Introduction to Computer Graphics

2016 Spring

National Cheng Kung University

Instructors: Min-Chun Hu 胡敏君

Shih-Chin Weng 翁士欽 (西基電腦動畫)

Data Representation

Curves and Surfaces

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.

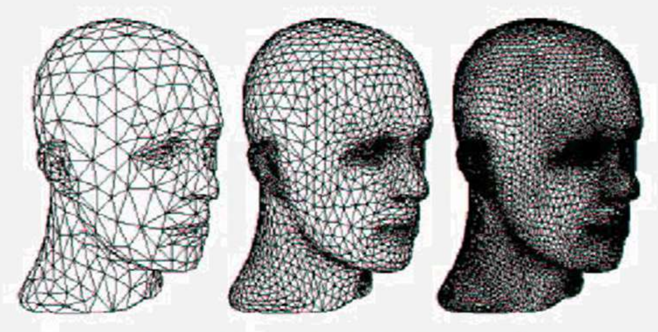
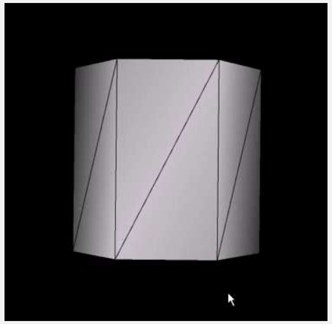
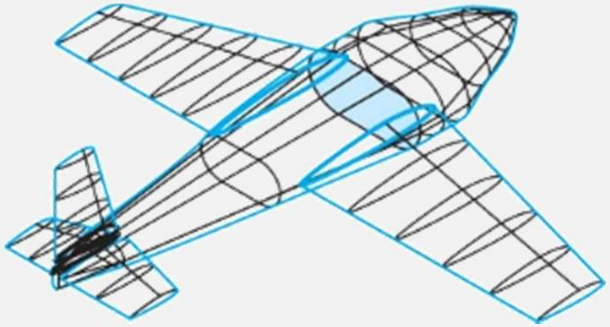


Figure from Zorin & Schroeder SIGGRAPH 99 Course Notes

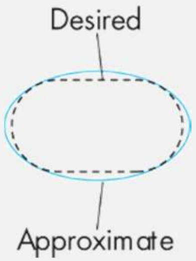


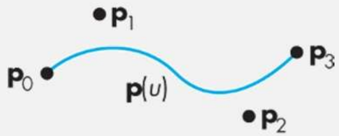
Design Criteria

- How to build an airplane using flexible strips of wood?

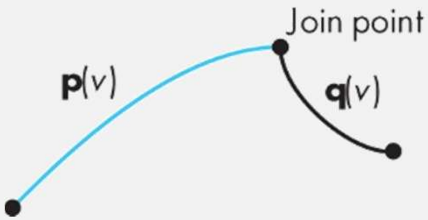


Desired cross-section curve





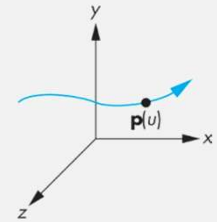
The approximate curve can be determined by the control/data points.

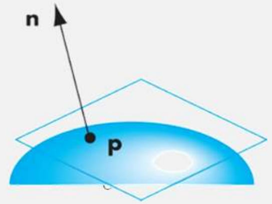


Avoid derivative discontinuity at the join point

Representation of Curves and Surface

- Explicit Representation:
  - $y = f(x)$ or $x = g(y)$
  - No guarantee that either form exists for a given curve (e.g., vertical line and circle)
- Implicit Representation:
  - $f(x, y) = 0$
    - Line: $ax + by + cz + d = 0$
    - Circle: $x^2 + y^2 + z^2 - r^2 = 0$
  - Does represent all lines and circles, but difficult to obtain all points on the curve/surface
- Parametric Representation:
  - Curve: $\mathbf{p}(u) = [x(u) \quad y(u) \quad z(u)]^T$
    - $\frac{d\mathbf{p}(u)}{du} = \left[\frac{dx(u)}{du} \quad \frac{dy(u)}{du} \quad \frac{dz(u)}{du}\right]^T$: the velocity with which the curve is traced out (tangent direction at point $\mathbf{p}$)
  - Surface: $\mathbf{p}(u, v) = [x(u, v) \quad y(u, v) \quad z(u, v)]^T$
    - $\frac{\partial \mathbf{p}}{\partial u} = \left[\frac{\partial x(u,v)}{\partial u} \quad \frac{\partial y(u,v)}{\partial u} \quad \frac{\partial z(u,v)}{\partial u}\right]^T$ and $\frac{\partial \mathbf{p}}{\partial v} = \left[\frac{\partial x(u,v)}{\partial v} \quad \frac{\partial y(u,v)}{\partial v} \quad \frac{\partial z(u,v)}{\partial v}\right]^T$: tangent plane at point $\mathbf{p}$
    - $\mathbf{n} = \frac{\partial \mathbf{p}}{\partial u} \times \frac{\partial \mathbf{p}}{\partial u}$: normal direction
  - Most flexible and robust form for computer graphics, but not unique





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

Parametric Cubic Polynomial Curves

- We can use polynomial functions to form curves:
  - $\mathbf{p}(u) = c_0 + c_1 u + c_2 u^2 + \cdots c_n u^n$
  - degree of freedom: n+1
  - cubic polynomial curve: $\mathbf{p}(u) = \boldsymbol{c_0} + \boldsymbol{c_1} u + \boldsymbol{c_2} u^2 + \boldsymbol{c_3} u^3 = \mathbf{u}^T \mathbf{c}$
    - Low freedom, but sufficient to produce the desired shape in a small region
- The problem is how to efficiently find out the coefficient $\boldsymbol{c_i}$
- Least square curve fitting:

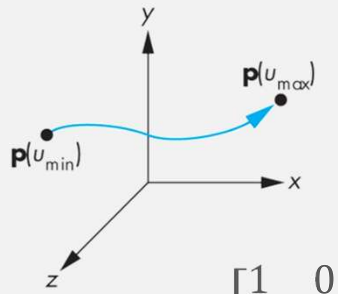
$$x(u) = c_{x0} + c_{x1} u + c_{x2} u^2 + c_{x3} u^3$$
$$y(u) = c_{y0} + c_{y1} u + c_{y2} u^2 + c_{y3} u^3$$
$$z(u) = c_{z0} + c_{z1} u + c_{z2} u^2 + c_{z3} u^3$$

⇨ Need 4 points to solve 12 unknowns

Least Square Curve Fitting

$$\mathbf{p}(u) = c_0 + c_1 u + c_2 u^2 + c_3 u^3$$

- Given 4 control points $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$, and $\mathbf{p}_3$
  - Assume the 4 points are with equally spaced values u:



$$\left.\begin{aligned}
\mathbf{p}_0 &= \mathbf{p}(0) = \mathbf{c}_0 \\
\mathbf{p}_1 &= \mathbf{p}\left(\frac{1}{3}\right) = \mathbf{c}_0 + \frac{1}{3}\mathbf{c}_1 + \left(\frac{1}{3}\right)^2 \mathbf{c}_2 + \left(\frac{1}{3}\right)^3 \mathbf{c}_3 \\
\mathbf{p}_2 &= \mathbf{p}\left(\frac{2}{3}\right) = \mathbf{c}_0 + \frac{2}{3}\mathbf{c}_1 + \left(\frac{2}{3}\right)^2 \mathbf{c}_2 + \left(\frac{2}{3}\right)^3 \mathbf{c}_3 \\
\mathbf{p}_3 &= \mathbf{p}(1) = \mathbf{c}_0 + \mathbf{c}_1 + \mathbf{c}_2 + \mathbf{c}_3
\end{aligned}\right\} \Rightarrow \mathbf{P} = \mathbf{A}\mathbf{c}$$

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & \frac{1}{3} & \left(\frac{1}{3}\right)^2 & \left(\frac{1}{3}\right)^3 \\ 1 & \frac{2}{3} & \left(\frac{2}{3}\right)^2 & \left(\frac{2}{3}\right)^3 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad \mathbf{c} = \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \end{bmatrix}$$

nonsingular

$$\mathbf{M}_I = \mathbf{A}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -5.5 & 9 & -4.5 & 1 \\ 9 & -22.5 & 18 & -4.5 \\ -4.5 & 13.5 & -13.5 & 4.5 \end{bmatrix} \quad \Rightarrow \mathbf{c} = \mathbf{M}_I \mathbf{P}$$

Interpolating Geometry Matrix

Cubic Interpolating Curves

- Rather than deriving a single interpolating curve of degree *m* for all the points, derive a set of cubic interpolating curves.
- If each segment is derived by letting *u* varying equally over the interval [0,1], then the matrix $\mathbf{M_I}$ is the same for each segment.
- Derivatives at the joint points will not be continuous.

