direction of the parameters (u or v)
- We can also close the curve / surface by recycling the control points

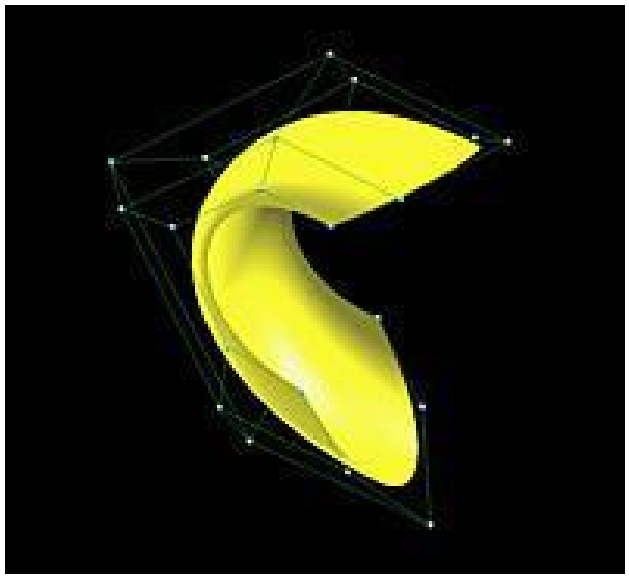


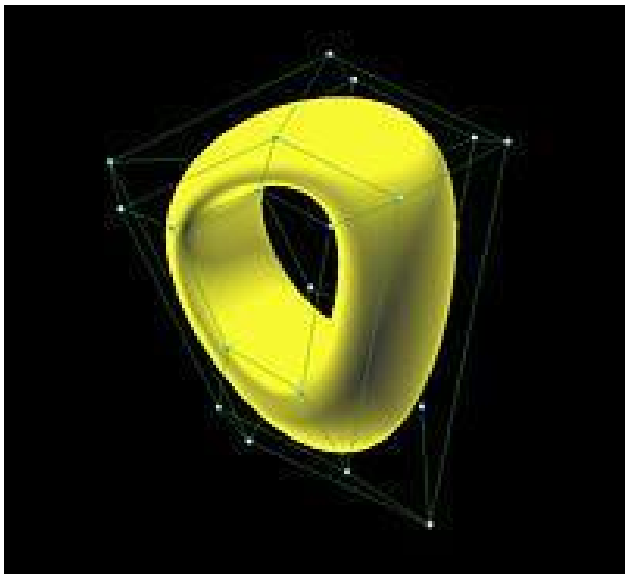


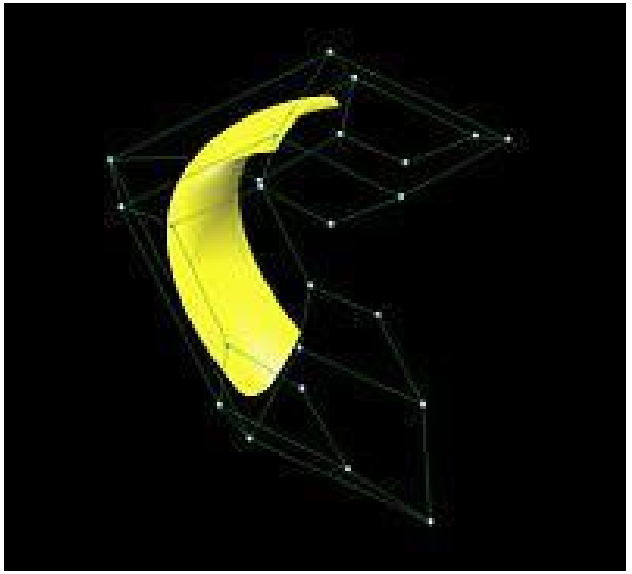


Closed B-spline Surfaces

- If a B-spline surface is closed in one direction, then the surface becomes a tube.
- Closed in two direction : torus
  - Problems handling objects of arbitrary topology, such as a ball, double torus







Today

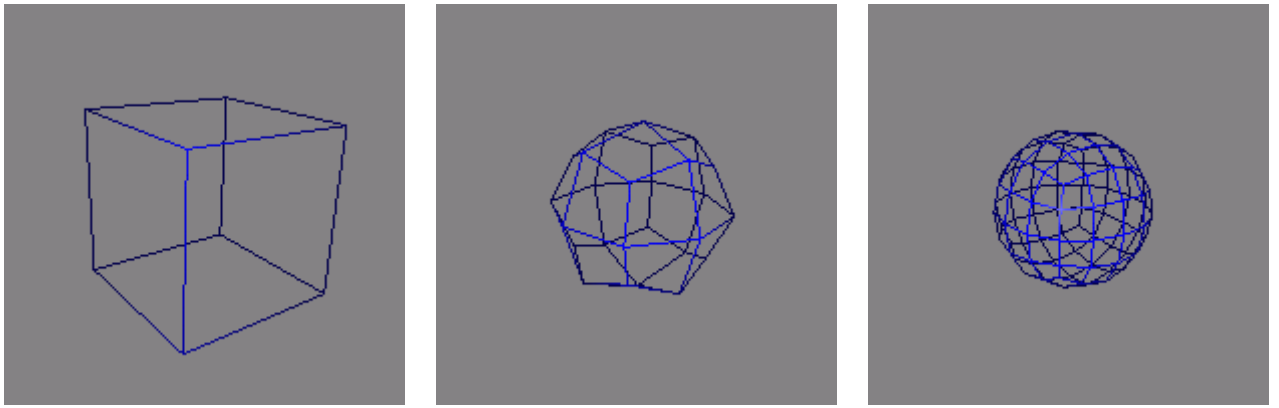
- More about Bezier and Bsplines
  - de Casteljau's algorithm
  - BSpline : General form
  - de Boor's algorithm
  - Knot insertion
- NURBS
- **Subdivision Surface**

Subdivision Surface

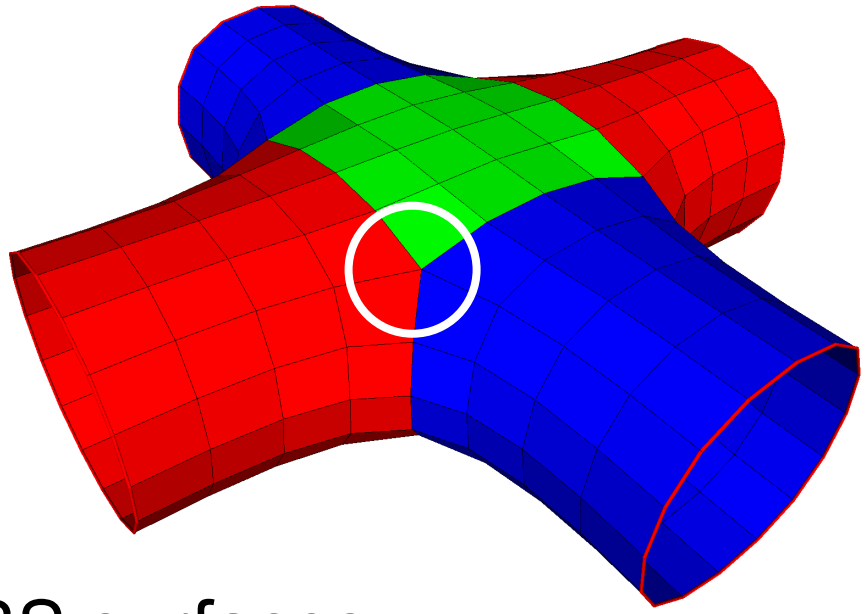
- A method to model smooth surfaces

3D subdivision surface

- Giving a rough shape first and subdivide it recursively
- Stop when the shape is smooth enough
- Used for modeling smooth surfaces



