Introduction to Computer Graphics

10. Curves and Surfaces

I-Chen Lin

National Chiao Tung University

Textbook: E.Angel, D. Shreiner Interactive Computer Graphics, 6th Ed., Pearson
Ref: D.D. Hearn, M. P. Baker, W. Carithers, Computer Graphics with OpenGL, 4th Ed., Pearson
Ref: Prof. S.Chenney, Computer Graphics course note, Univ. Wisconsin

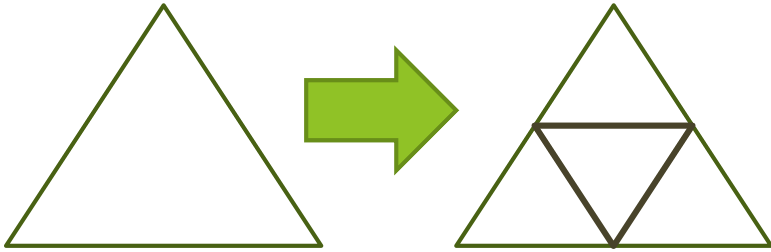
Limitations of Polygons

- Inherently an approximation
  - Planar facets and silhouettes.
  - Otherwise, it needs a very large numbers of polygons.
- Fixed resolution
- No natural parameterization
  - Deformation is relatively difficult
  - Hard to extract information like curvature or to keep smoothness

Figures from MIT EECS 6.837, Durand and Cutler

Subdivision

- Subdividing a polygon can alleviate the problem of polygonal mesh representation.



- E.g. Loop's subdivision
  - Split a triangle into four smaller ones.
  - Choose locations of new vertices by weighted average of the original neighbor vertices.

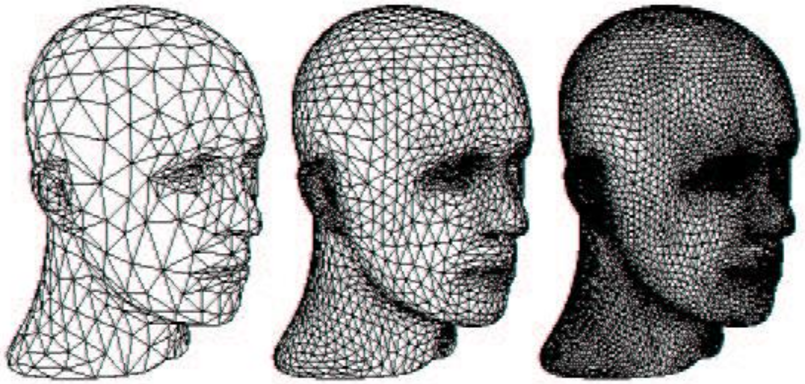
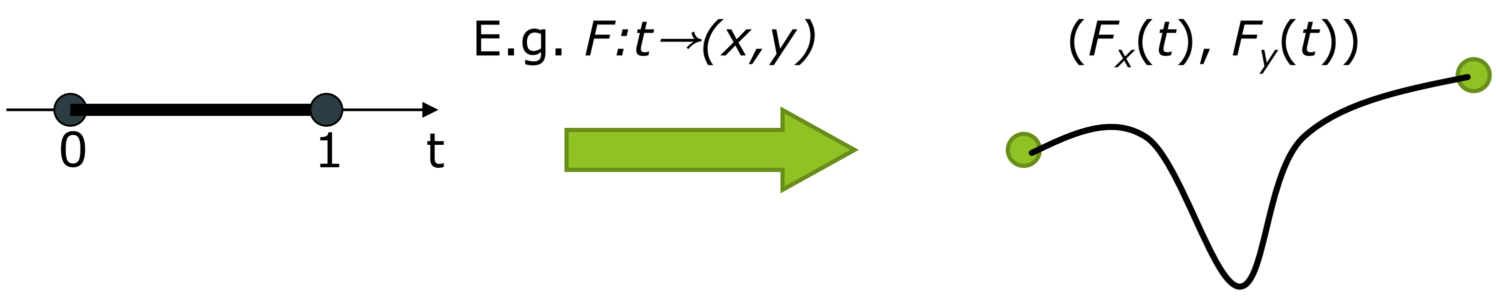


Figure from Zorin & Schroeder
SIGGRAPH 99 Course Notes

What are Parametric Curves?

- Define a mapping from parameter space to 3D points
  - A function that takes parameter values and gives back 3D points
  - 1D for curves; 2D for surfaces.
- The result is a parametric curve or surface



Why Parametric Curves?

- Intended to provide the generality of polygon meshes but with fewer parameters for smooth surfaces.

- Faster to create a curve, and easier to edit an existing curve.

- Easier to animate than polygon meshes.

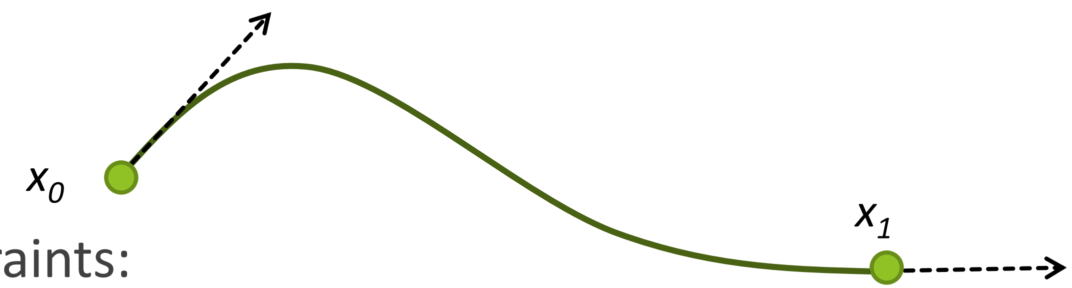
- Normal vectors and texture coordinates can be easily defined everywhere.

Polynomial functions as curves

- We can use polygonal functions to form curves.
- X value of cubic curves $fx(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3$
- The problem is how to efficiently find out the coefficient c_i.
- You may have learned least square curve fitting …

Hermite Curves

- A Hermite curve is a curve for which the user provides:
  - The endpoints of the curve
  - The parametric derivatives of the curve at the endpoints (tangents with length) (*dfx/dt*, *dfy/dt*), where *fx* and *fy* are functions of *t.*



- For *x*, we have constraints:
  - The curve must pass through x_0 when $t=0$
  - The derivative must be x'_0 when $t=0$
  - The curve must pass through x_1 when $t=1$
  - The derivative must be x'_1 when $t=1$

Hermite Curves

$$fx(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 \qquad \frac{dfx(t)}{dt} = c_1 + 2c_2 t + 3c_3 t^2$$

C_i are unknown, t = 0~1

The curve must pass through x_0 when $t=0$

- $fx(0) = c_0 = x_0$

The derivative must be x'_0 when $t=0$

- $fx'(0) = c_1 = x'_0$

The curve must pass through x_1 when $t=1$

- $fx(1) = c_0 + c_1 + c_2 + c_3 = x_1$

The derivative must be x'_1 when $t=1$

- $fx'(1) = c_1 + 2c_2 + 3c_3 = x'_1$

Hermite Curves

- Solving for the unknowns gives:

$$fx(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3$$

$$c_3 = -2x_1 + 2x_0 + x_1' + x_0'$$
$$c_2 = 3x_1 - 3x_0 - x_1' - 2x_0'$$
$$c_1 = x_0'$$
$$c_0 = x_0$$

- After rearranging, we get

$$\begin{aligned}\mathbf{x} = {} & \mathbf{x}_1(-2t^3 + 3t^2) \\ & + \mathbf{x}_0(2t^3 - 3t^2 + 1) \\ & + \mathbf{x}_1'(t^3 - t^2) \\ & + \mathbf{x}_0'(t^3 - 2t^2 + t)\end{aligned}$$

