



Parametric Surfaces

COS 426, Spring 2019

Adam Finkelstein

Princeton University

3D Object Representations



- Points
 - Range image
 - Point cloud
- Surfaces
 - Polygonal mesh
 - Parametric
 - Subdivision
 - Implicit
- Solids
 - Voxels
 - BSP tree
 - CSG
 - Sweep
- High-level structures
 - Scene graph
 - Application specific

Parametric Surfaces



- Applications
 - Design of smooth surfaces in cars, ships, etc.



Parametric Surfaces



- Applications
 - Design of smooth surfaces in cars, ships, etc.

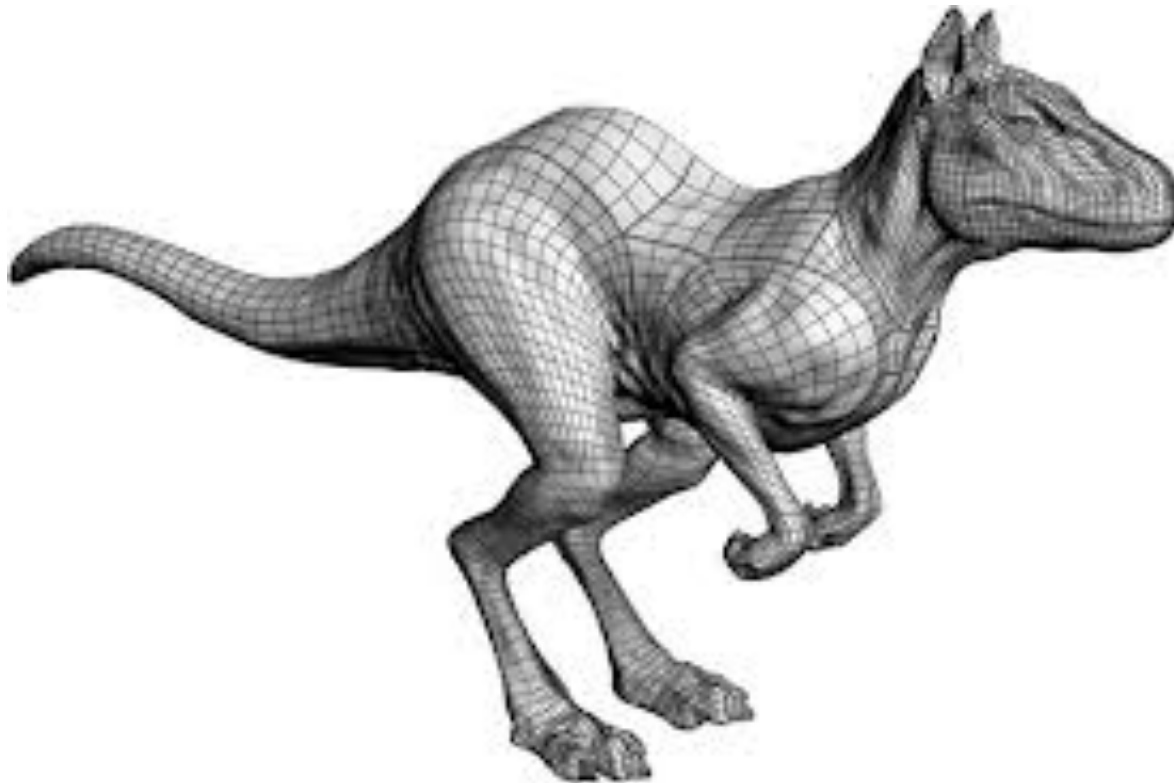
Visualization



Parametric Surfaces



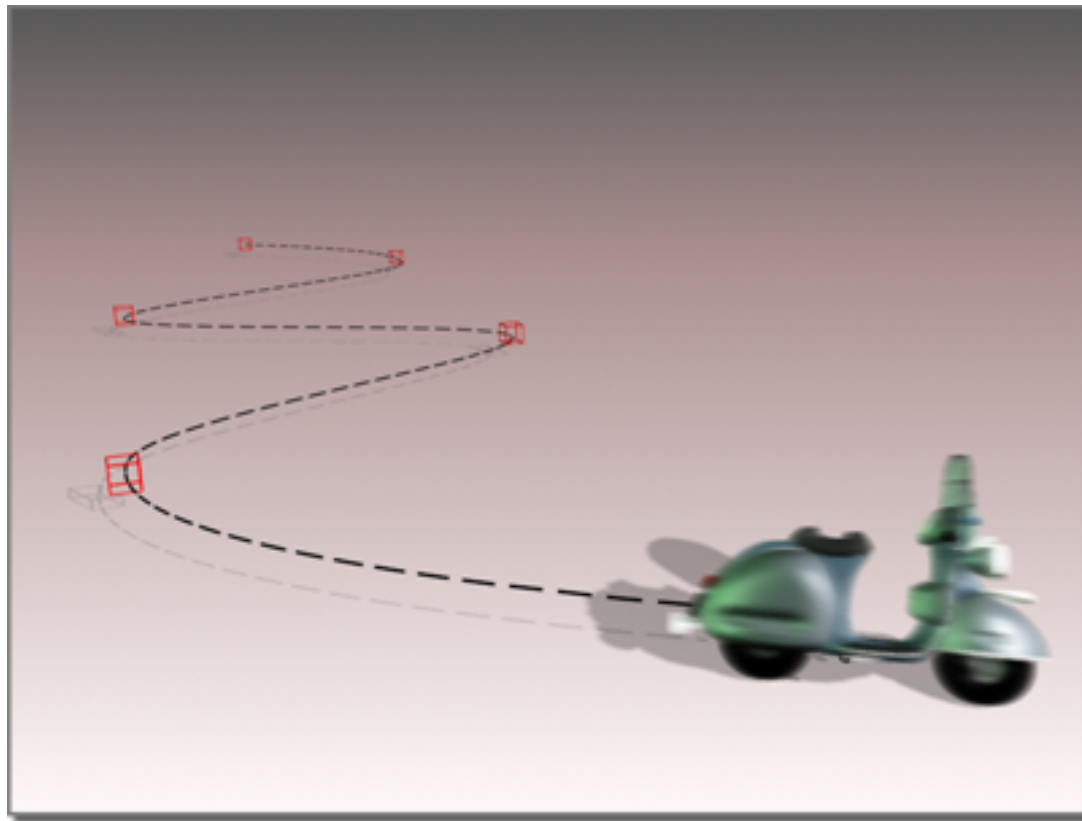
- Applications
 - Design of smooth surfaces in cars, ships, etc.
 - Creating characters or scenes for movies