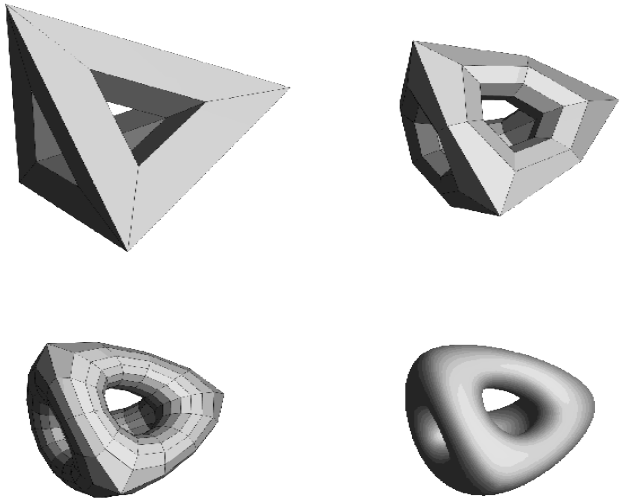
Motivation



- Shape modelling
  - Topological restrictions of NURBS surfaces
    - Plane, Cylinder, and Torus
    - It is difficult to maintain smoothness at seams of patchwork.
      - Example: hiding seams in Woody (*Toy Story*) [DeRose98]
  - NURBS also require the control nets consist of a regular rectangular grid of control points
- LOD in a scene
  - A coarse shape when far away, a smooth dense surface when closer to the camera

Subdivision surface

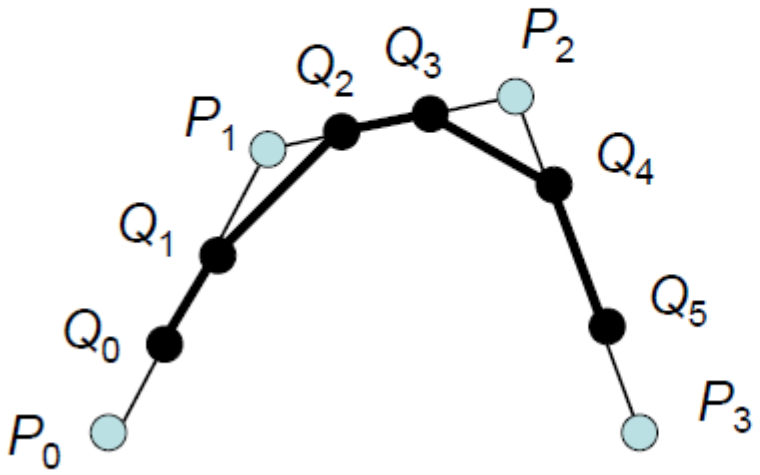
- Can handle arbitrary topology



Different Schemes

- Doo-Sabin ‘78
- Catmull-Clark ‘78
- Etc (Loop, Butterfly, and many others)

A Primer: Chaiken's Algorithm



$$Q_0 = \frac{3}{4}P_0 + \frac{1}{4}P_1$$

$$Q_1 = \frac{1}{4}P_0 + \frac{3}{4}P_1$$

$$Q_2 = \frac{3}{4}P_1 + \frac{1}{4}P_2$$

$$Q_3 = \frac{1}{4}P_1 + \frac{3}{4}P_2$$

Apply Iterated Function System

$$Q_i = \frac{3}{4}P_i + \frac{1}{4}P_{i+1}$$

$$Q_{i+1} = \frac{1}{4}P_i + \frac{3}{4}P_{i+1}$$

$$Q_4 = \frac{3}{4}P_2 + \frac{1}{4}P_3$$

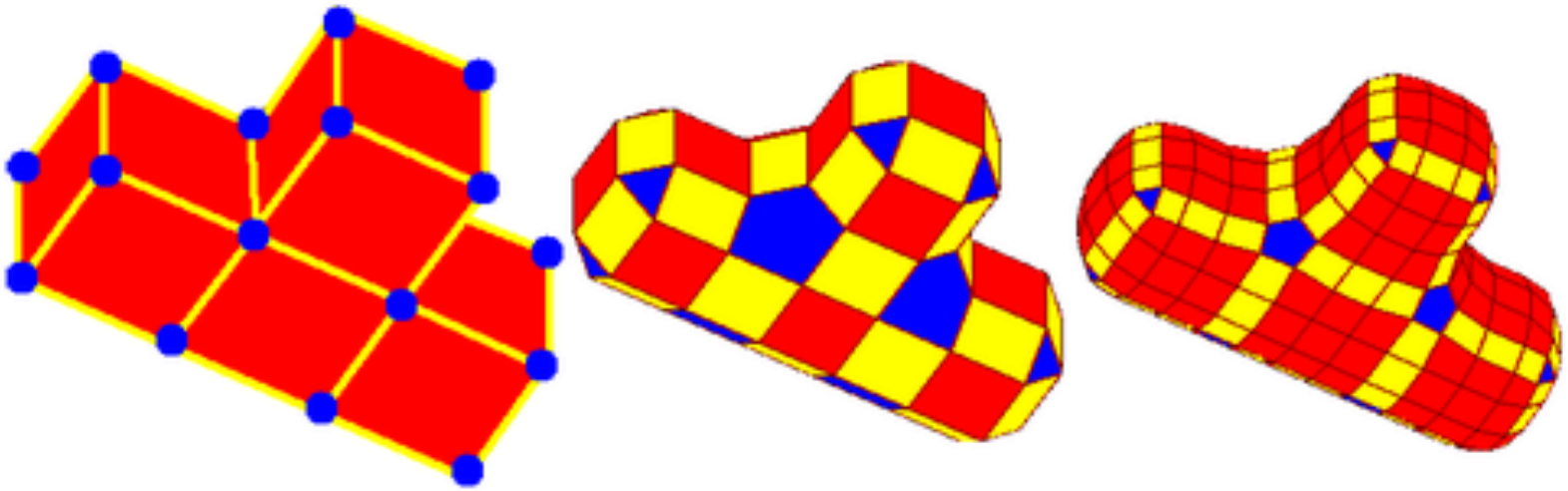
$$Q_5 = \frac{1}{4}P_2 + \frac{3}{4}P_3$$