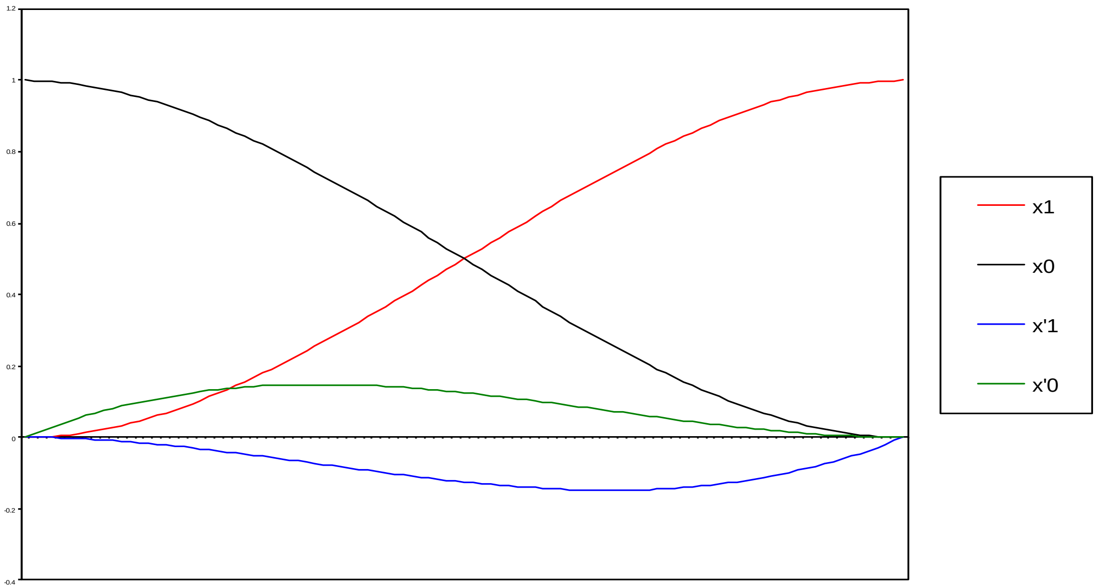
$$x = \begin{bmatrix} x_0 & x_1 & x_0' & x_1' \end{bmatrix} \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

- Extending to 3D

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_0 & x_1 & x_0' & x_1' \\ y_0 & y_1 & y_0' & y_1' \\ z_0 & z_1 & z_0' & z_1' \end{bmatrix} \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

The Blending Weights

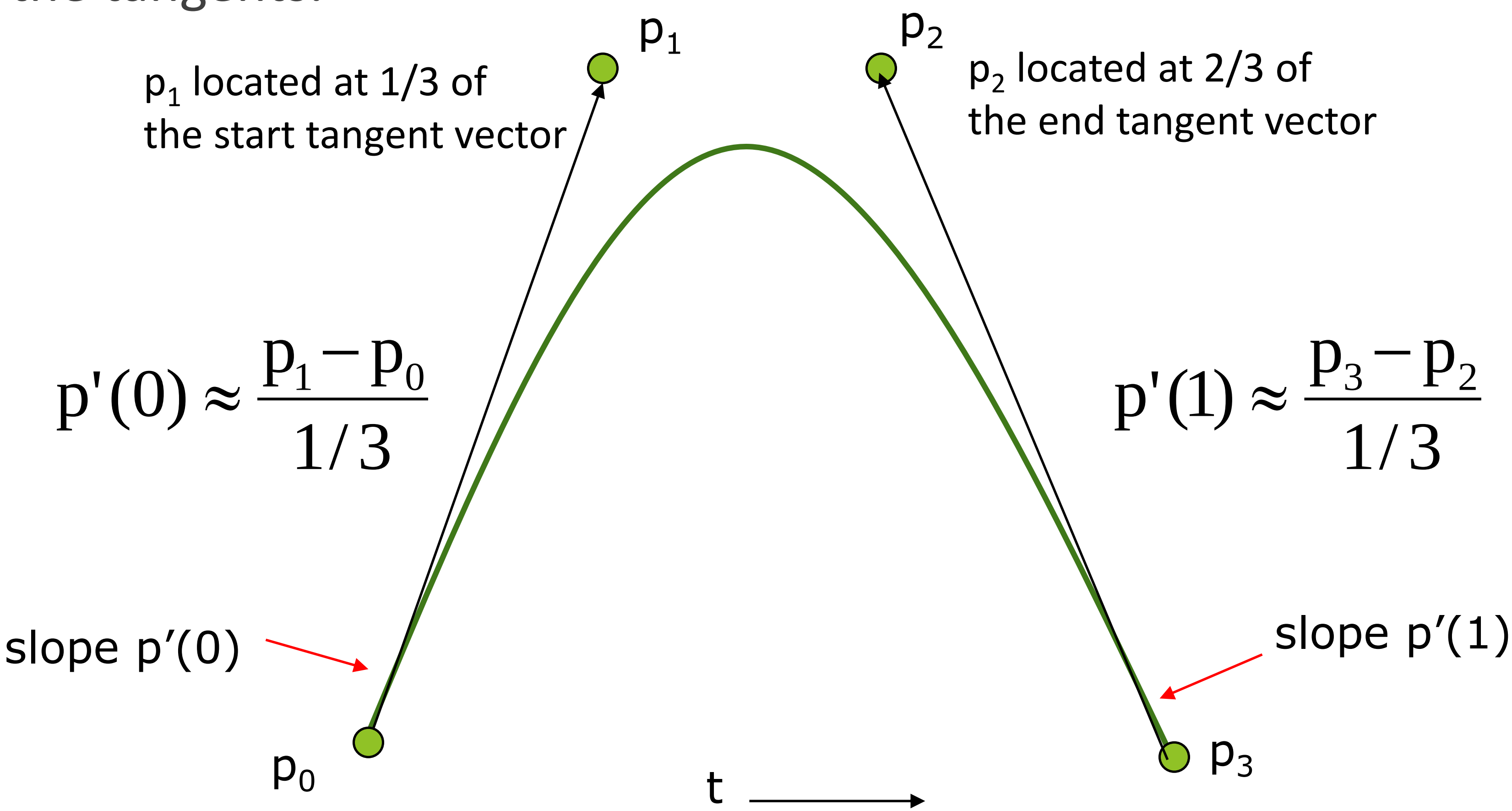
- A point on a Hermite curve is obtained by weighted blending each control point and tangent vector.



Weights of each component

Bezier Curves

- Two control points define endpoints, and two points control the tangents.



$$p'(0) \approx \frac{p_1 - p_0}{1/3}$$

$$p'(1) \approx \frac{p_3 - p_2}{1/3}$$

Bezier and Hermite curves

- The endsite conditions are the same.
  - $fx(0) = x_0 = c_0$
  - $fx(1) = x_3 = c_0+c_1+c_2+c_3$

$$fx(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3$$

$$\frac{dfx(t)}{dt} = c_1 + 2c_2 t + 3c_3 t^2$$

- Approximating derivative conditions
  - $fx'(0) = 3(x_1 - x_0) = c1$
  - $fx'(1) = 3(x_3 - x_2) = c_1+2*c_2+3*c_3$

$$c_3 = x_3 - 3x_2 + 3x_1 - x_0$$
$$c_2 = 3x_2 - 6x_1 + 3x_0$$
$$c_1 = 3x_1 - 3x_0$$
$$c_0 = x_0$$

- Or replacing the orignal Hermite matrix. (x_s, x_e, x'_s, x'_e)

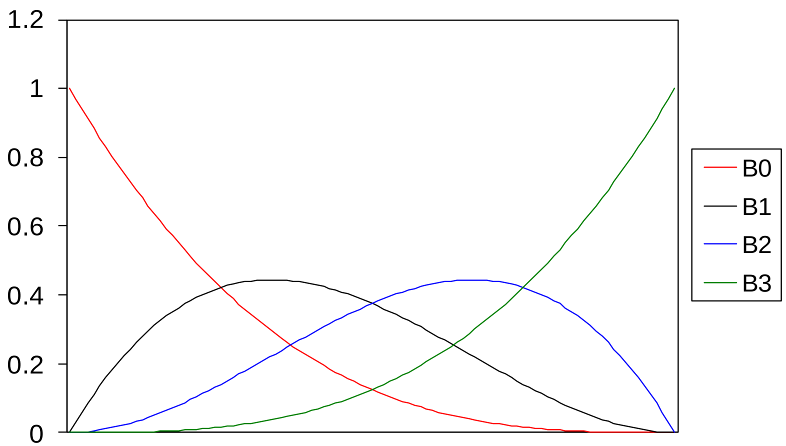
$$\begin{bmatrix} x_s \\ x_e \\ x'_s \\ x'_e \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$x(t) = \begin{bmatrix} x_s & x_e & x'_s & x'_e \end{bmatrix} \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