Blending Functions

$$\mathbf{p}(u) = c_0 + c_1 u + c_2 u^2 + c_3 u^3 = \mathbf{u}^T \mathbf{c} = \mathbf{u}^T \mathbf{M_I} \mathbf{P} = \mathbf{b}(u)^T \mathbf{P}$$

$$\mathbf{b}(u) = \begin{bmatrix} b_0(u) \\ b_1(u) \\ b_2(u) \\ b_3(u) \end{bmatrix}$$ Blending Polynomials

$$\mathbf{p}(u) = \mathbf{b}(u)^T \mathbf{P} = b_0(u)\mathbf{p}_0 + b_1(u)\mathbf{p}_1 + b_2(u)\mathbf{p}_2 + b_3(u)\mathbf{p}_3 = \sum_{i=0}^{3} b_i(u)\mathbf{p}_i$$

- The polynomials blend together the individual contributions of each control point and enable us to see the effect of a given control point on the entire curve.

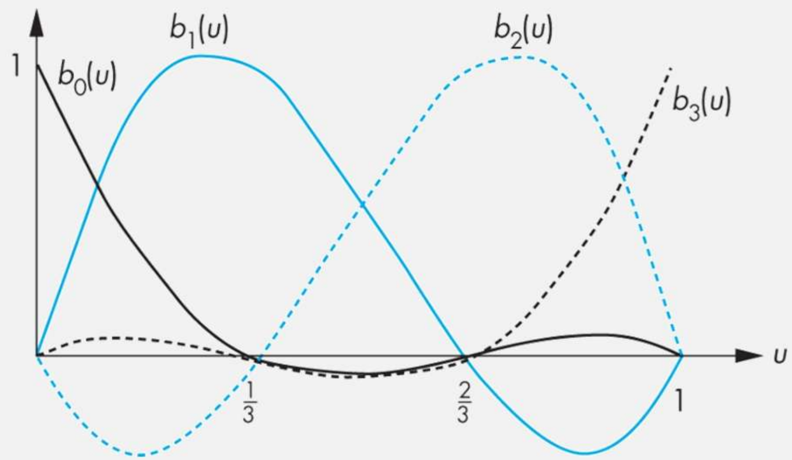
Blending Functions (Cont.)

$$b_0(u) = -\frac{9}{2}\left(u - \frac{1}{3}\right)\left(u - \frac{2}{3}\right)(u - 1)$$

$$b_1(u) = \frac{27}{2}u\left(u - \frac{2}{3}\right)(u - 1)$$

$$b_2(u) = -\frac{27}{2}u\left(u - \frac{1}{3}\right)(u - 1)$$

$$b_3(u) = \frac{9}{2}u\left(u - \frac{1}{3}\right)\left(u - \frac{2}{3}\right)$$

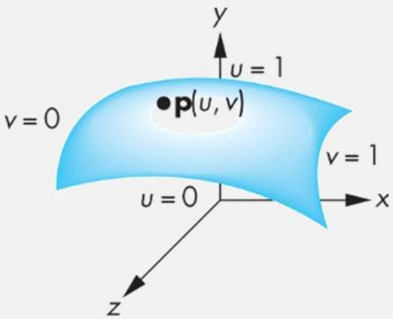


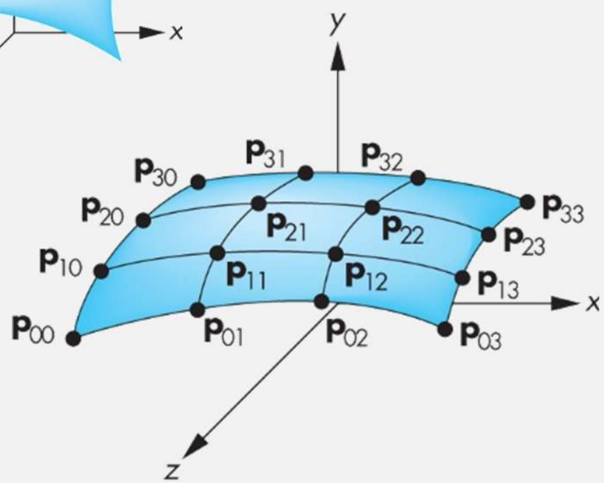
Bicubic Surface Patch

$$\mathbf{p}(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} u^i v^j c_{ij} = \mathbf{u}^T\mathbf{C}\mathbf{v}$$

Need 16 control points to solve the 4x4 unknows in the $\mathbf{C}$ matrix

$$\mathbf{u} = \begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \end{bmatrix} \quad \mathbf{v} = \begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix}$$





- Rather than writing down and solving 16 equations, we consider the curve at $v = 0$ that interpolates $\mathbf{p}_{00}, \mathbf{p}_{10}, \mathbf{p}_{20},$ and $\mathbf{p}_{30}$:

$$\mathbf{p}(u,0) = \mathbf{u}^T\mathbf{M}_{\mathbf{I}}\begin{bmatrix} \mathbf{p}_{00} \\ \mathbf{p}_{10} \\ \mathbf{p}_{20} \\ \mathbf{p}_{30} \end{bmatrix} = \mathbf{u}^T\mathbf{C}\mathbf{v} = \mathbf{u}^T\mathbf{C}\begin{bmatrix} \mathbf{1} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

- Each value of $v = \frac{1}{3}, \frac{2}{3}, 1$ defines a interpolating curve with the similar form

Bicubic Surface Patch (Cont.)

$$\mathbf{p}(u,0) = \mathbf{u}^T\mathbf{M_I}\begin{bmatrix}\mathbf{p}_{00}\\ \mathbf{p}_{10}\\ \mathbf{p}_{20}\\ \mathbf{p}_{30}\end{bmatrix} = \mathbf{u}^T\mathbf{C}\begin{bmatrix}\mathbf{1}\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix} \Rightarrow \mathbf{u}^T\mathbf{M_I}\mathbf{P} = \mathbf{u}^T\mathbf{C}\mathbf{A}^{\boldsymbol{T}}$$

$\mathbf{A}$: inverse of $\mathbf{M_I}$

Consider 4 curves at
4 different v values
(16 equations)

$$\Rightarrow \mathbf{C} = \mathbf{M_I}\mathbf{P}(\mathbf{A}^{\boldsymbol{T}})^{-1} = \mathbf{M_I}\mathbf{P}\mathbf{M_I}^{\boldsymbol{T}}$$

$$\Rightarrow \mathbf{p}(u,v) = \mathbf{u}^T\mathbf{C}\mathbf{v} = \mathbf{u}^T\mathbf{M_I}\mathbf{P}\mathbf{M_I}^{\boldsymbol{T}}\mathbf{v}$$

- Represented in blending functions: $\mathbf{p}(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} b_i(u)b_j(v)\,\mathbf{p}_{ij}$

Hermite Curves

- A Hermite curve is a curve for which the user provides:
  - The endpoints of the curve:

    $\mathbf{p}_0 = \mathbf{p}(0) = \mathbf{c}_0$

    $\mathbf{p}_3 = \mathbf{p}(\mathbf{1}) = \mathbf{c}_0 + \mathbf{c}_1 + \mathbf{c}_2 + \mathbf{c}_3$