Parametric Curves

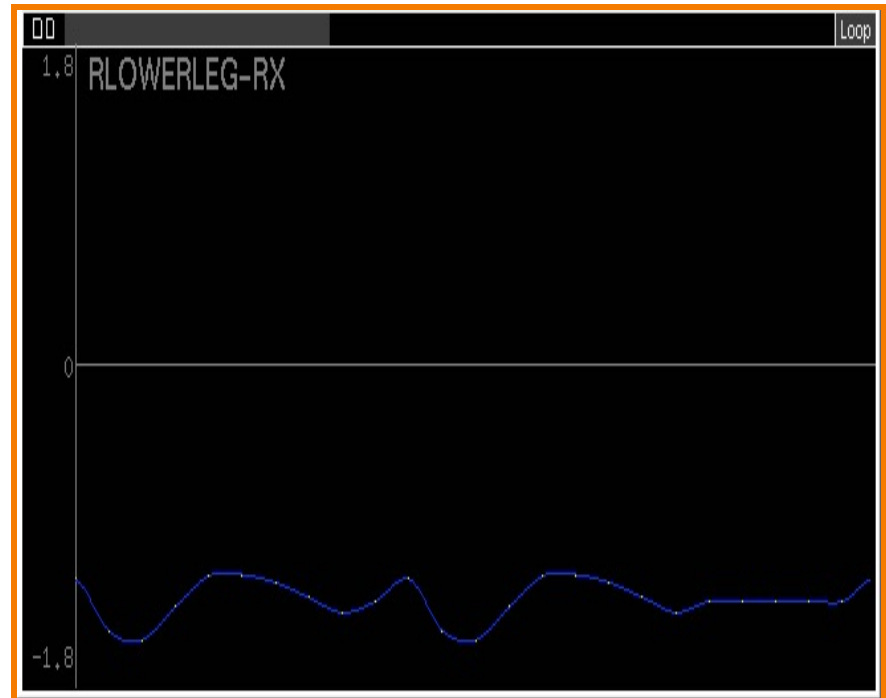
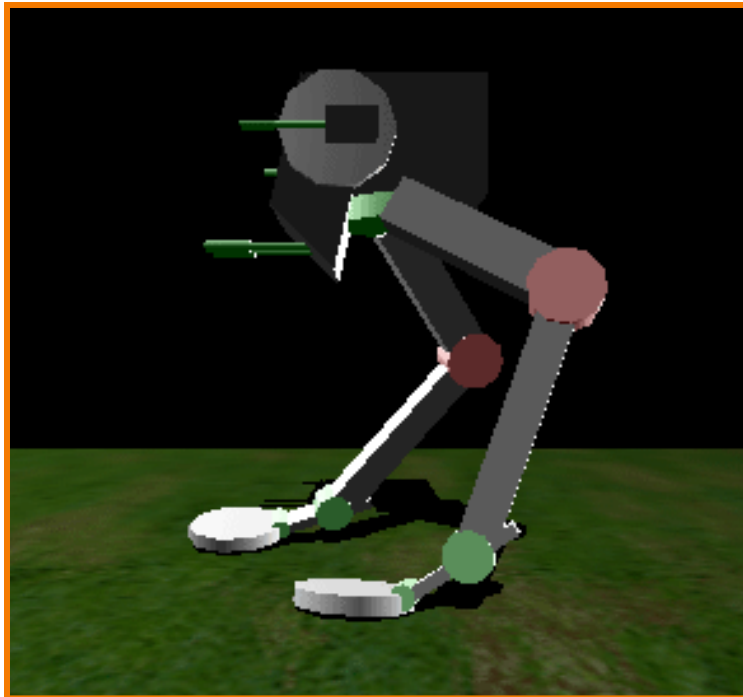