Bezier Curves

- A Bezier curve (x value) becomes

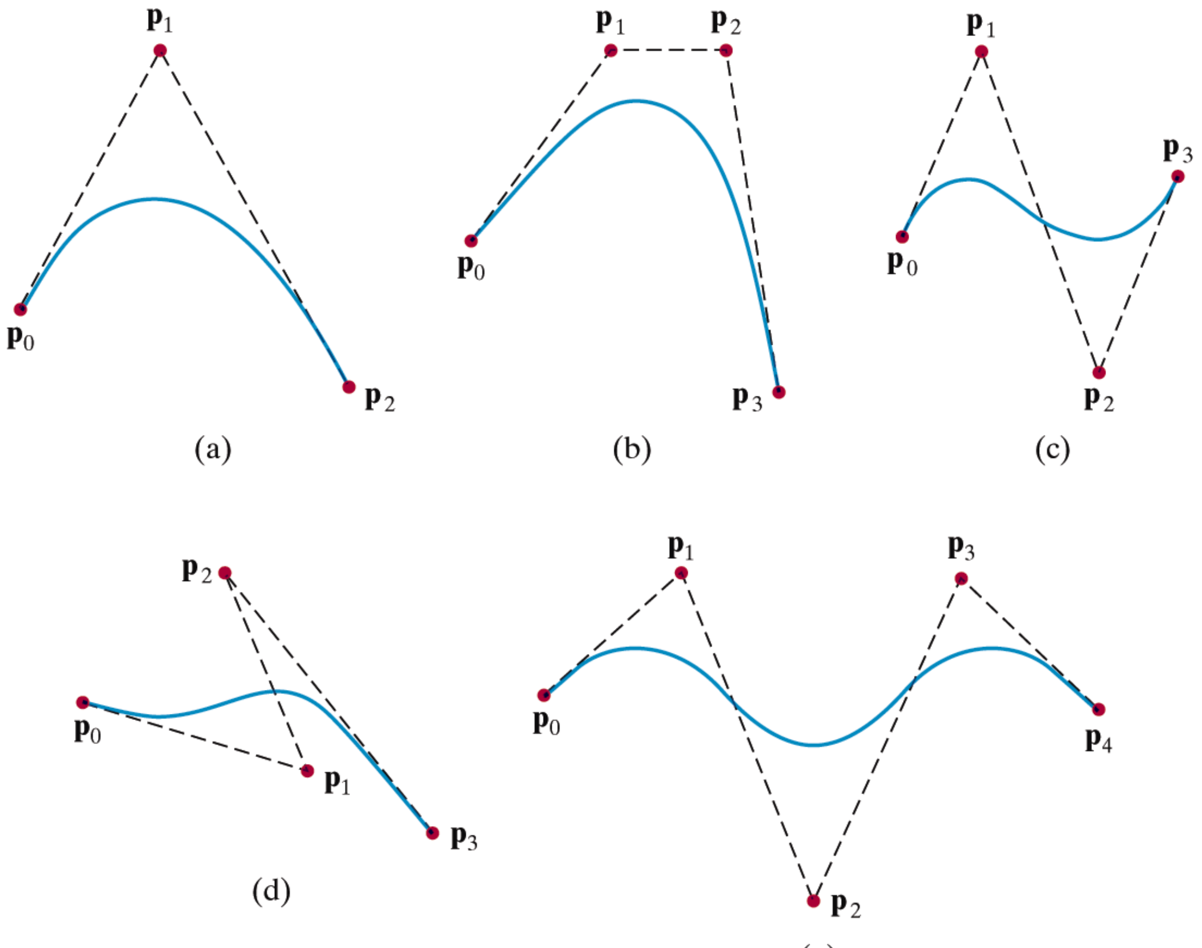
$$x(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$



The blending weights

$$x(t) = \begin{bmatrix} (1-t)^3 \\ 3t(1-t)^2 \\ 3t^2(1-t) \\ t^3 \end{bmatrix}^T \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Examples of Bezier curves



Bernstein Polynomials

- The blending functions of cubic bezier curves are a special case of the Bernstein polynomials (d:degree, k:index)

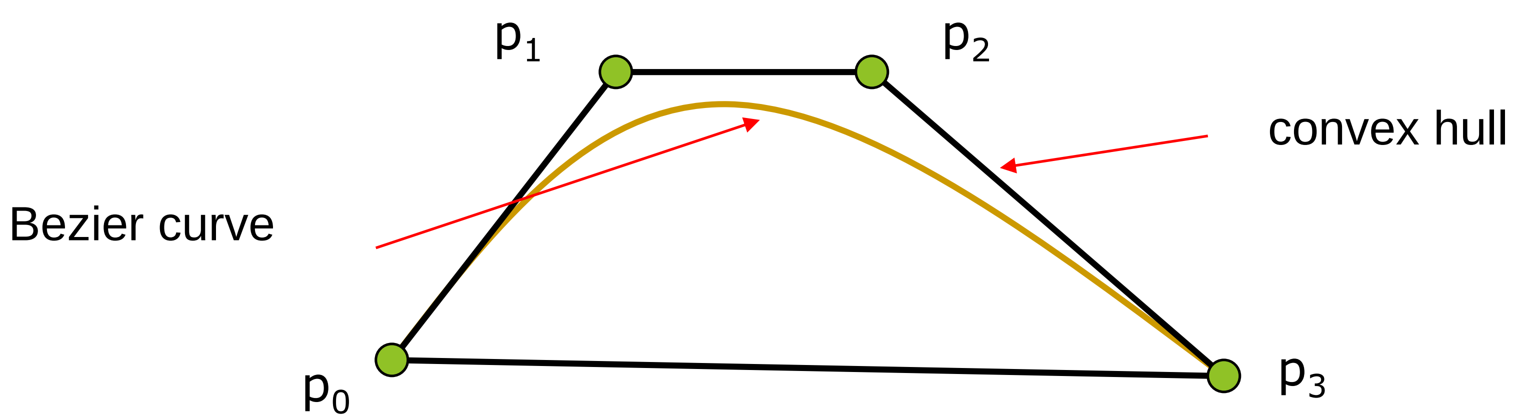
$$b_{\mathrm{kd}}(t) = \frac{d!}{k!(d-k)!} t^k (1-t)^{d-k}$$

$$B(t) = \sum_{k=0}^{d} b_{kd}(t) p_k \qquad t = 0 \sim 1$$

- These polynomials give the blending polynomials for any degree Bezier form
  - For any degree they all sum to 1
  - They are all between 0 and 1 inside (0,1)

Convex Hull Property

- The properties of the Bernstein polynomials ensure that all Bezier curves lie in the convex hull of their control points



Bezier Curve Subdivision

- Subdividing control polylines
  - produces two new control polylines for each half of the curve
  - defines the same curve
  - all control points are closer to the curve