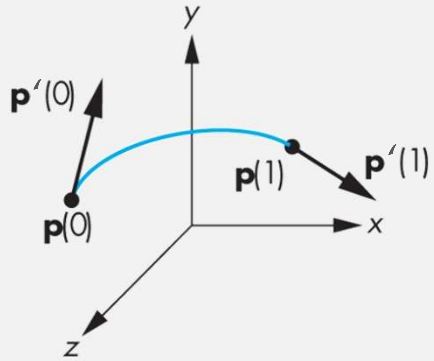
  - The derivatives of the curve at the endpoints:

    $\mathbf{p}'_0 = \mathbf{p}'(0) = \mathbf{c}_1$

    $\mathbf{p}'_3 = \mathbf{p}'(\mathbf{1}) = \mathbf{c}_1 + 2\mathbf{c}_2 + 3\mathbf{c}_3$

$\mathbf{p}(u) = c_0 + c_1 u + c_2 u^2 + c_3 u^3$

$\mathbf{p}'(u) = \mathbf{c}_1 + 2u\mathbf{c}_2 + 3u^2\mathbf{c}_3$



Hemite Geometry Matrix

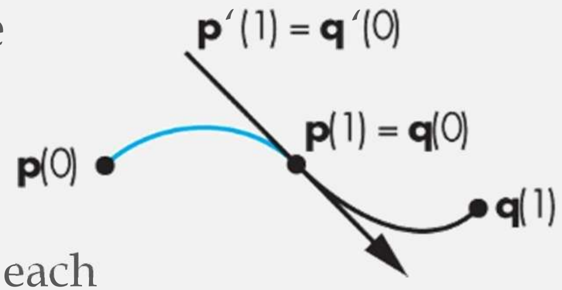
$$\mathbf{Q} = \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_3 \\ \mathbf{p}'_0 \\ \mathbf{p}'_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \mathbf{c} \qquad \mathbf{M}_\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & 2 & 1 & 1 \end{bmatrix}$$

$$\mathbf{c} = \mathbf{M}_\mathbf{H}\mathbf{Q}$$

$$\mathbf{p}(u) = \mathbf{u}^T\mathbf{c} = \mathbf{u}^T\mathbf{M}_H\mathbf{Q}$$

Hermite Curves (Cont.)

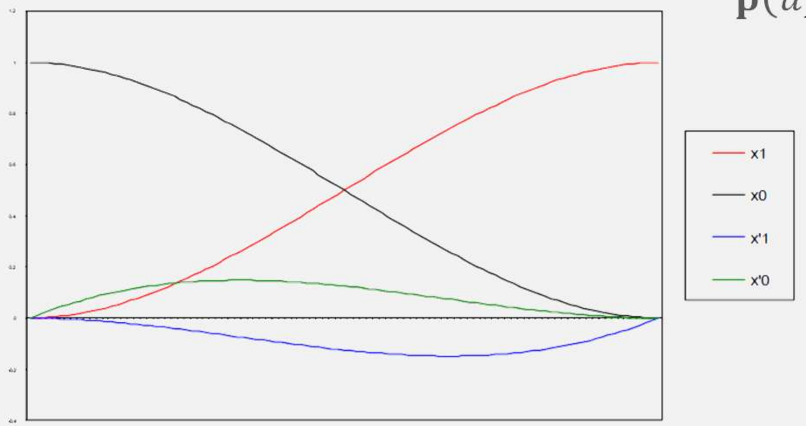
- Both the resulting function and the first derivative are continuous over all segments
- Represented in blending functions:
  - A point on a Hermite curve is obtained by weighted blending each control point and tangent vector.



$$\mathbf{p}(u) = \mathbf{u}^T\mathbf{c} = \mathbf{u}^T\mathbf{M}_H\mathbf{Q}$$

$$\mathbf{p}(u) = \mathbf{b}(u)^T\mathbf{Q}$$

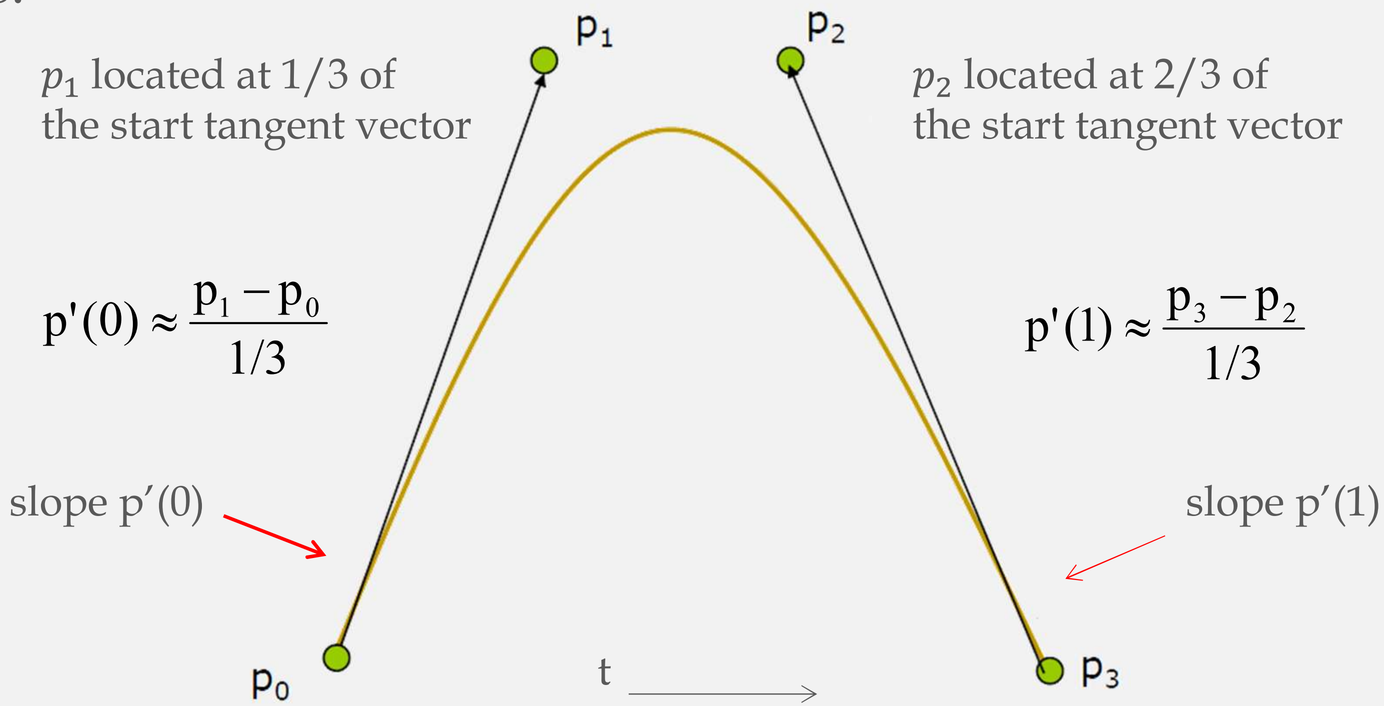
$$\mathbf{b}(u) = \mathbf{M}_H{}^T\mathbf{u} = \begin{bmatrix} 2u^3 - 3u^2 + 1 \\ -2u^3 + 3u^2 \\ u^3 - 2u^2 + u \\ u^3 - u^2 \end{bmatrix}$$



Weights of each component

Bezier Curves

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



Bezier Curves (Cont.)