Limit Curve Surface

http://www.multires.caltech.edu/teaching/demos/java/chaikin.htm

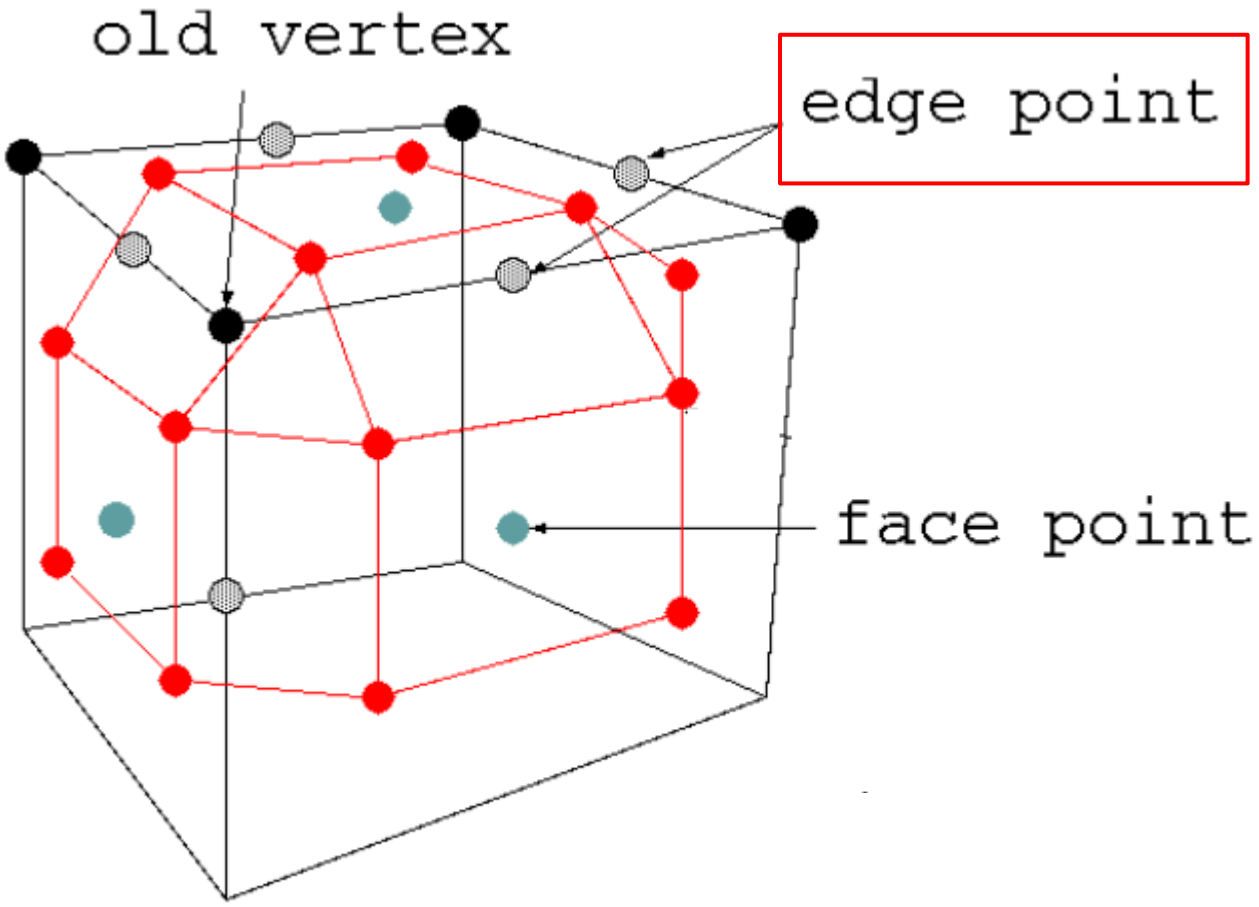
Doo-Sabin Subdivision

- Proposed by Doo and Sabin in 1978
- An extension of Chaiken's algorithm to 3D mesh surfaces



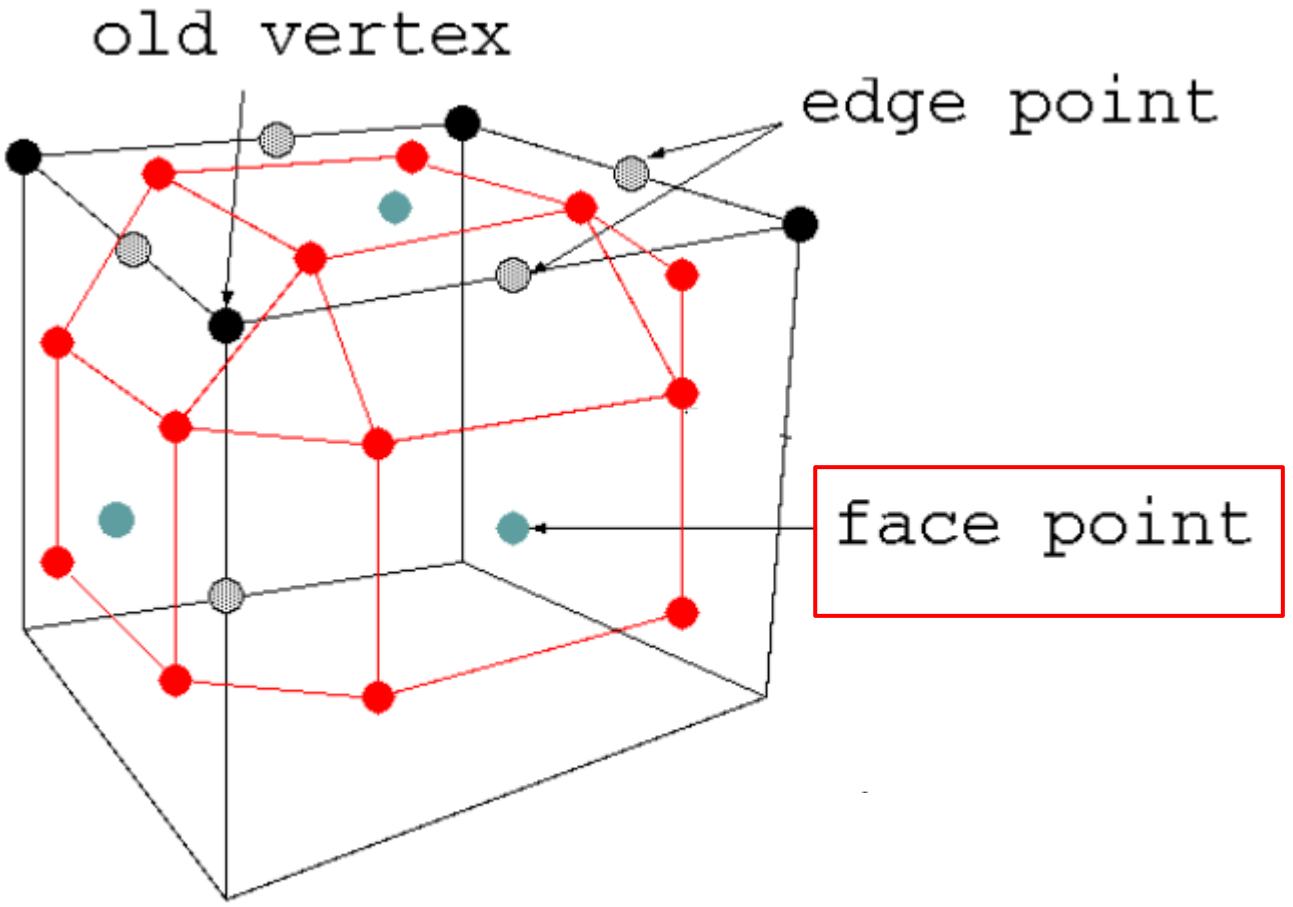
Doo-Sabin Subdivision (2)

- **An *edge point*** is formed from the midpoint of each edge
- A *face point* is formed as the centroid of each polygon of the mesh.