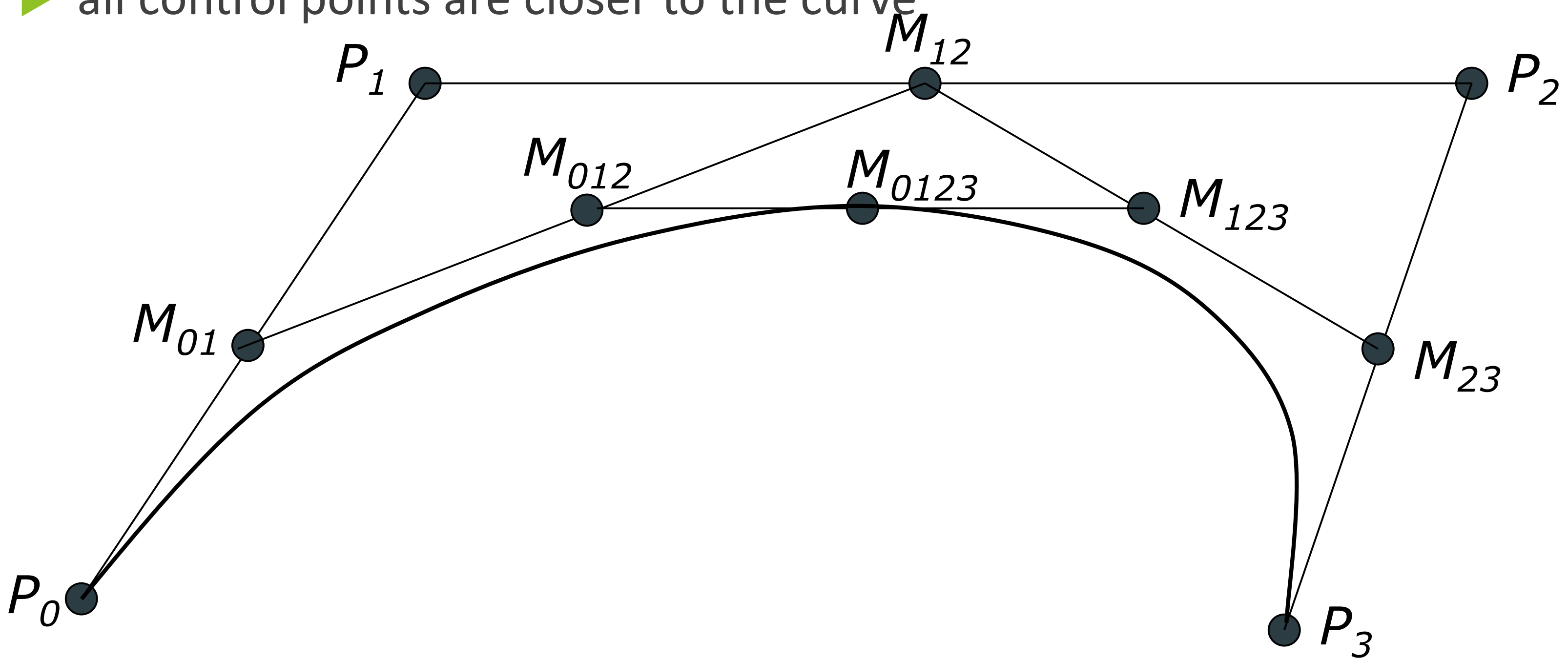


Figure from Prof. S.Chenney, Computer Graphics coursenote, Univ. Wisconsin

de Casteljau's Algorithm

- You can find the point on a Bezier curve for any parameter value *t* by subdivision
- If you want *t=0.25*, instead of taking midpoints take points 0.25 of the way

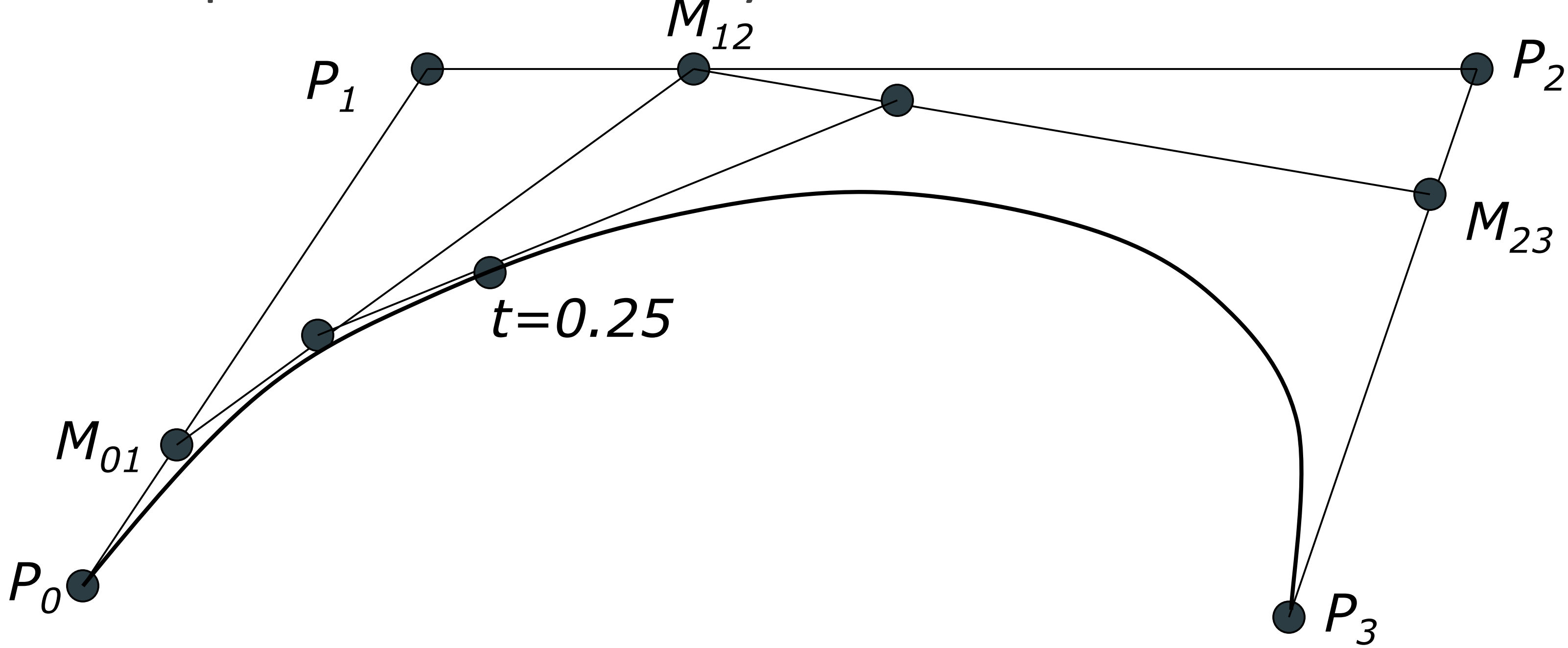
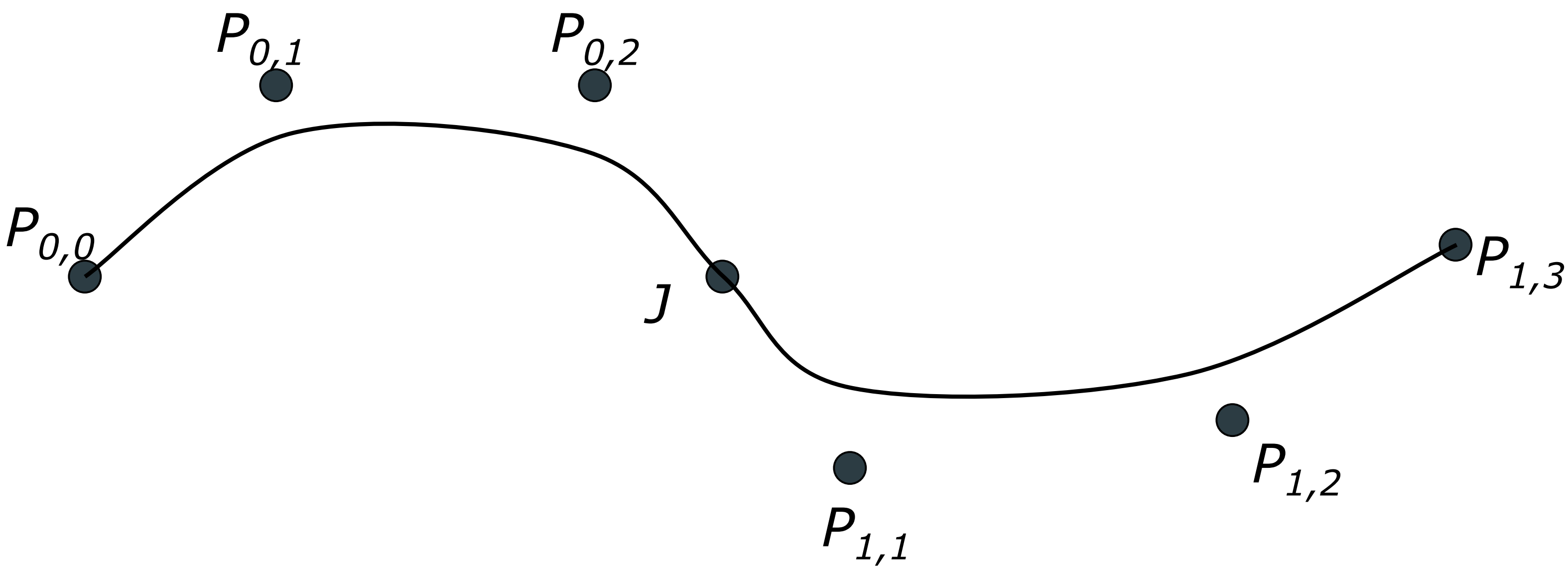