$\mathbf{p}(u) = c_0 + c_1 u + c_2 u^2 + c_3 u^3$

- The endsite conditions are the same
  - $P(0) = P_0 = c_0$
  - $P(1) = P_3 = c_0 + c_1 + c_2 + c_3$
- Approximating derivative conditions
  - $P'(0) = 3(P_1 - P_0) = c_1$
  - $P'(1) = 3(P_3 - P_2) = c_1 + 2c_2 + 3c_3$
- Replacing the original Hermite matrix.

$$\mathbf{c} = \mathbf{M}_\mathrm{B}\mathbf{P} \qquad \mathbf{M}_\mathrm{B} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

Bezier Geometry Matrix

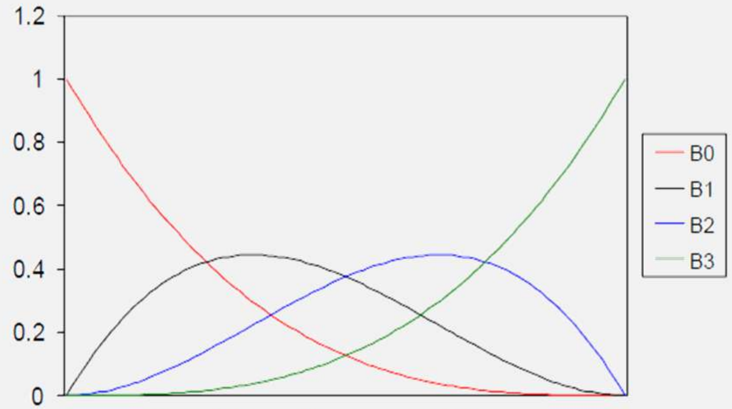
$$\mathbf{p}(u) = \mathbf{u}^T\mathbf{c} = \mathbf{u}^T\mathbf{M}_\mathbf{B}\mathbf{P}$$

Bezier Curves (Cont.)

- Represented in blending functions:
  - A point on a Bezier curve is obtained by weighted blending each control point

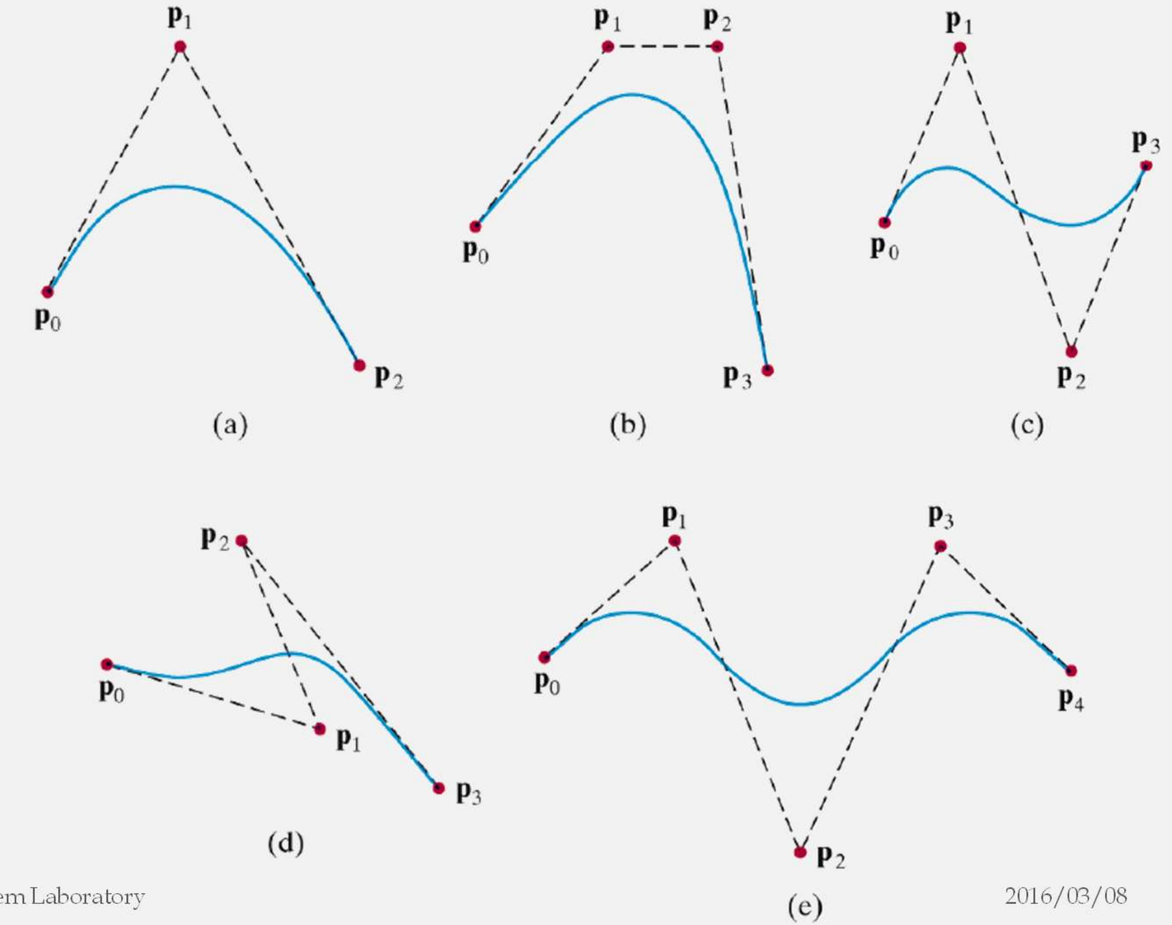
$$\mathbf{p}(u) = \mathbf{b}(u)^T \mathbf{P}$$

$$\mathbf{b}(u) = \mathbf{M_B}^T \mathbf{u} = \begin{bmatrix} (1-u)^3 \\ 3u(1-u)^2 \\ 3u^2(1-u)^2 \\ u^3 \end{bmatrix}$$



The blending weights

Examples of Bezier curves



Bernstein Polynomials

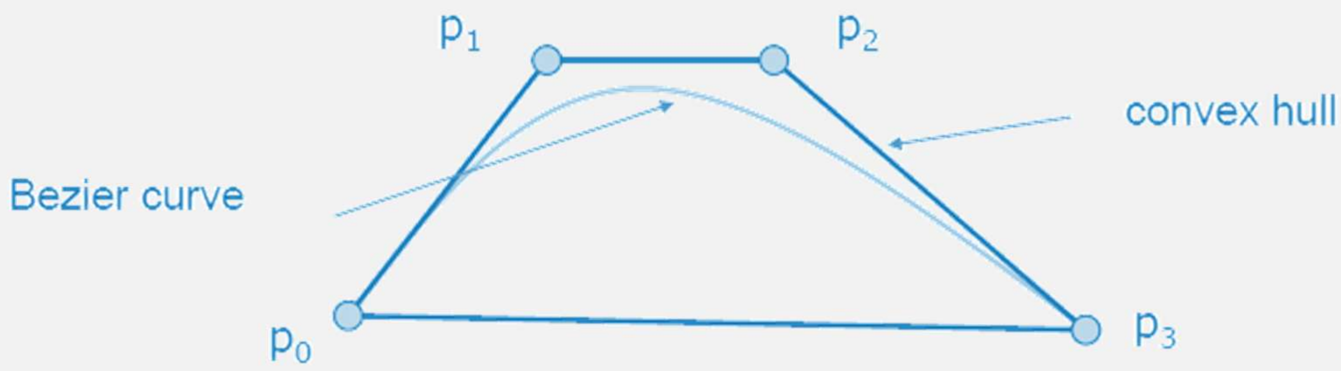
- The blending functions of cubic Bezier curves are a special case of the Bernstein polynomials

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

- These polynomials give the blending polynomials for any degree Bezier form
  - All the zeros are either at u=0 or at u=1
  - For any degree they all sum to 1: $\sum_{i=0}^{d} b_{id}(u) = 1$
  - They are all between 0 and 1 inside [0,1]

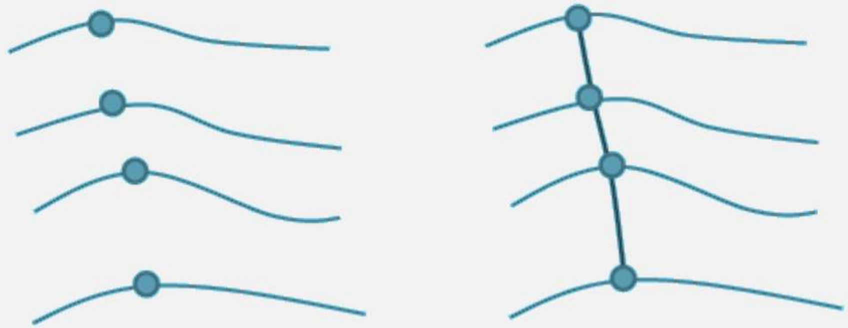
Convex Hull Property

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



Bezier 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 u
  - Evaluating this curve at v gives the point p(u,v)



Bezier Patch (Cont.)