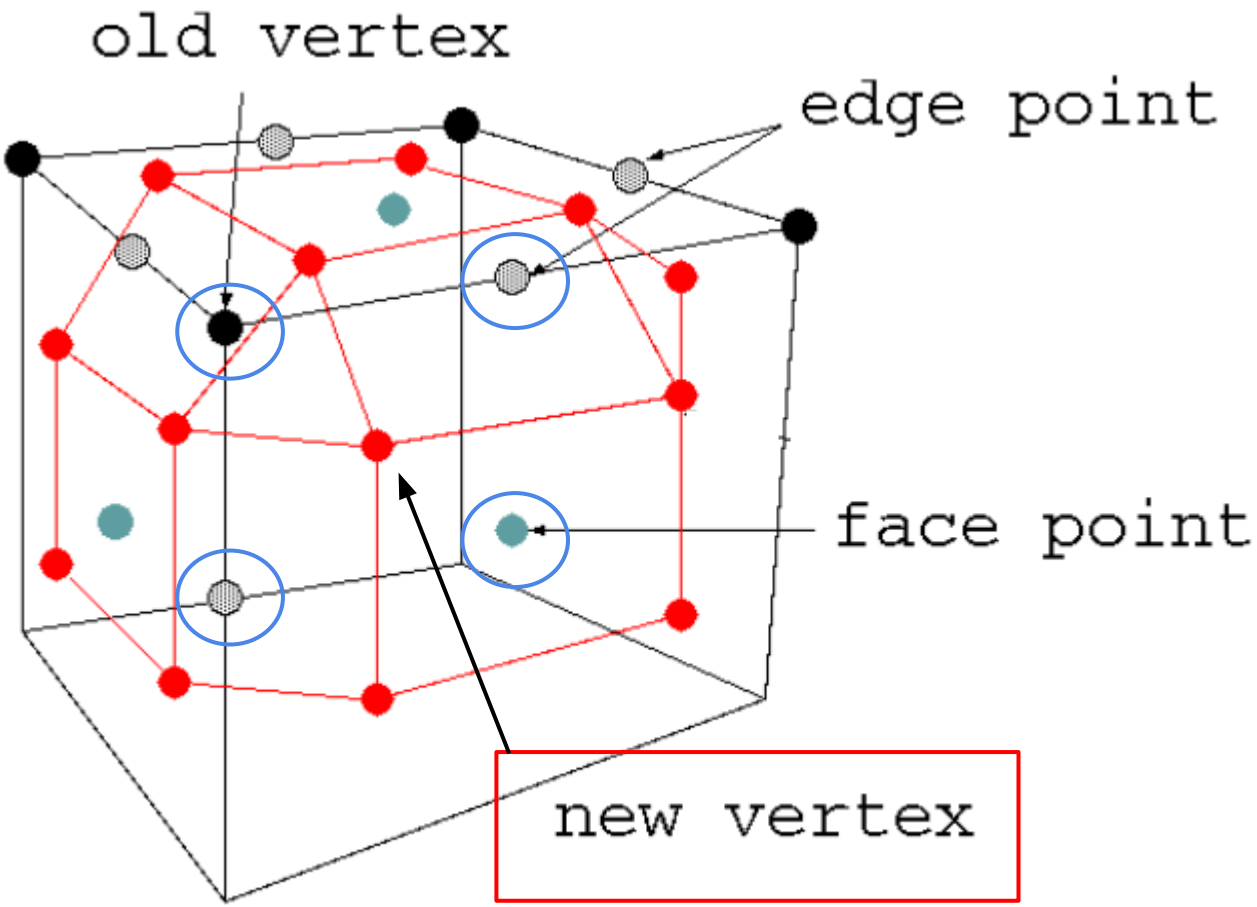
Doo-Sabin Subdivision (2)

- An *edge point* is formed from the midpoint of each edge
- **A *face point* is formed as the centroid of each polygon of the mesh.**

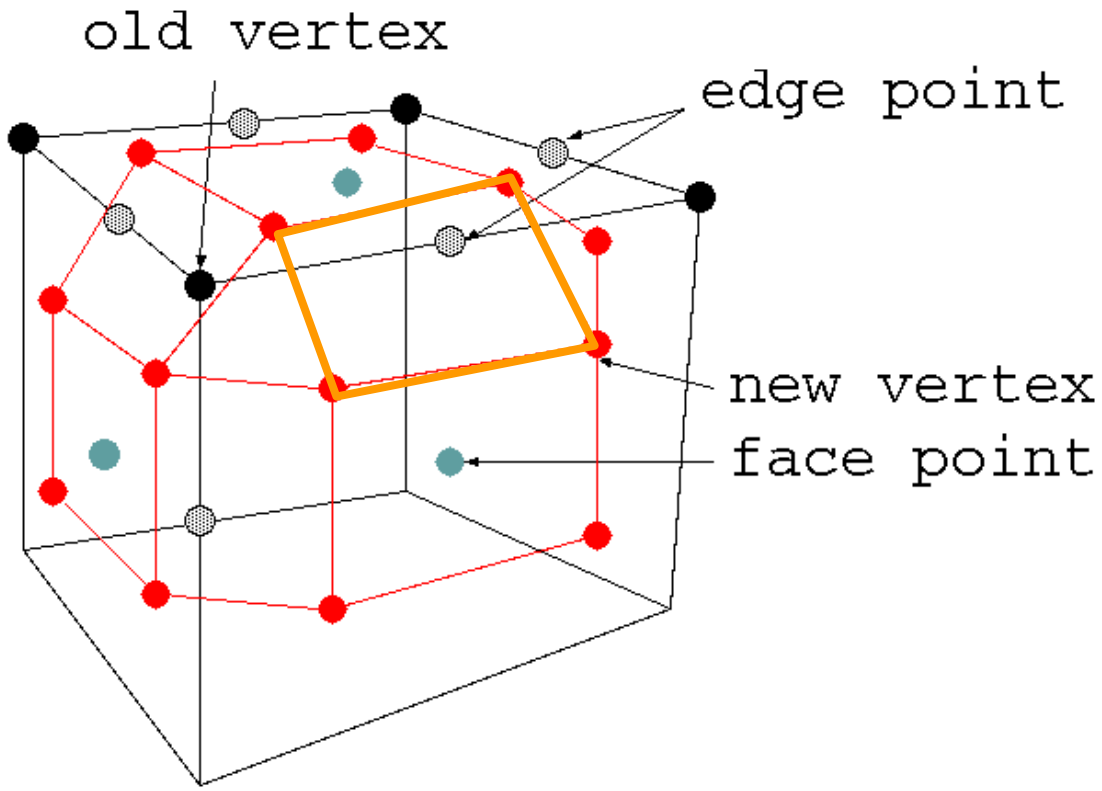


Doo-Sabin Subdivision (2)

- Finally, each vertex in new mesh is formed as average of
  - a vertex in the old mesh,
  - a face point for a polygon that touches that old vertex, and
  - the edge points for the two edges that belong to that polygon and touch that old vertex.



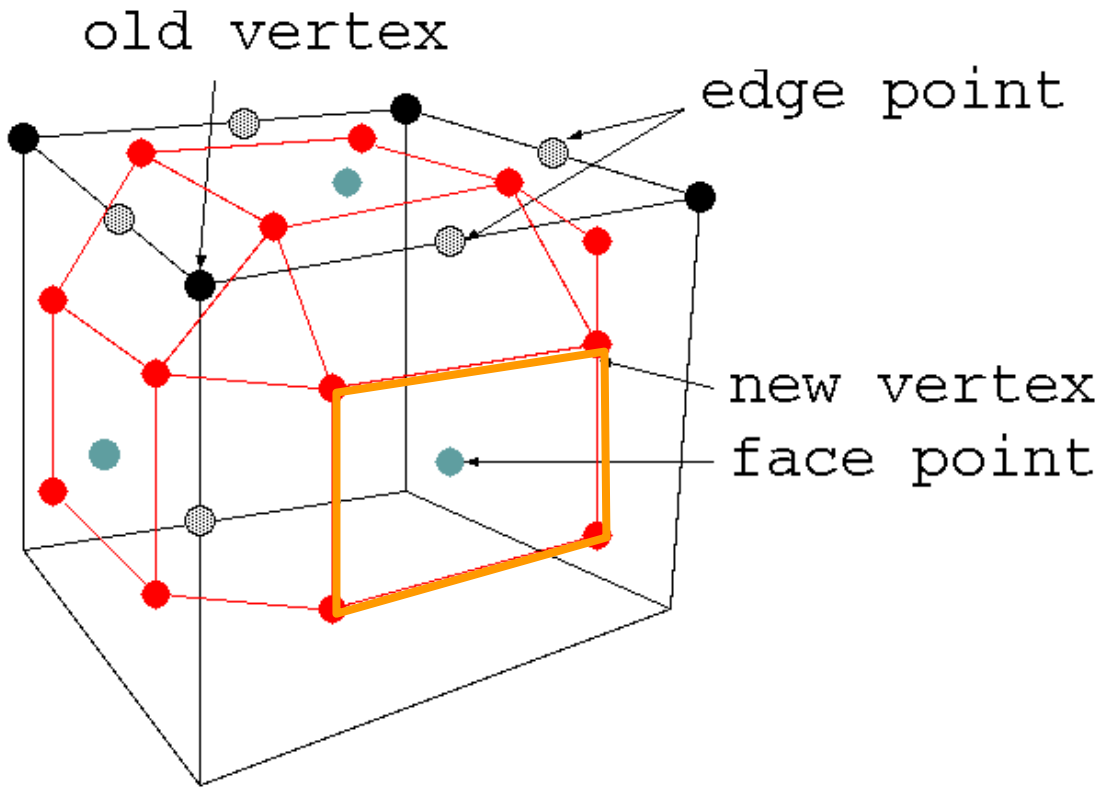
Doo-Sabin Subdivision (3)