Figure from Prof. S.Chenney, Computer Graphics coursenote, Univ. Wisconsin

Bezier Continuity

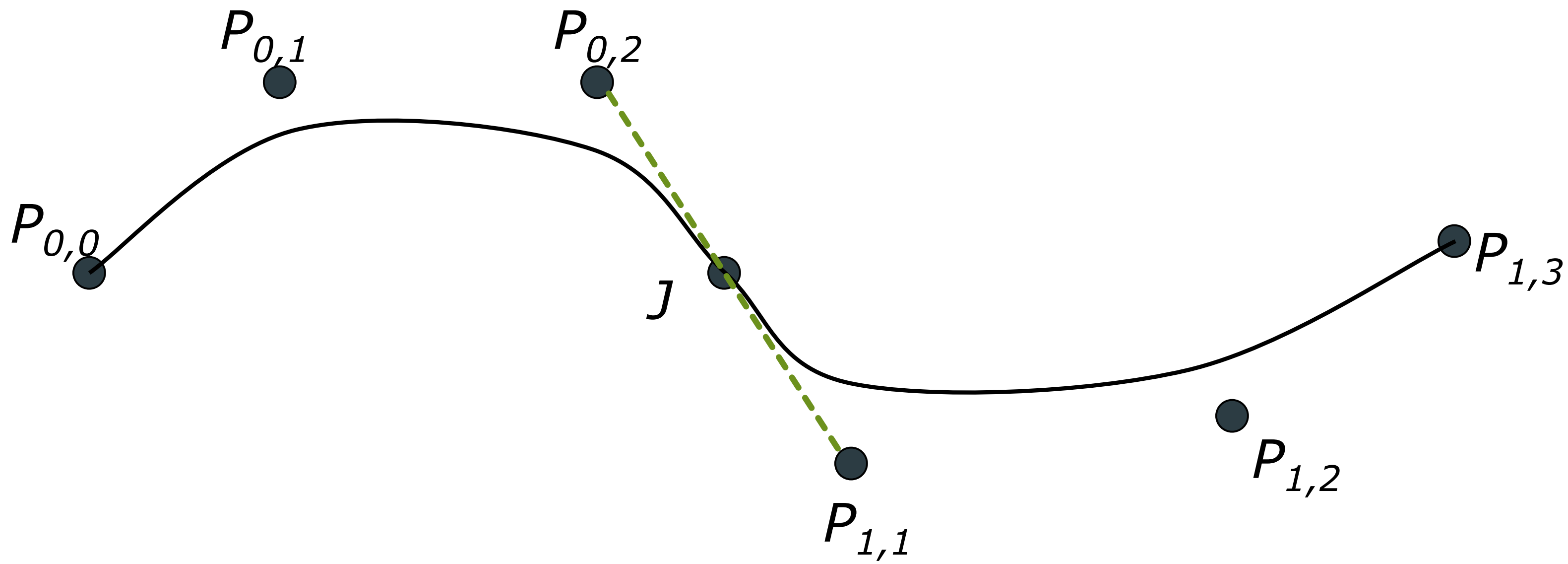
- We can make a long curve by concatenating multiple short Bezier curves.



- How to keep the continuity?

Continuity Properties

- C0 continuous :curve/surface has no breaks
- G1 continuous : tangent at joint has same ***direction***
- C1 continuous : tangent at join has same ***direction and magnitude***
- Cn continuous : curve/surface through ***nth derivative*** is continuous



B-splines

- How to reach both C_2 continuity and local controllability ?
  - Slightly loose the endpoint constraints.
  - B-splines do not interpolate any of control points.

- Uniform cubic B-spline basis functions

$$\mathbf{p}(t) = \sum_{j=0}^{3} p_j B_j^3(t)$$
$$= p_0 B_0^3(t) + p_1 B_1^3(t) + p_2 B_2^3(t) + p_3 B_3^3(t)$$

$$B_0^3(t) = \frac{1}{6}(1-t)^3$$

$$B_1^3(t) = \frac{1}{6}(3t^3 - 6t^2 + 4)$$

$$B_2^3(t) = \frac{1}{6}(-3t^3 + 3t^2 + 3t + 1)$$

$$B_3^3(t) = \frac{1}{6}t^3$$

B-spline Matrix

$$\mathbf{p}(t) = \sum_{j=0}^{3} p_j B_j^3(t)$$

$$= p_0 B_0^3(t) + p_1 B_1^3(t) + p_2 B_2^3(t) + p_3 B_3^3(t)$$

$$B_0^3(t) = \frac{1}{6}(1-t)^3$$

$$B_1^3(t) = \frac{1}{6}(3t^3 - 6t^2 + 4)$$

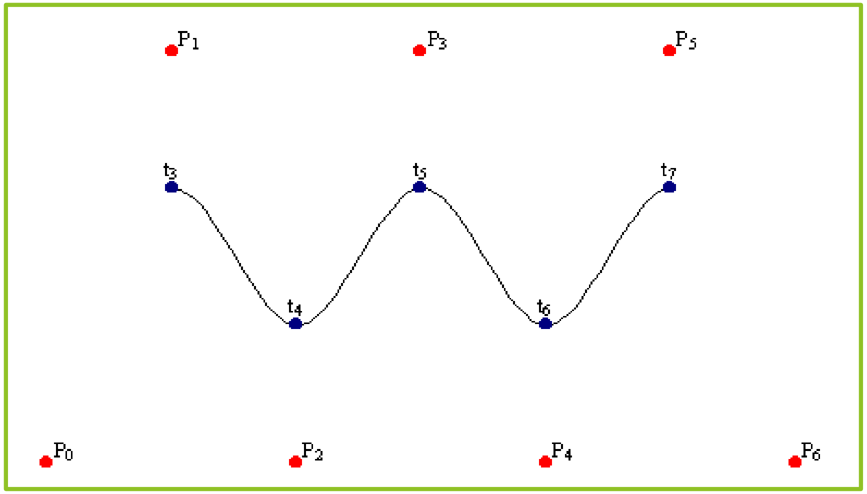
$$B_2^3(t) = \frac{1}{6}(-3t^3 + 3t^2 + 3t + 1)$$