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

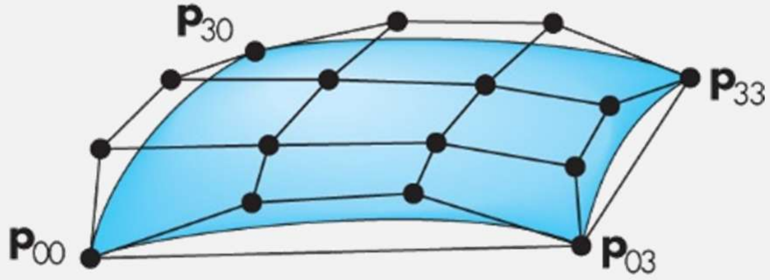
$$\mathrm{p}(u,v)=\sum_{i=0}^{3}\sum_{j=0}^{3} b_i(u)b_j(v)\mathbf{p}_{ij}=\mathbf{u}^T\mathbf{M}_B\mathbf{P}\mathbf{M}_B^T\mathbf{v}$$

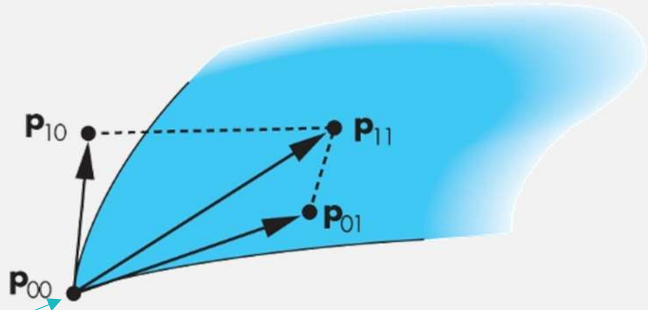
$$\mathbf{p}(0,0)=\mathbf{p}_{00}$$

$$\frac{\partial \mathbf{p}}{\partial u}(0,0)=3(\mathbf{p}_{10}-\mathbf{p}_{00})$$

$$\frac{\partial \mathbf{p}}{\partial \boldsymbol{v}}(0,0)=3(\mathbf{p}_{01}-\mathbf{p}_{00})$$

$$\frac{\partial^2 \mathbf{p}}{\partial u \partial v}(0,0)=9(\mathbf{p}_{00}-\mathbf{p}_{01}+\mathbf{p}_{10}-\mathbf{p}_{11})$$

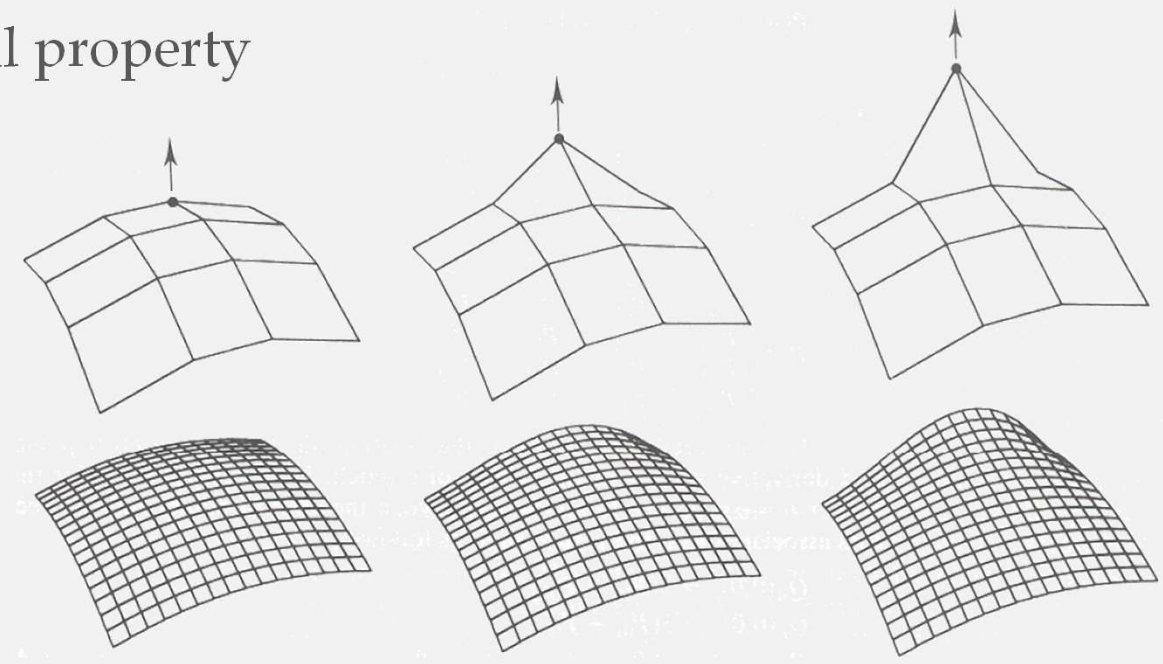




Twist at the corner

Bezier Patches

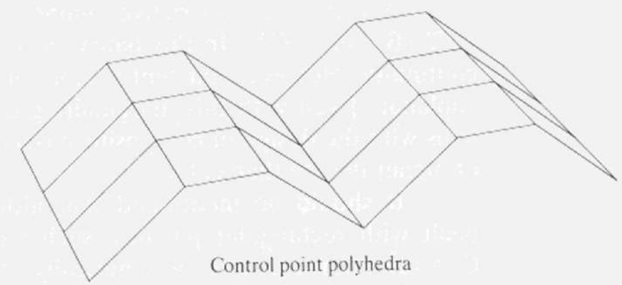
- Interpolates four corner points
- Convex hull property

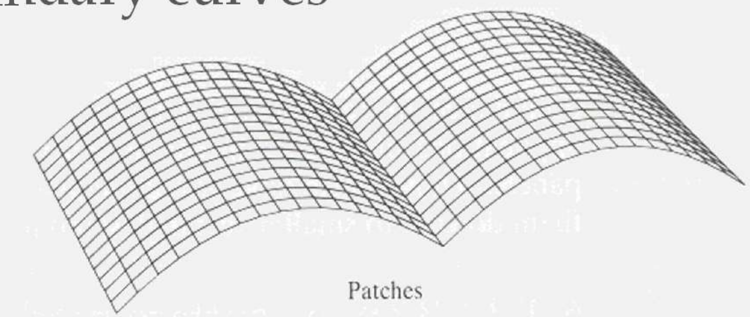


Watt, 3D Graphics

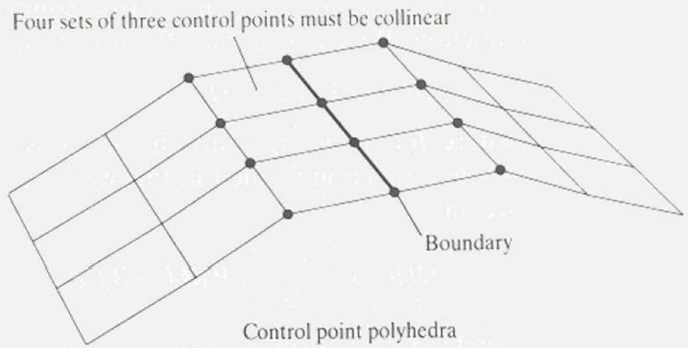
Bezier Surfaces

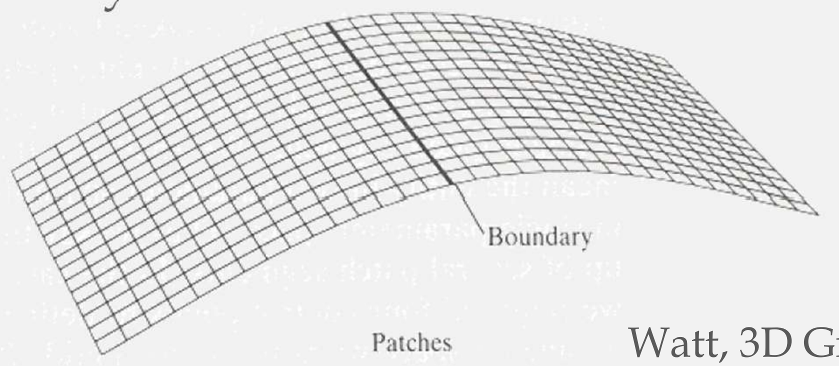
- C0 continuity requires aligning boundary curves