- Applications
 - Defining motion trajectories for objects or cameras



Parametric Curves

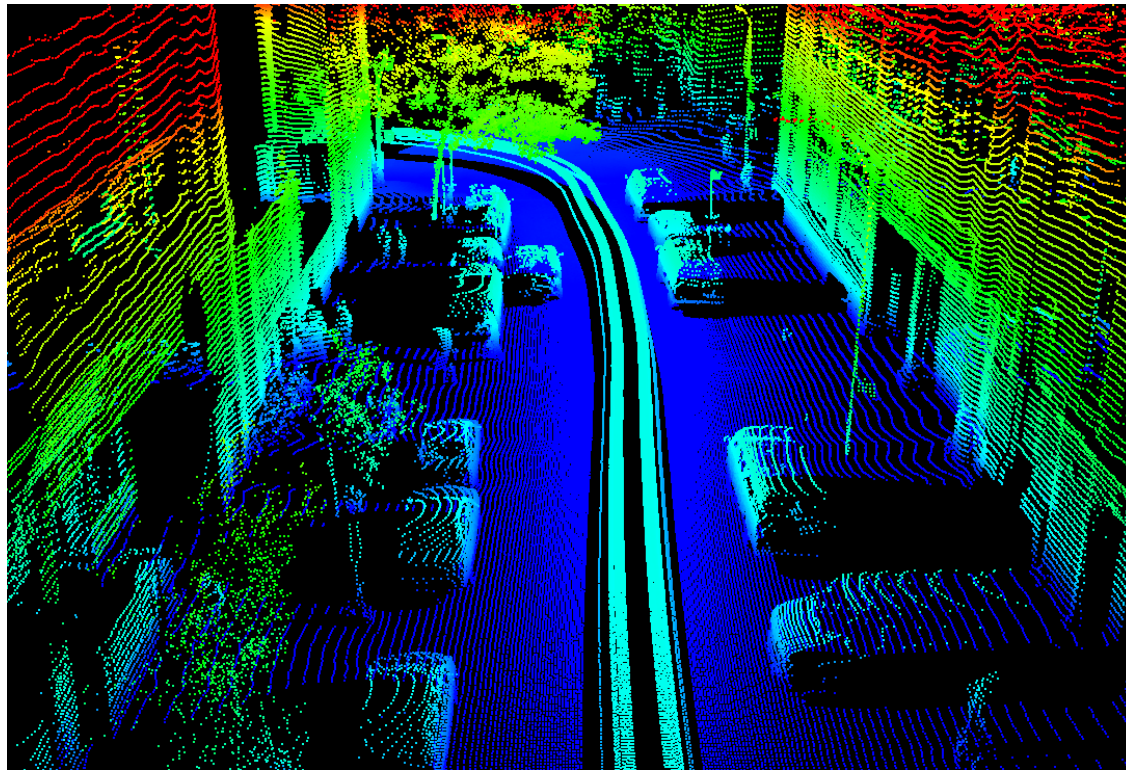
- Applications
 - Defining motion trajectories for objects or cameras
 - Defining smooth interpolations of sparse data



Parametric Curves



- Applications
 - Defining motion trajectories for objects or cameras
 - Defining smooth interpolations of sparse data



Google
Street View

Outline



- Parametric curves
 - Cubic B-Spline
 - Cubic Bézier
- Parametric surfaces
 - Bi-cubic B-Spline
 - Bi-cubic Bézier

Outline



➤ Parametric curves

- Cubic B-Spline
- Cubic Bézier

• Parametric surfaces

- Bi-cubic B-Spline
- Bi-cubic Bézier

Parametric Curves

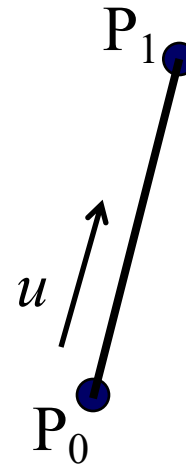
- Defined by parametric functions:
 - $x = f_x(u)$
 - $y = f_y(u)$

- Example: line segment

$$f_x(u) = (1-u)x_0 + ux_1$$

$$f_y(u) = (1-u)y_0 + uy_1$$

$$u \in [0..1]$$



Parametric Curves

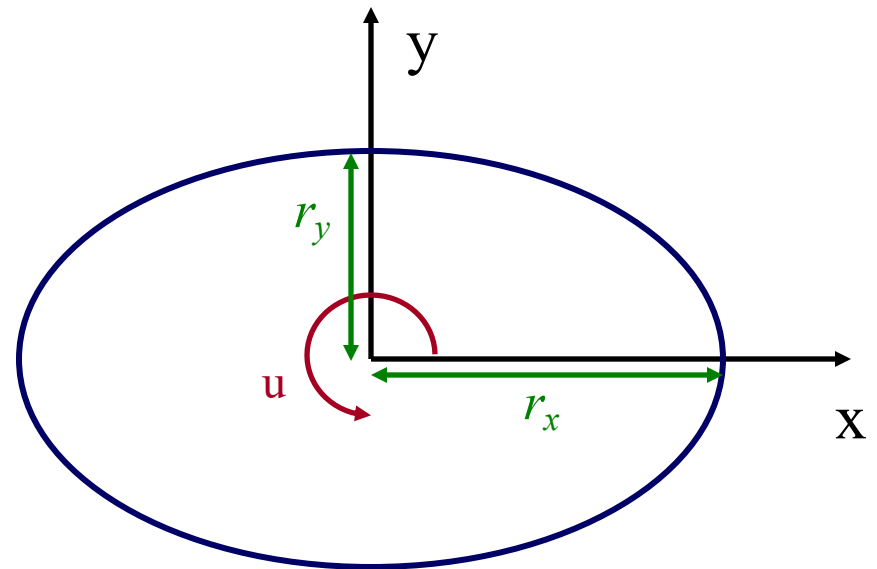
- Defined by parametric functions:
 - $x = f_x(u)$
 - $y = f_y(u)$