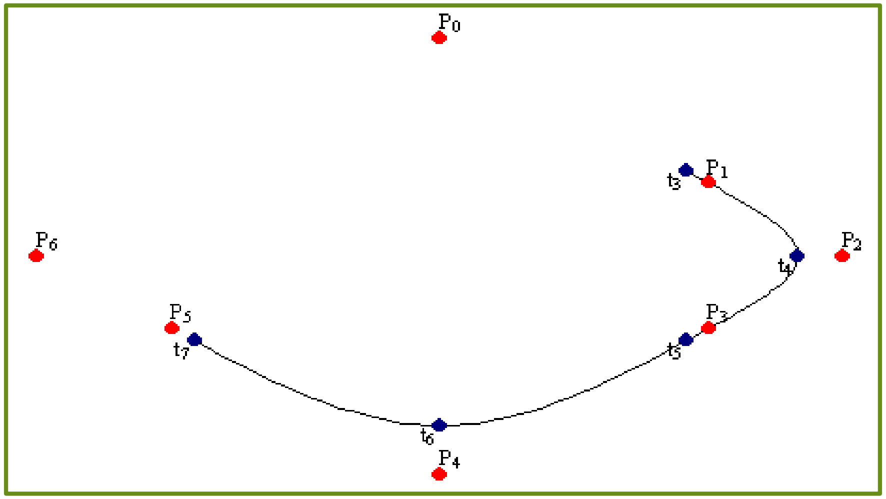
$$B_3^3(t) = \frac{1}{6}t^3$$

$$P(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

B-spline curves

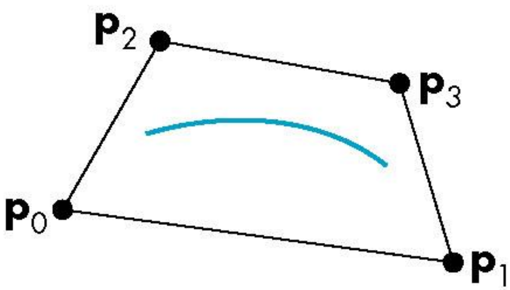
- Start with a sequence of control points
- Select four from middle of sequence (p_{i-2}, p_{i-1}, p_i, p_{i+1})
- Bezier and Hermite goes between pi-2 and pi+1
- B-Spline doesn't interpolate (touch) any of them but approximates going through p_{i-1} and p_i .



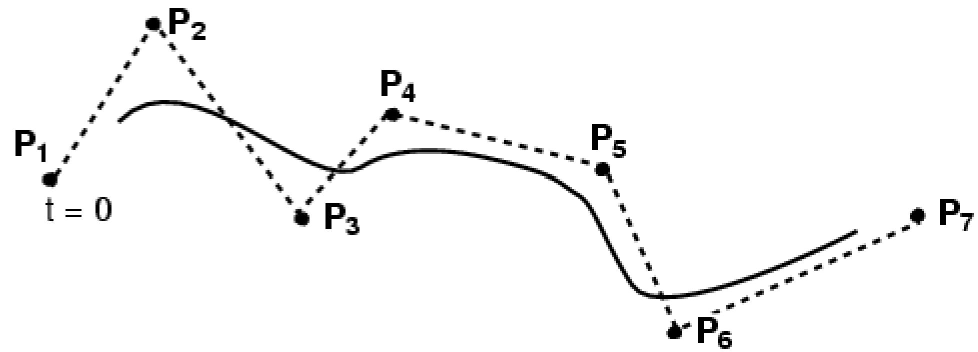


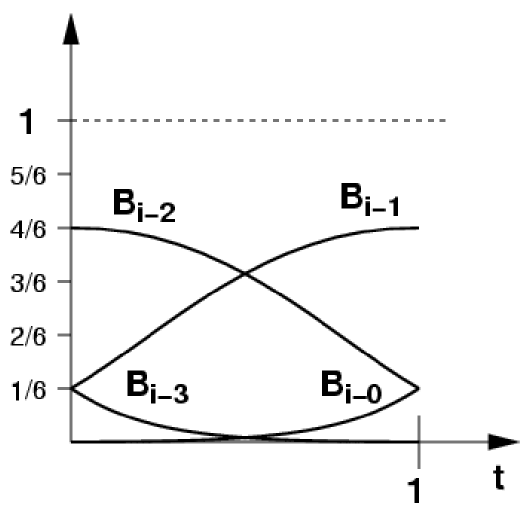
Figures from CG lecture note, U. Virginia

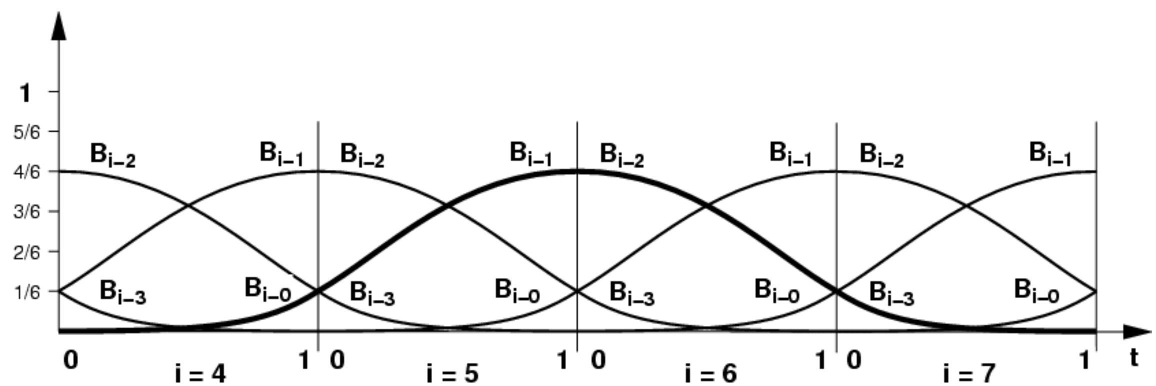
The Blending Weights



Convex hull property





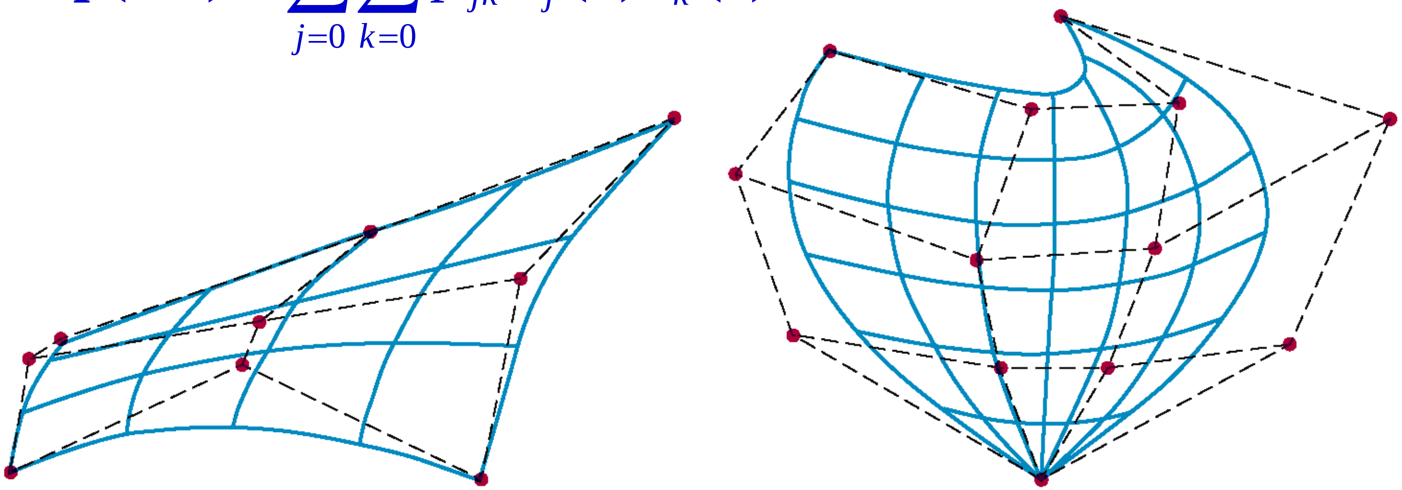


Figures from MIT EECS 6.837, Durand and Cutler

Bezier Patch

- Bezier curves can be extended to surfaces {from *t* to (u,v)}.

$$\mathbf{p}(u,v) = \sum_{j=0}^{n} \sum_{k=0}^{n} p_{jk} B_j^n(u) B_k^n(v)$$



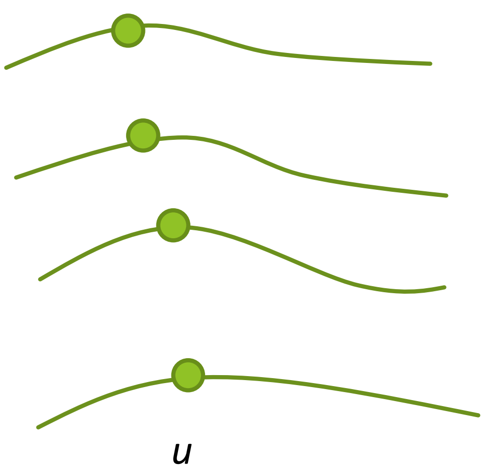
(a) (b)