- C1 continuity requires aligning boundary curves and derivatives





Watt, 3D Graphics

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

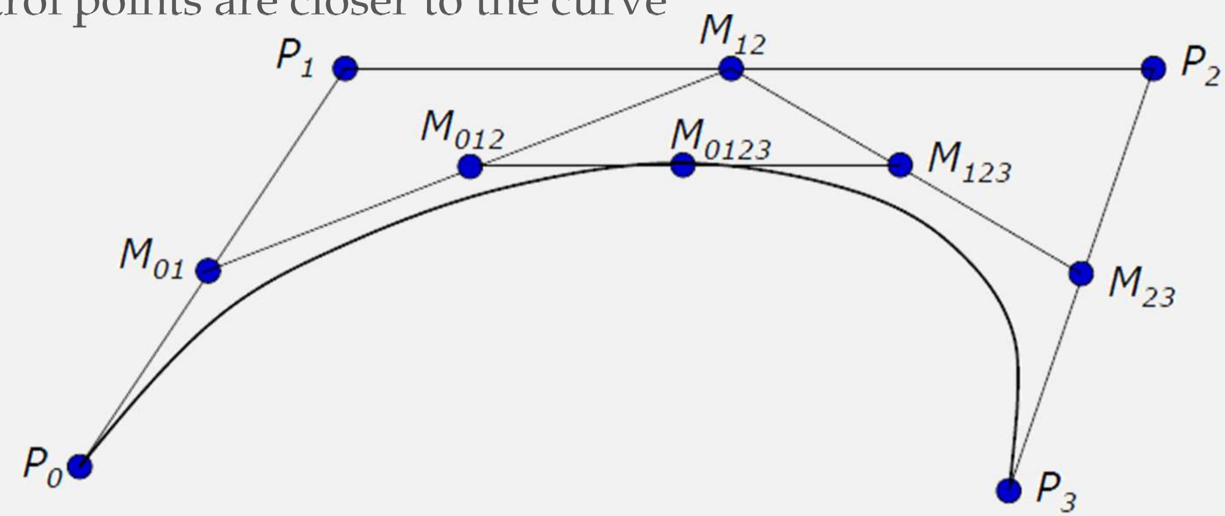
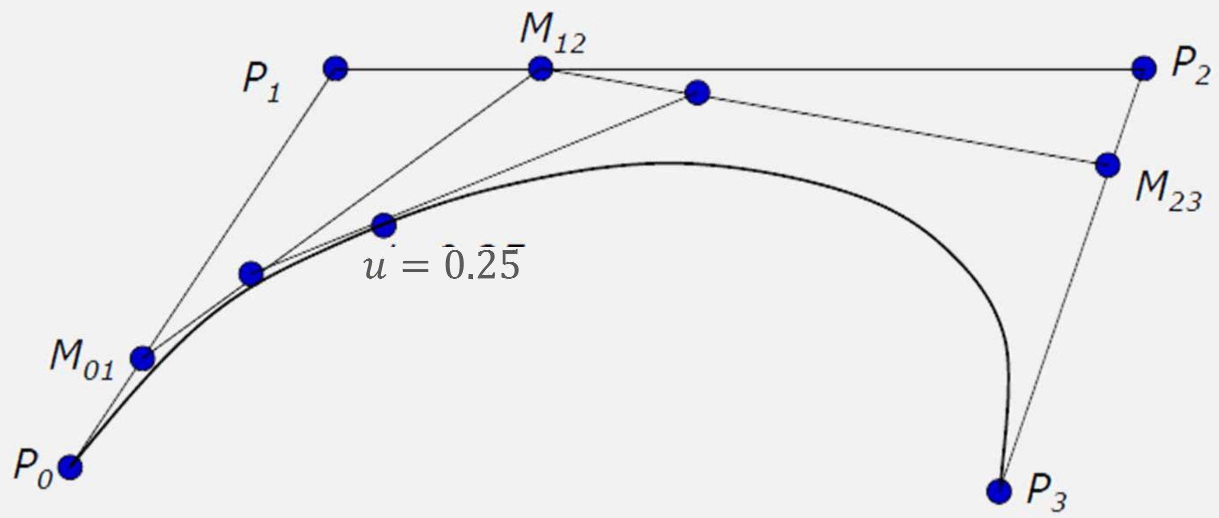


Figure from Prof. S.Chenney, Computer Graphics coursenote, Univ. Wisconsin

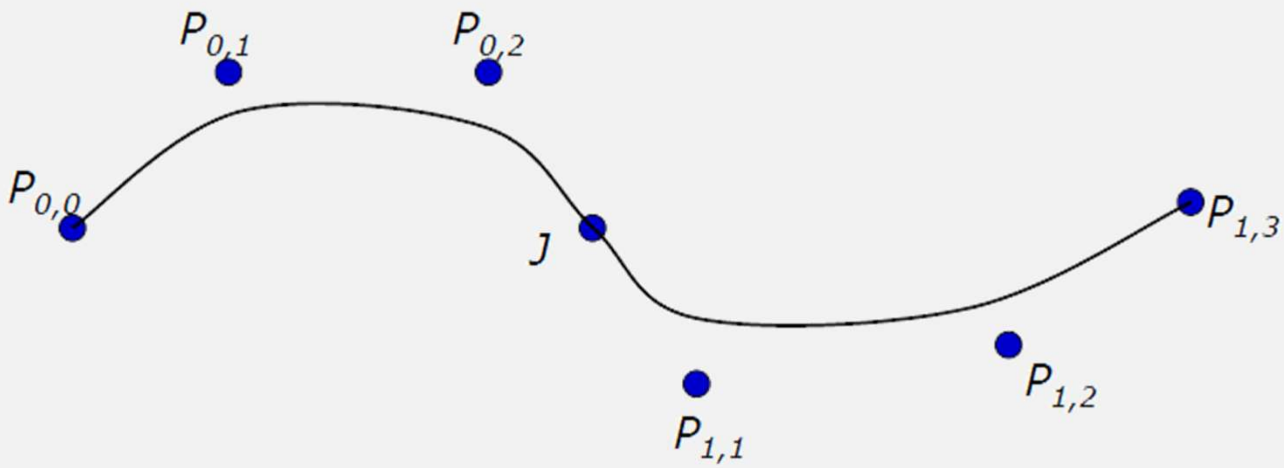
de Casteljau's Algorithm

- You can find the point on a Bezier curve for any parameter value u by subdivision
  - Eg. u=0.25