The new mesh will be composed of

- **quadrilaterals for each edge in the old mesh,**
- a smaller n-sided polygon for each n-sided polygon in the old mesh, and
- an n-sided polygon for each n-valence vertex (Valence being the number of edges that touch the vertex).

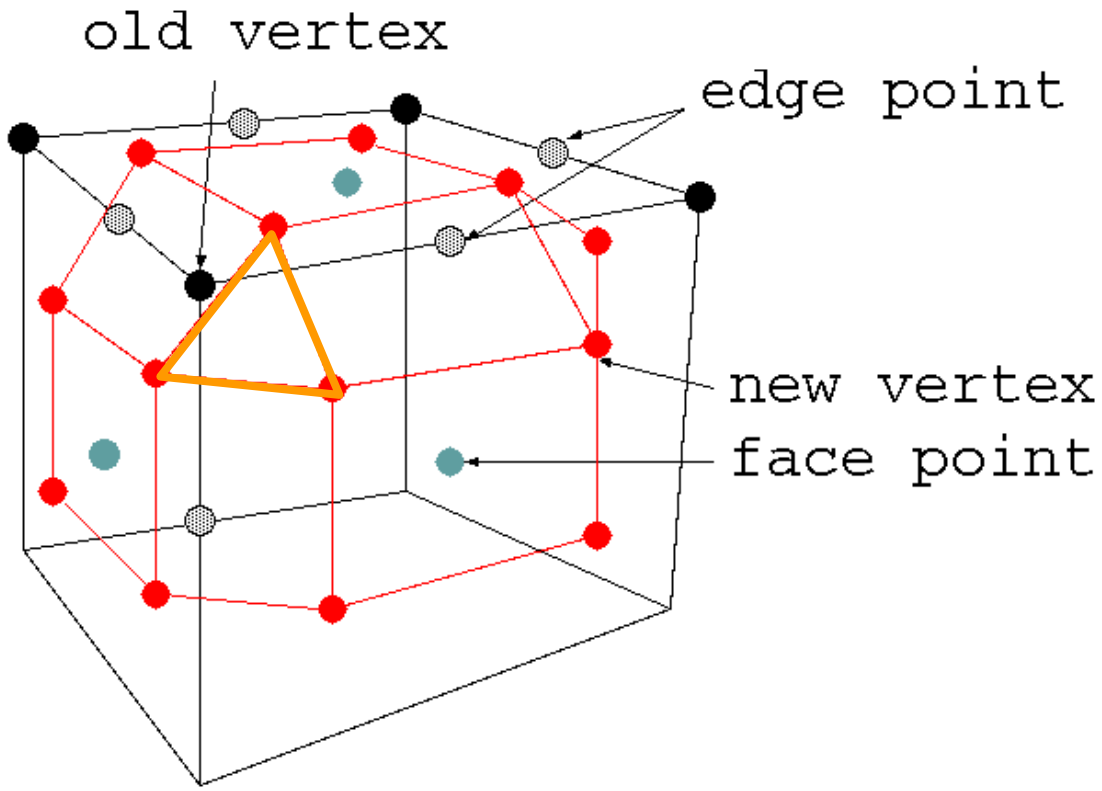
Doo-Sabin Subdivision (3)



The new mesh will be composed of

- quadrilaterals for each edge in the old mesh,
- **a smaller n-sided polygon for each n-sided polygon in the old mesh, and**
- an n-sided polygon for each n-valence vertex (Valence being the number of edges that touch the vertex).

Doo-Sabin Subdivision (3)

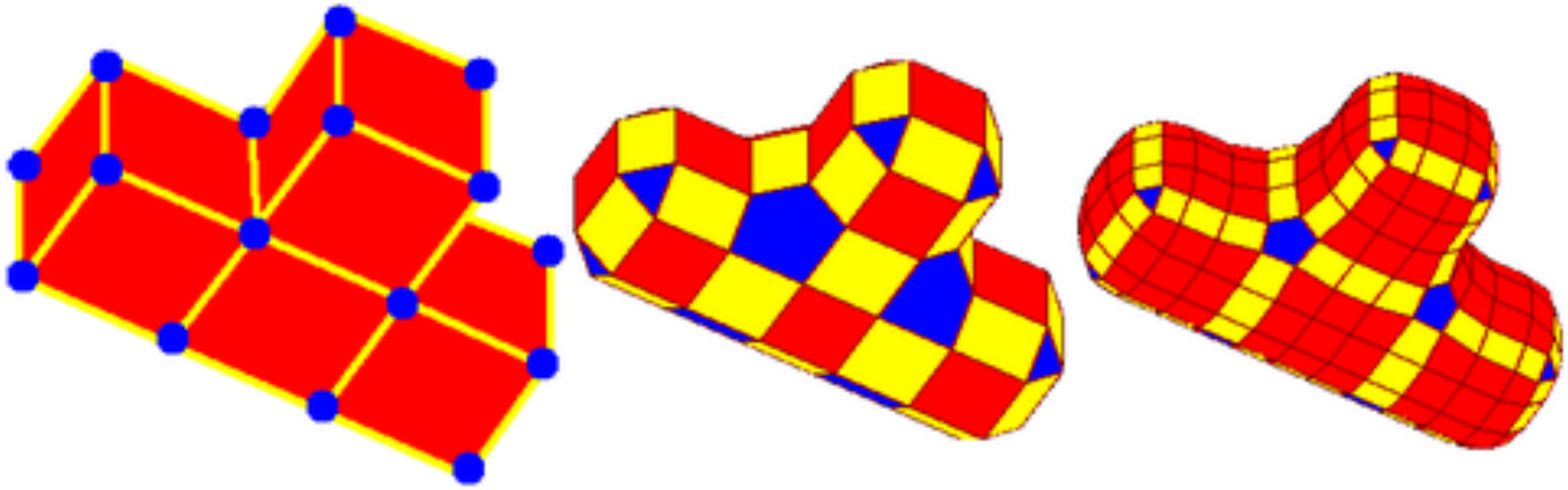


The new mesh will be composed of

- quadrilaterals for each edge in the old mesh,
- a smaller n-sided polygon for each n-sided polygon in the old mesh, and
- **an n-sided polygon for each n-valence vertex (Valence being the number of edges that touch the vertex).**