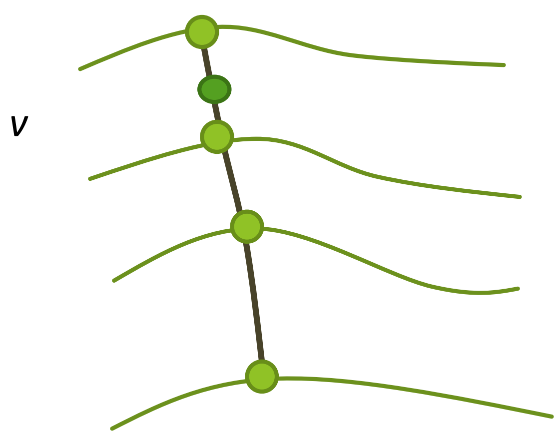
Figure 8-39

Wire-frame Bézier surfaces constructed with (a) nine control points arranged in a 3 by 3 mesh and (b) sixteen control points arranged in a 4 by 4 mesh. Dashed lines connect the control points.

Bezier Patch

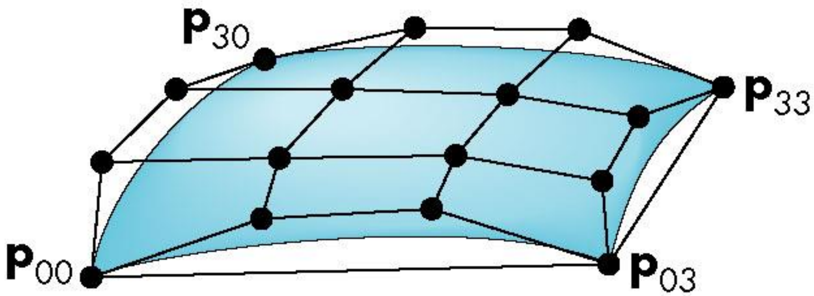
- Edge curves are Bezier curves.
- Any curve of constant *u* or *v* is a Bezier curve
  - Each row of 4 control points defines a Bezier curve in *u*
  - Evaluating each of these curves at the same *u* provides 4 virtual control points
  - The virtual control points define a Bezier curve in t
  - Evaluating this curve at v gives the point *p*(*u*,*v*)





Matrix Form of Bezier Patch

Patch lies in
convex hull



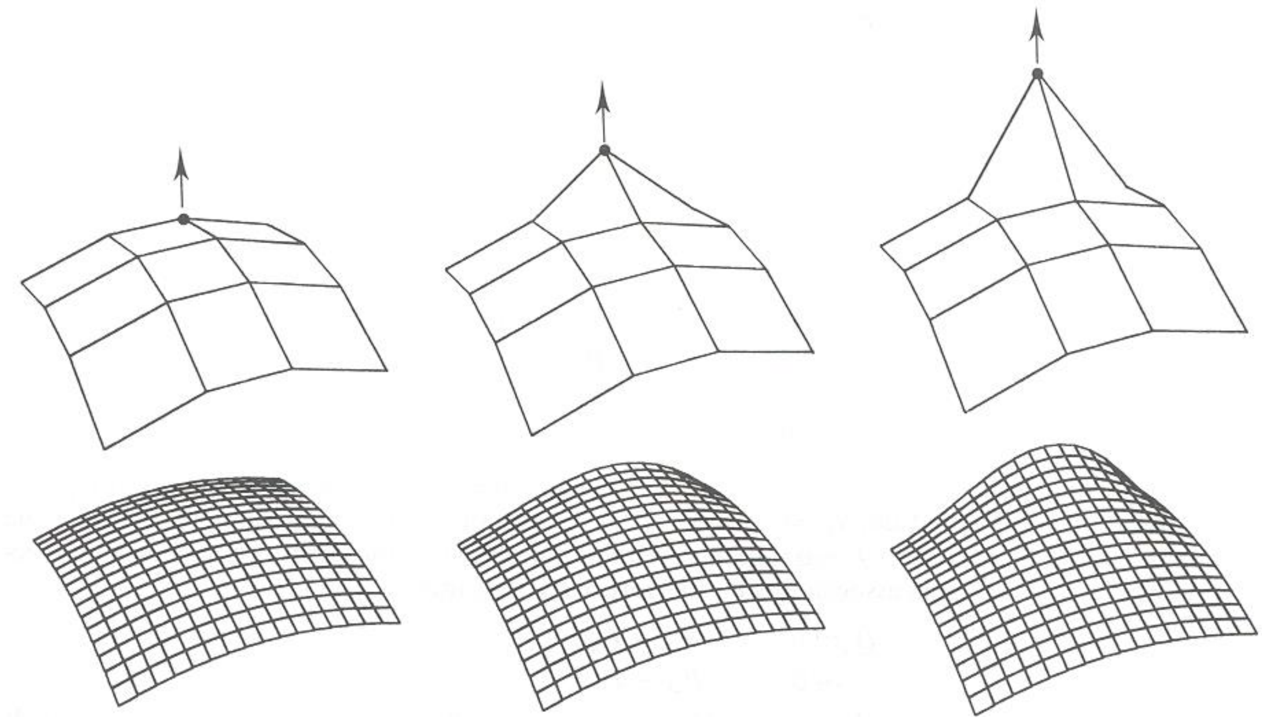
$$\mathbf{p}(u,v) = \sum_{j=0}^{n}\sum_{k=0}^{n} p_{jk} B_j^n(u) B_k^n(v)$$

$$= \mathbf{U}^T \cdot \mathbf{M} \cdot \mathbf{G} \cdot \mathbf{M}^T \cdot \mathbf{V}$$

$$= \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_{00} & p_{10} & p_{20} & p_{30} \\ p_{01} & p_{11} & p_{21} & p_{31} \\ p_{02} & p_{12} & p_{22} & p_{32} \\ p_{03} & p_{13} & p_{23} & p_{33} \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

Bezier Patches

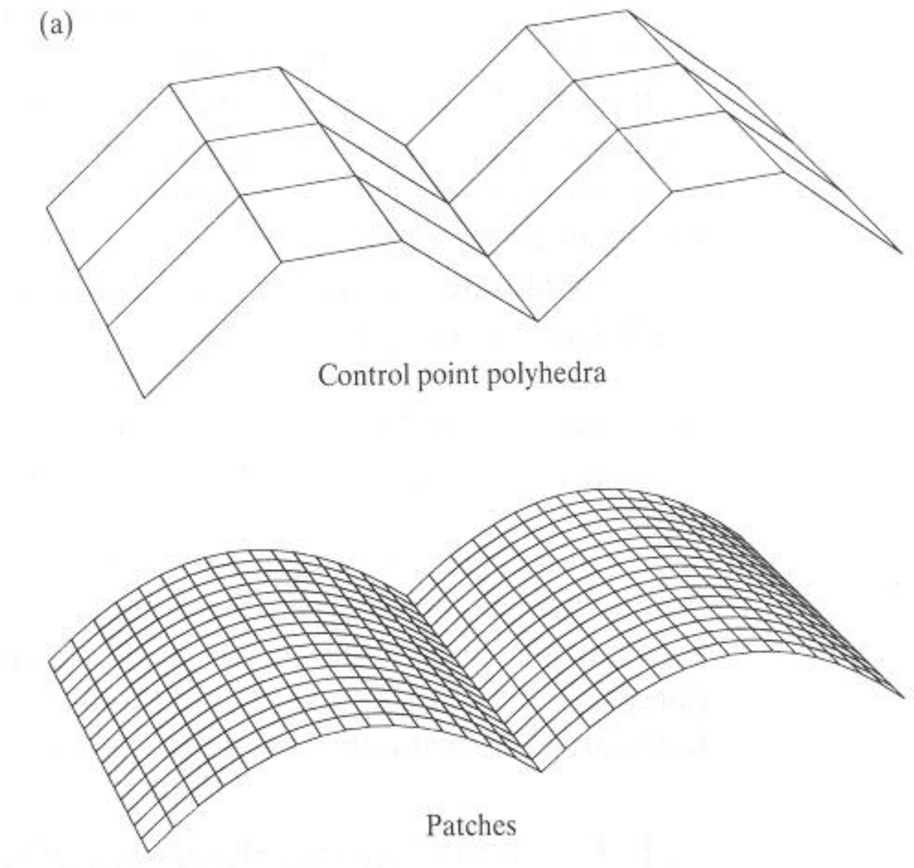
- Interpolates four corner points
- Convex hull property