Bezier Continuity

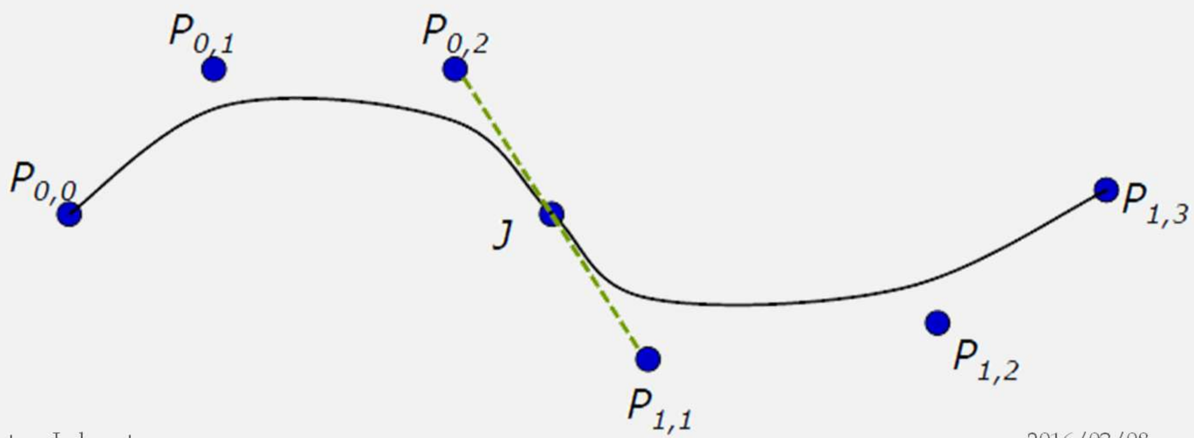
- We can make a long curve by concatenating multiple short Bezier curves.



- How to keep the continuity?

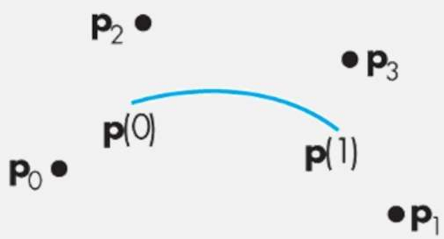
Continuity Properties

- C^0 continuous :curve/surface has no breaks
- G^1 continuous : tangent at joint has same direction
- C^1 continuous : tangent at join has same direction and magnitude
- C^n continuous : curve/surface through nth derivative is continuous



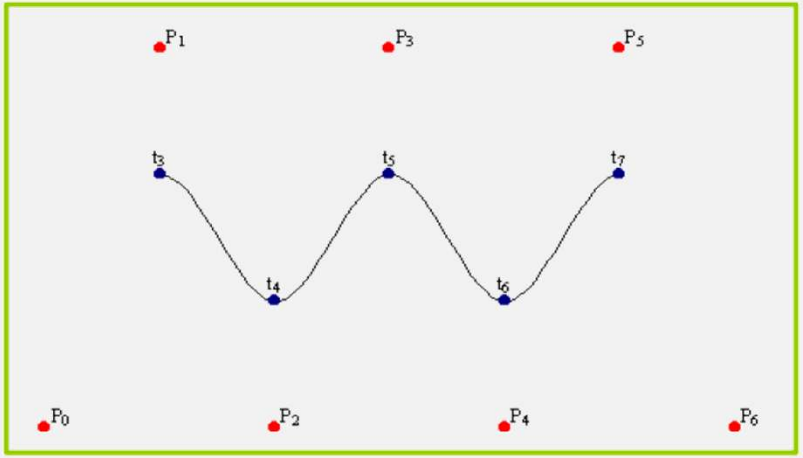
B-splines

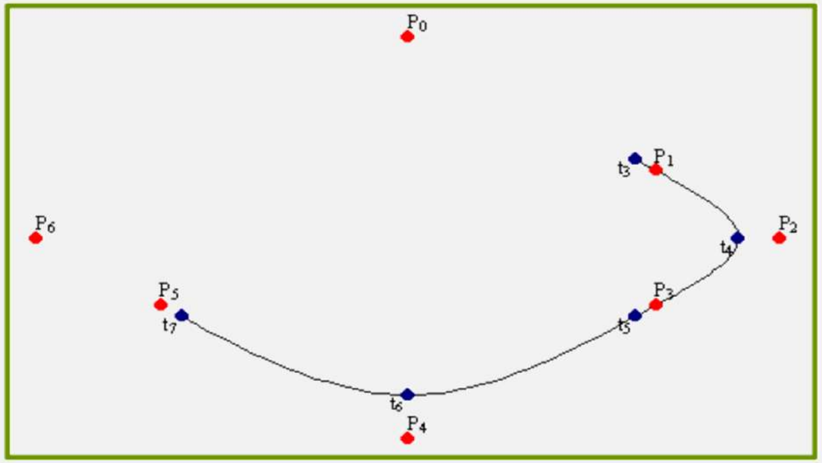
- How to reach both C^2 continuity and local controllability ?
  - Slightly loose the endpoint constraints.
  - B-splines do not interpolate any of control points.



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 p_{i-2} and p_{i+1}
- B-Spline doesn't interpolate (touch) any of them but approximates the curve by going through p_{i-1} and p_i.





Figures from CG lecture note, U. Virginia

B-spline Curves (Cont.)

$$\mathbf{p}(0) = \mathbf{q}(1) = \frac{1}{6}(\mathbf{p}_{i-2} + 4\mathbf{p}_{i-1} + \mathbf{p}_i)$$

$$\mathbf{p}'(0) = \mathbf{q}'(1) = \frac{1}{2}(\mathbf{p}_i - \mathbf{p}_{i-2})$$

Since $\mathbf{p}(u) = \mathbf{c}_0 + \mathbf{c}_1 u + \mathbf{c}_2 u^2 + \mathbf{c}_3 u^3 = u^T \mathbf{c}$

$$\mathbf{p}(0) = \mathbf{c}_0 = \frac{1}{6}(\mathbf{p}_{i-2} + 4\mathbf{p}_{i-1} + \mathbf{p}_i)$$

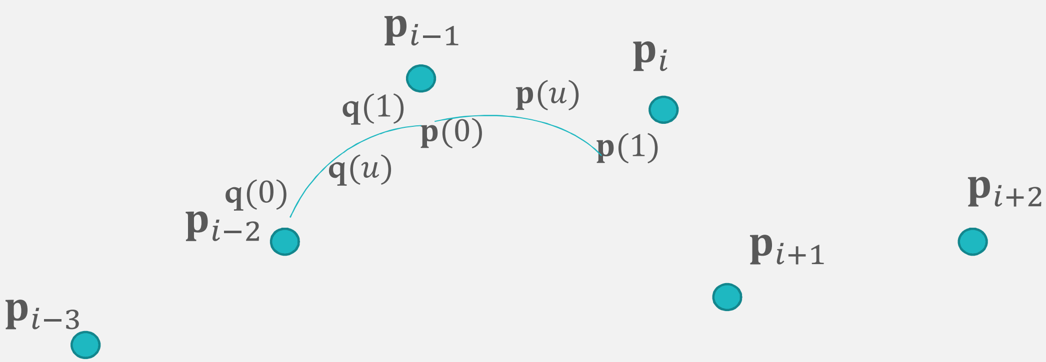
$$\mathbf{p}'(0) = \mathbf{c}_1 = \frac{1}{2}(\mathbf{p}_i - \mathbf{p}_{i-2})$$

$$\mathbf{p}(1) = \mathbf{c}_0 + \mathbf{c}_1 + \mathbf{c}_2 + \mathbf{c}_3 = \frac{1}{6}(\mathbf{p}_{i-1} + 4\mathbf{p}_i + \mathbf{p}_{i+1})$$

$$\mathbf{p}'(1) = \mathbf{c}_1 + 2\mathbf{c}_2 + 3\mathbf{c}_3 = \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_{i-1})$$

$$\Rightarrow \mathbf{P} = \mathbf{A}\mathbf{c}$$

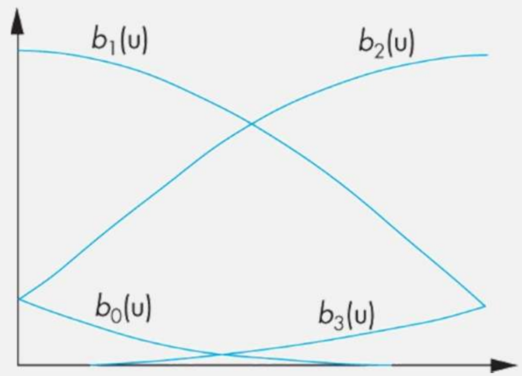
$$\mathbf{M}_S = \mathbf{A}^{-1} = \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 & 0 \\ -3 & 0 & 3 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \quad \Rightarrow \mathbf{c} = \mathbf{M}_S \mathbf{P}$$