Doo-Sabin Subdivision

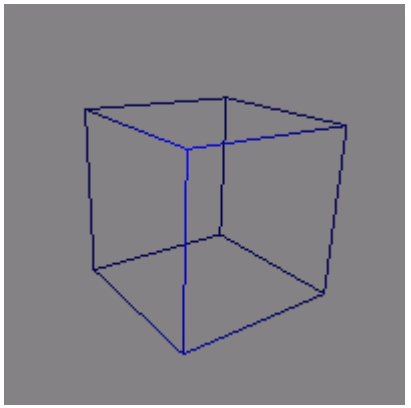
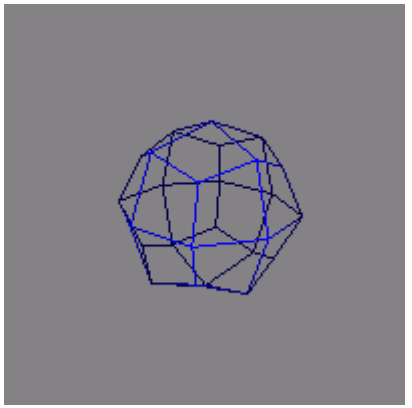
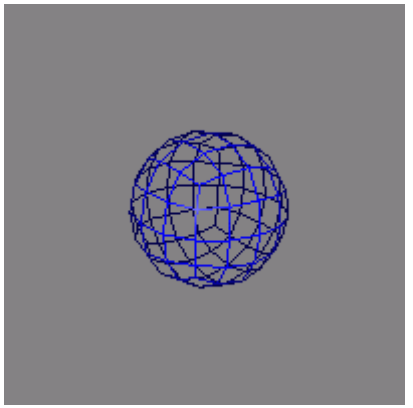
- Proposed by Doo and Sabin in 1978
- An extension of Chaiken's algorithm to 3D mesh surfaces



http://www.rose-hulman.edu/~finn/CCLI/Applets/DooSabinApplet.html

Catmull-Clark Subdivision

- A face with n edges are subdivided into n quadrilaterals
- Quads are better than triangles at capturing the symmetries of natural and man-made objects.

Catmull-Clark Subdivision

- FACE

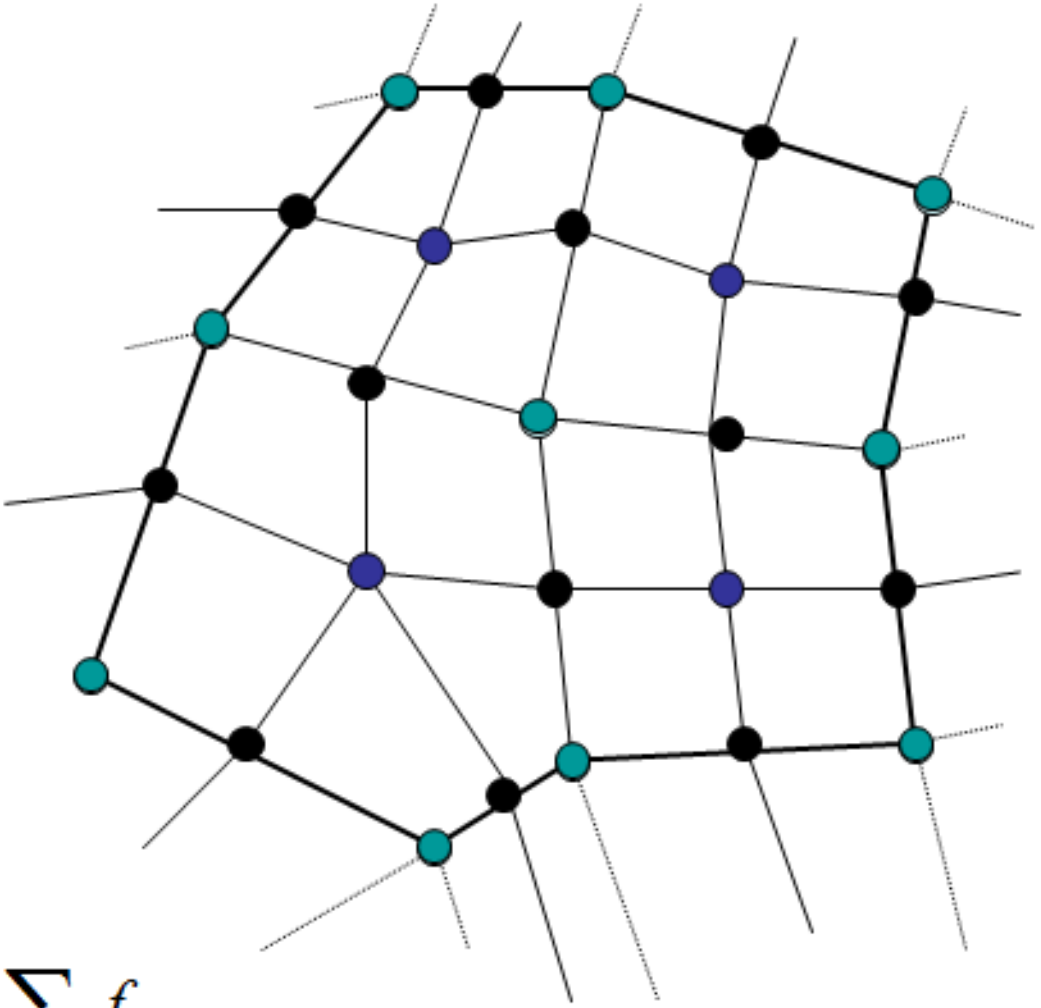
$$f = \frac{1}{n}\sum_{1}^{n} v_i$$

- EDGE

$$e = \frac{v_1 + v_2 + f_1 + f_2}{4}$$

- VERTEX

$$v_{i+1} = \frac{n-2}{n} v_i + \frac{1}{n^2}\sum_j e_j + \frac{1}{n^2}\sum_j f_j$$



http://www.rose-hulman.edu/~finn/CCLI/Applets/CatmullClarkApplet.html

Modeling with Catmull-Clark

- Subdivision produces smooth continuous surfaces.
- How can “sharpness” and creases be controlled in a modeling environment?

ANSWER: Define new subdivision rules for “creased” edges and vertices.

- Tag Edges sharp edges.
- If an edge is sharp, apply new sharp subdivision rules.
- Otherwise subdivide with normal rules.

Sharp Edges…

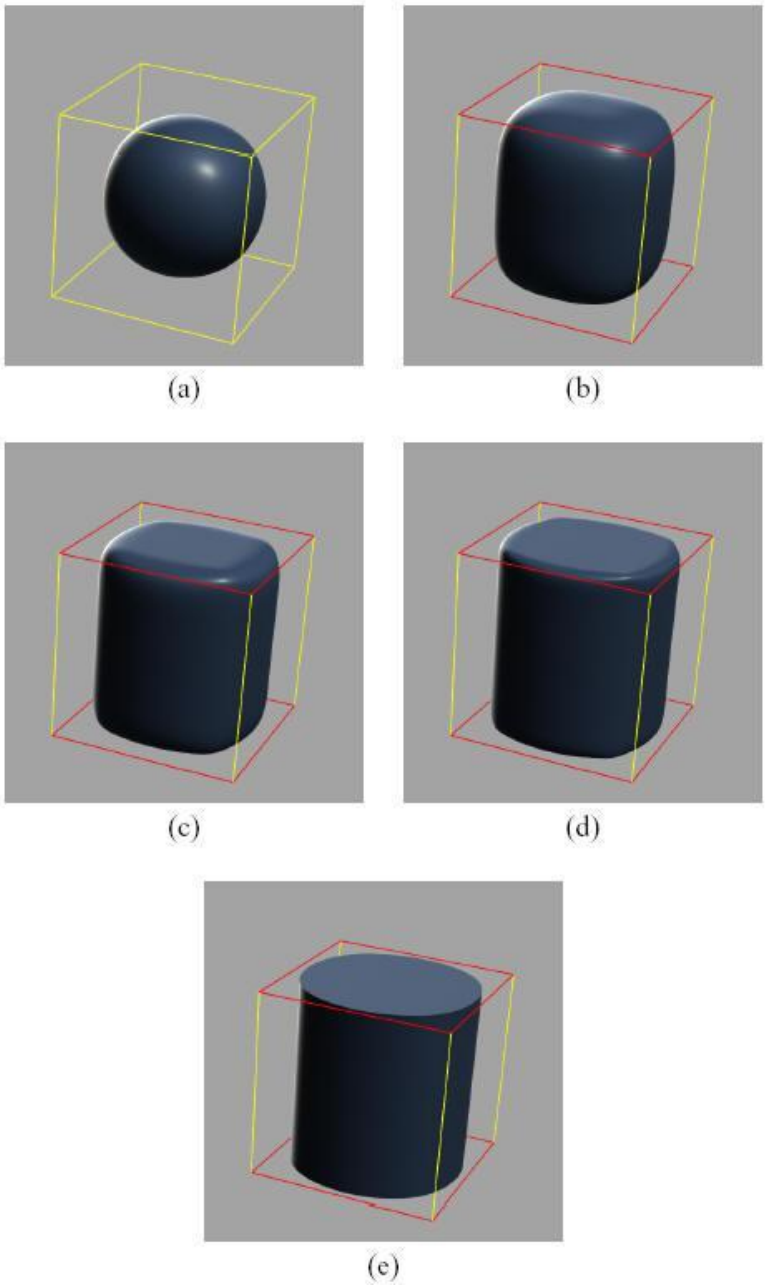
- Tag Edges as "**sharp**" or "not-sharp"
  - n = 0 – "**not sharp**"
  - n > 0 – sharp