- Example: ellipse

$$f_x(u) = r_x \cos(2\pi u)$$

$$f_y(u) = r_y \sin(2\pi u)$$

$$u \in [0..1]$$



Parametric curves



How to easily define arbitrary curves?

$$x = f_x(u)$$

$$y = f_y(u)$$

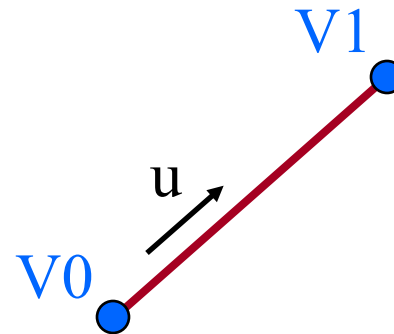


Parametric curves

How to easily define arbitrary curves?

$$x = f_x(u)$$

$$y = f_y(u)$$

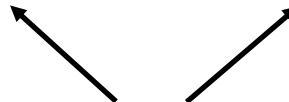


Use functions that “blend” control points

$$x = f_x(u) = V0_x * (1 - u) + V1_x * u$$

$$y = f_y(u) = V0_y * (1 - u) + V1_y * u$$

Simple functions of u



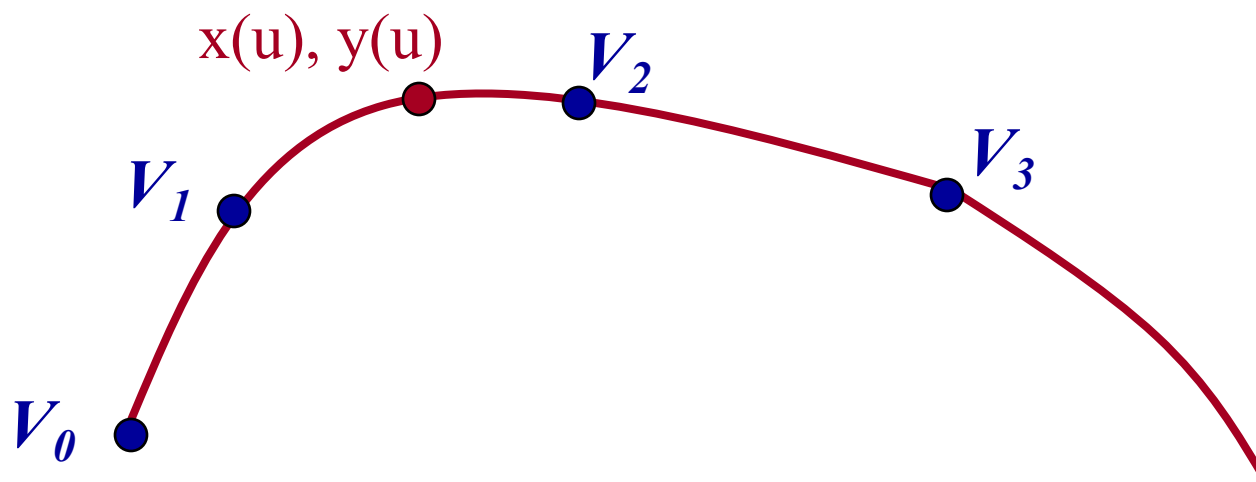
Parametric curves



More generally:

$$x(u) = \sum_{i=0}^n B_i(u) * Vi_x$$

$$y(u) = \sum_{i=0}^n B_i(u) * Vi_y$$



Parametric curves



What $B(u)$ functions should we use?