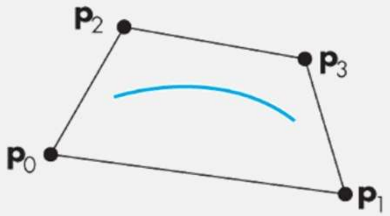
The Blending Weights

$$\mathbf{p}(u) = c_0 + c_1\, u + c_2 u^2 + c_3 u^3 = \mathbf{u}^T \mathbf{c} = \mathbf{u}^T \mathbf{M}_S \mathbf{P} = \mathbf{b}(u)^T \mathbf{P}$$

$$\mathbf{b}(u) = \begin{bmatrix} b_0(u) \\ b_1(u) \\ b_2(u) \\ b_3(u) \end{bmatrix} = \mathbf{M_S}^T \mathbf{u} = \frac{1}{6} \begin{bmatrix} (1-u)^3 \\ 4 - 6u^2 + 3u^2 \\ 1 + 3u + 3u^2 - 3u^3 \\ u^3 \end{bmatrix}$$

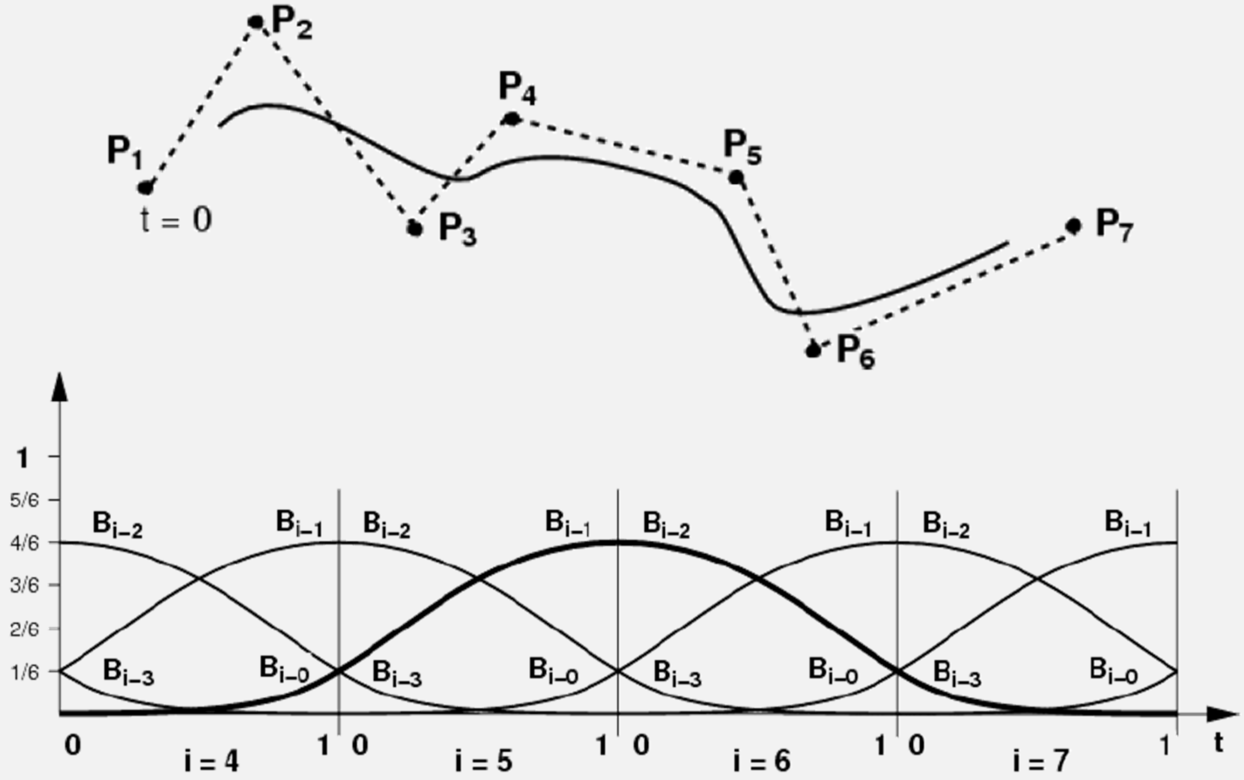


$\sum_{i=0}^{3} b_i(u) = 1$ and $0 < b_i(u) < 1$ in the interval $0 < u < 1$



The curve must lie in the convex hull

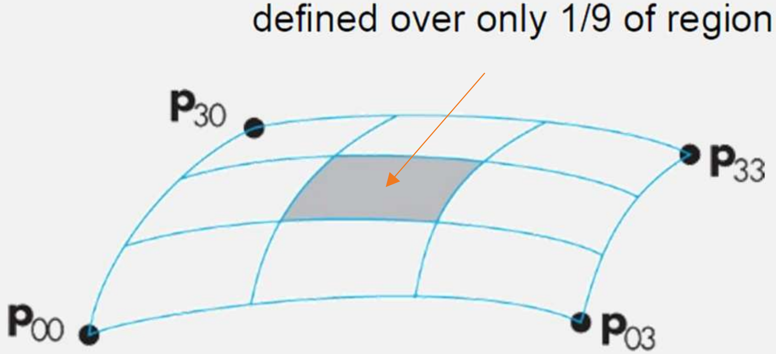
The Blending Weights (Cont.)



Figures from MIT EECS 6.837, Durand and Cutler

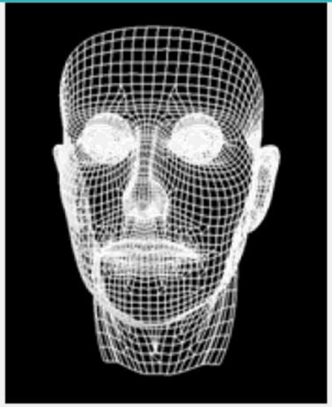
B-Spline Surface (Patch)

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

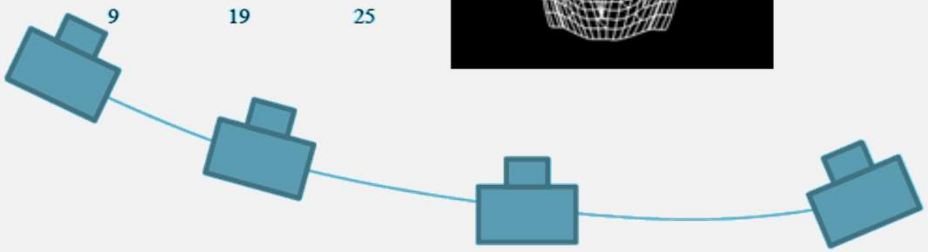


Applications of Splines and Surfaces

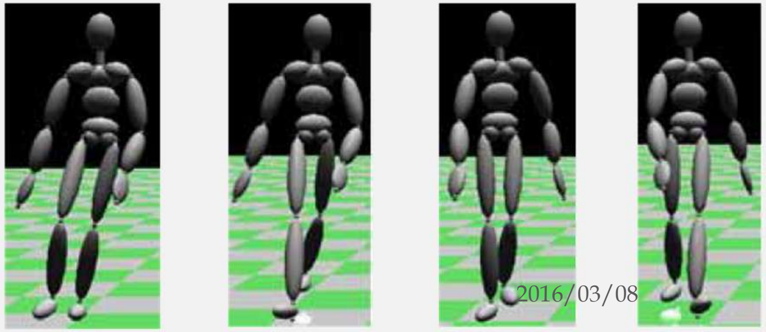
- Modeling and editing 3D objects.



- Smooth paths (e.g. camera views)



- Key-frame animation.



- etc….