During Subdivision,

- if an edge is "sharp", use sharp subdivision rules. Newly created edges, are assigned a sharpness of n-1.
- If an edge is "not-sharp", use normal smooth subdivision rules.

IDEA: Edges with a sharpness of "n" do not get subdivided smoothly for "n" iterations of the algorithm.

- In the picture on the right, the control mesh is a unit cube
- Different sharpness applied



Sharp Rules

- FACE (unchanged)

$$f = \frac{1}{n}\sum_{1}^{n} v_i$$

- EDGE

$$e = \frac{v_1 + v_2}{2}$$

- VERTEX

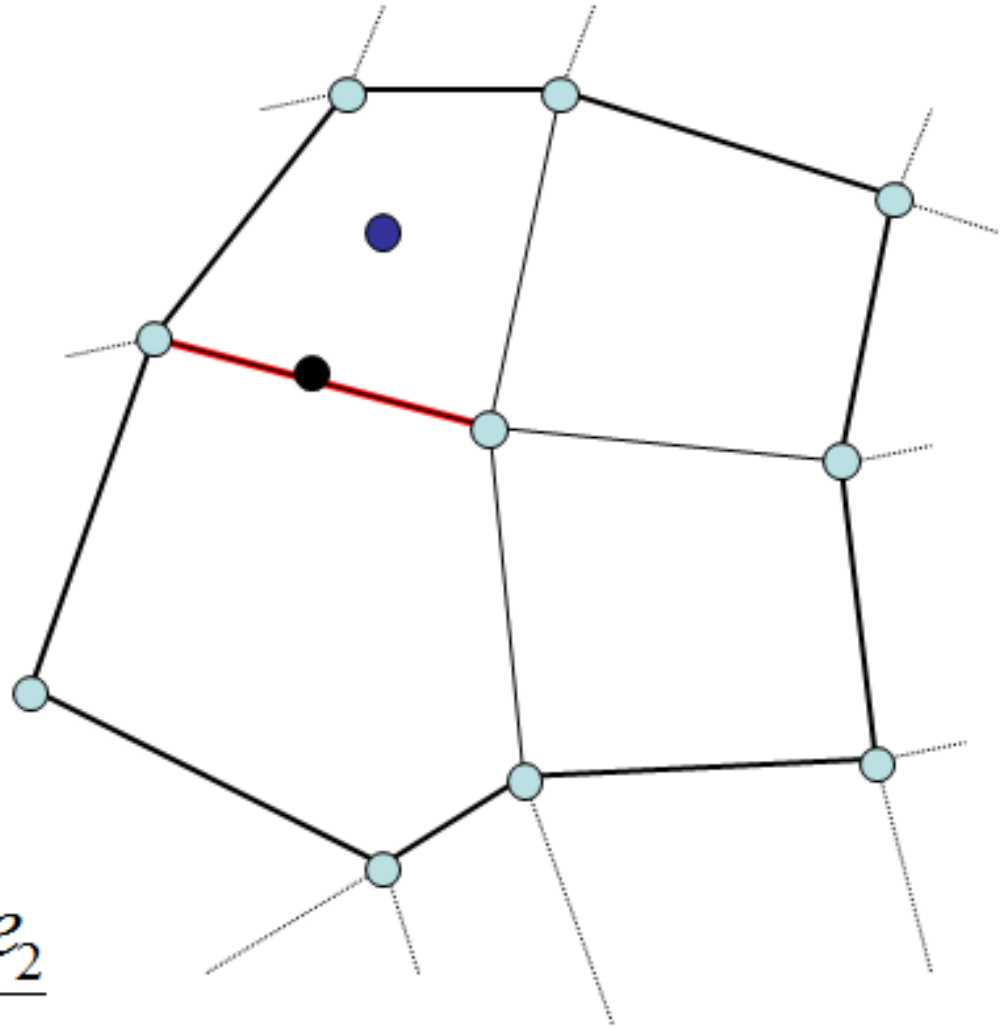
adj. Sharp edges

corner >2

$$v_{i+1} = v_i$$

crease 2

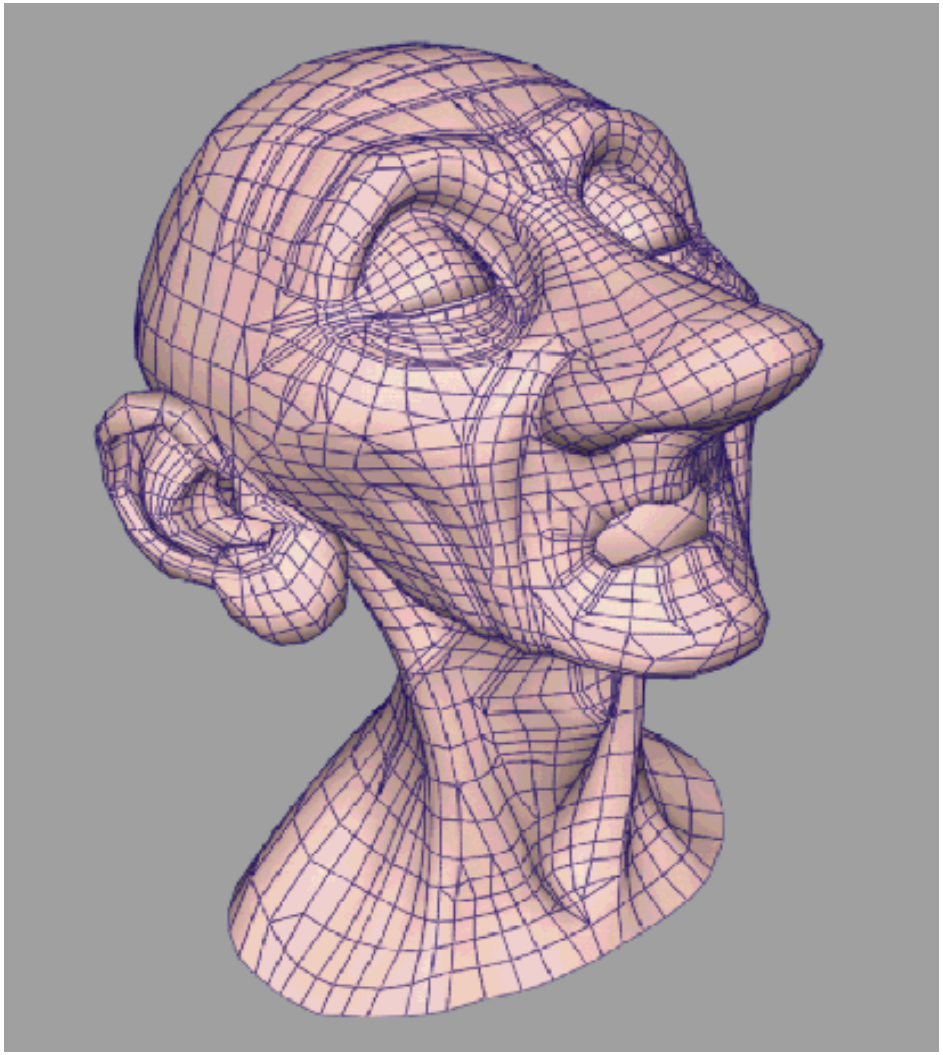
$$v_{i+1} = \frac{e_1 + 6v_i + e_2}{8}$$