Watt, 3D Graphics, Figure 6.22

Bezier Surfaces

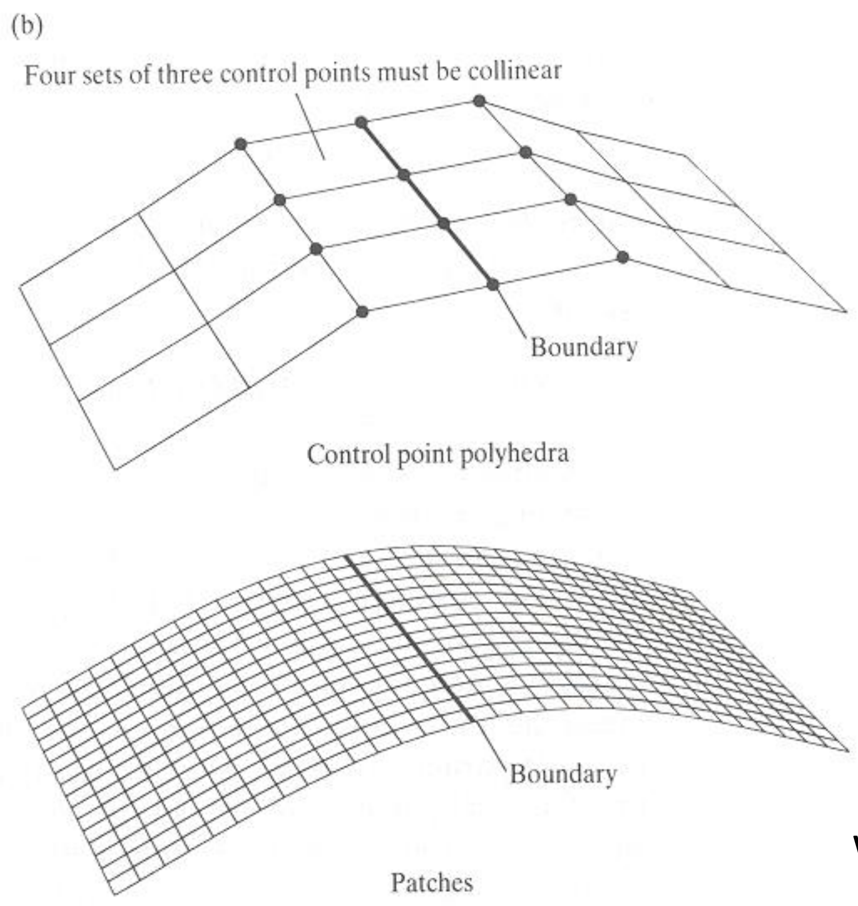
- C0 continuity requires aligning boundary curves



Watt, 3D Graphics, Figure 6.26

Bezier Surfaces

- C1 continuity requires aligning boundary curves and derivatives

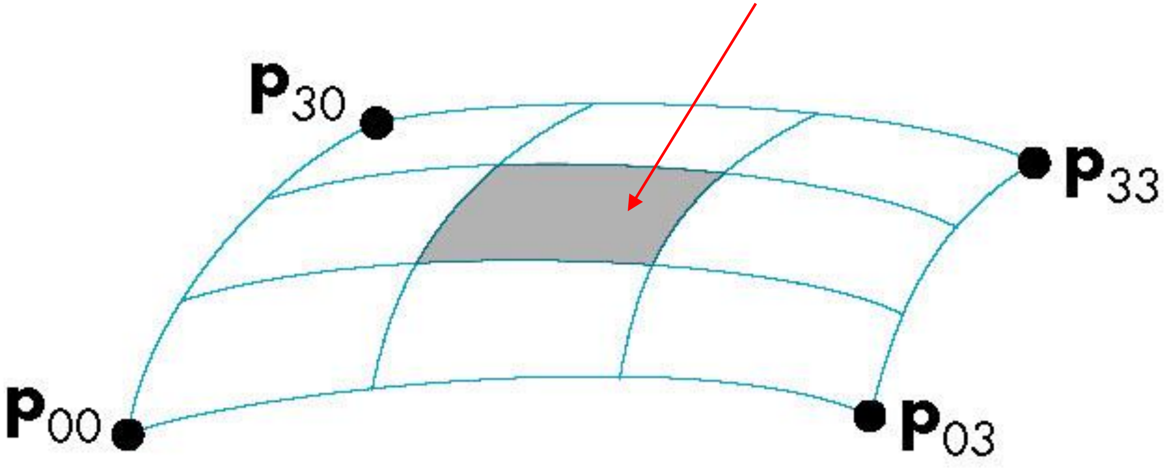


Watt, 3D Graphics, Figure 6.26

B-Spline Surface (Patch)

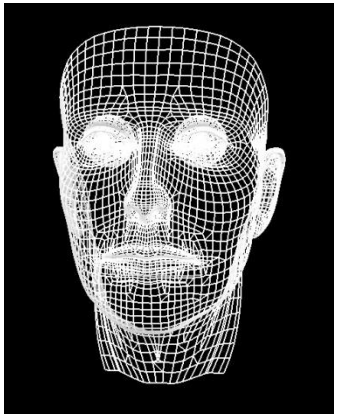
$$p(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} b_i(u)\, b_j(v)\, p_{ij} = u^T \mathbf{M}_S \mathbf{P} \mathbf{M}_S^T v$$

defined over only 1/9 of region

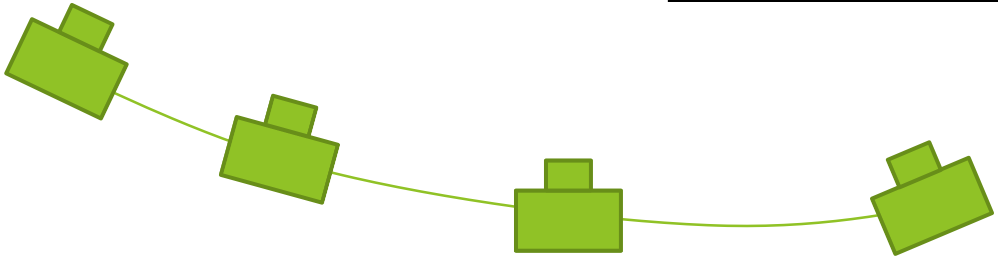


Applications of Splines and Surfaces

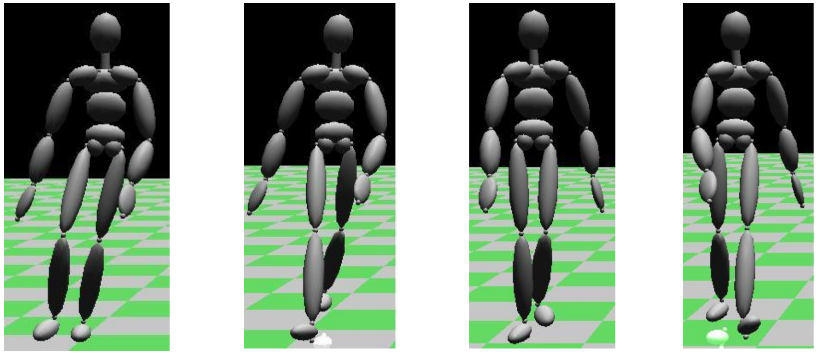
- Modeling and editing 3D objects.



- Smooth paths (e.g. camera views)



- Key-frame animation.



- Etc….