Another example of creases

Subdivision Surfaces in character animation [DeRose98]

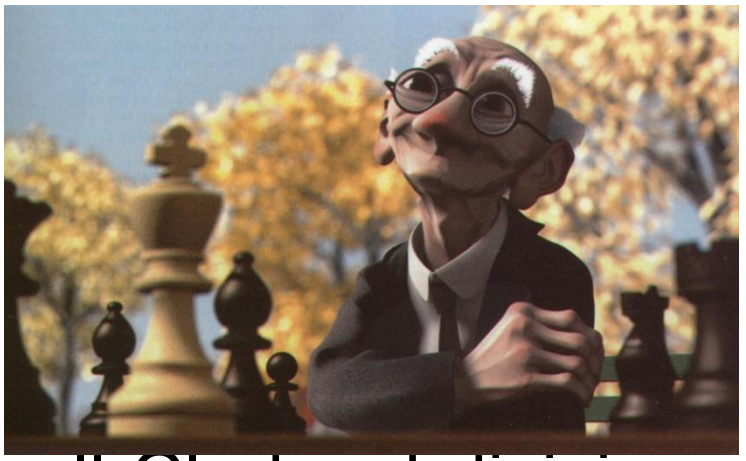
- Used for first time in Geri's game to overcome topological restriction of NURBS
- Modeled Geri's head, hands, jacket, pants, shirt, tie, and shoes



Geri's Game

- Academy Award winning movie by Pixar
  - Made by subdivision surface

[http://www.youtube.com/watch?v=1m7dcbIKvlw]

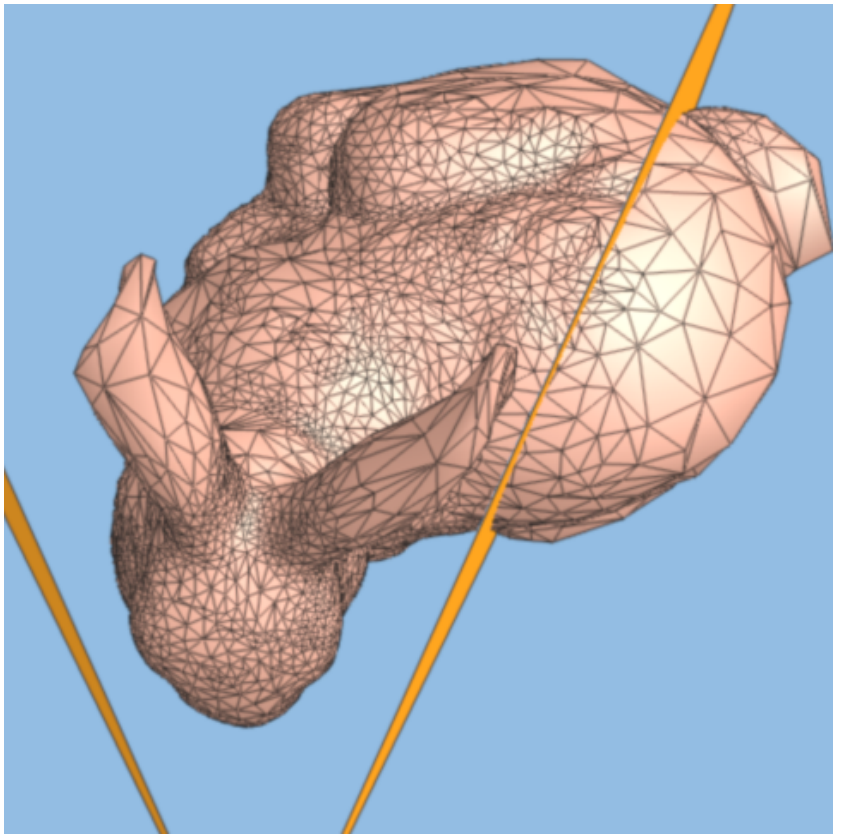


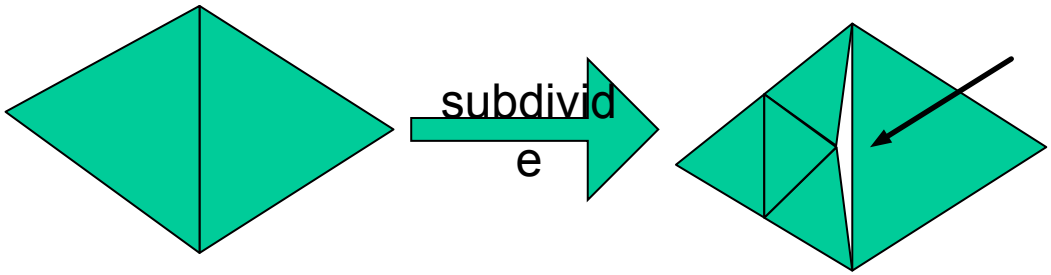
Demo of Catmull-Clark subdivision surface

- http://www.youtube.com/watch?v=lU8f0hnorU8&feature=related

Adaptive Subdivision

- Not all regions of a model need to be subdivided.
- Idea: Use some criteria and adaptively subdivide mesh where needed.
  - Curvature
  - Screen size (make triangles < size of pixel)
  - View dependence
    - Distance from viewer
    - Silhouettes
    - In view frustum
  - Careful! Must ensure that “cracks” aren’t made





Subdivision Surface Summary

- Advantages
  - Simple method for describing complex surfaces
  - Relatively easy to implement
  - Arbitrary topology
  - Local support
  - Guaranteed continuity
  - Multi-resolution
- Difficulties
  - Intuitive specification
  - Parameterization
  - Intersections

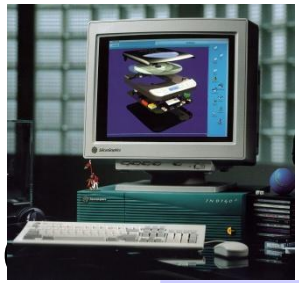
- Edwin Catmull

  Utah – NYIT – Lucas Films – Pixar -

  present: President of Disney Animation Studios and Pixar Animation Studios

- Jim Clark

  Utah – UCSC - Stanford – Silicon Graphics - Netscape - …



He also co-produced the m[illegible] Cove.

Readings

- A very good website for parametric curves / surfaces http://www.cs.mtu.edu/~shene/COURSES/cs3621/
- DeRose, Tony, Michael Kass, and Tien Truong. Subdivision Surfaces in Character Animation. *SIGGRAPH 98*.
- Clark, E., and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Geometric Design*, Vol. 10, No. 6, 1978.
- Doo, D. and M. Sabin. Behavior of Recursive Division Surfaces Near Extraordinary Points. Computer-Aided Design. Vol. 10, No. 6, 1978.