$$x(u) = \sum_{i=0}^n B_i(u) * Vi_x$$

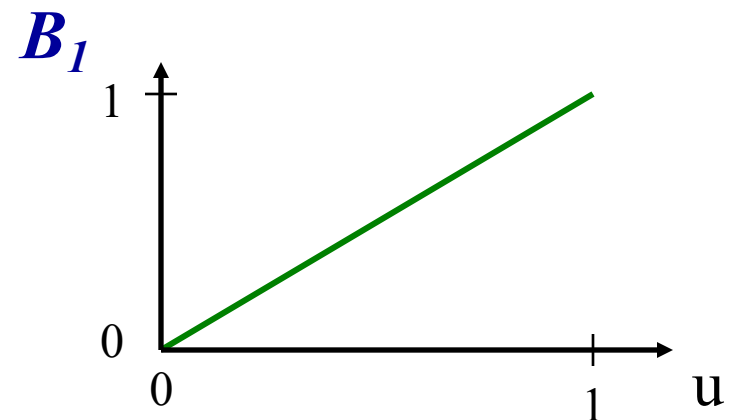
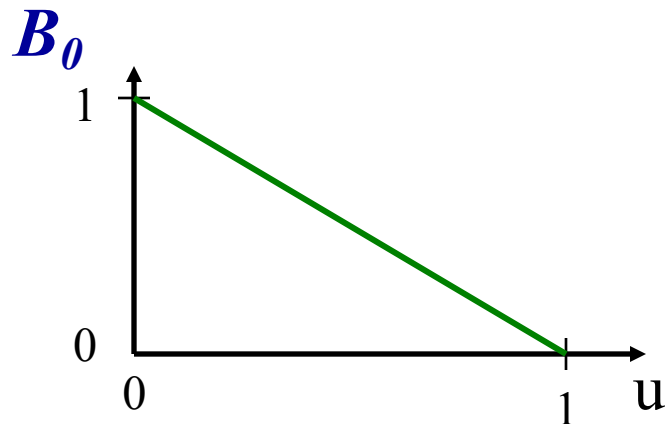
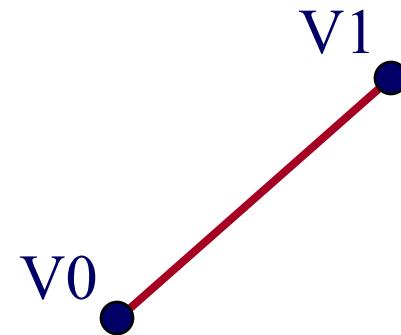
$$y(u) = \sum_{i=0}^n B_i(u) * Vi_y$$

Parametric curves

What $B(u)$ functions should we use?

$$x(u) = \sum_{i=0}^n B_i(u) * Vi_x$$

$$y(u) = \sum_{i=0}^n B_i(u) * Vi_y$$

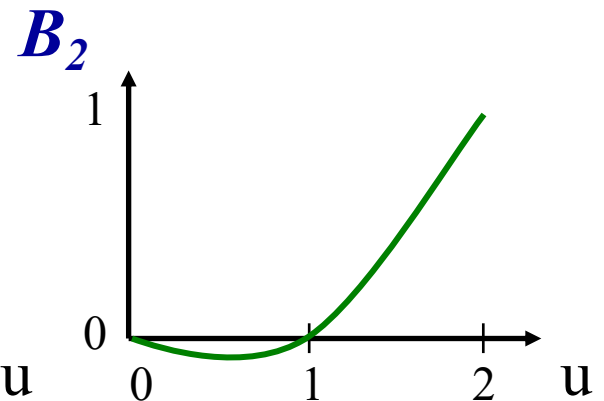
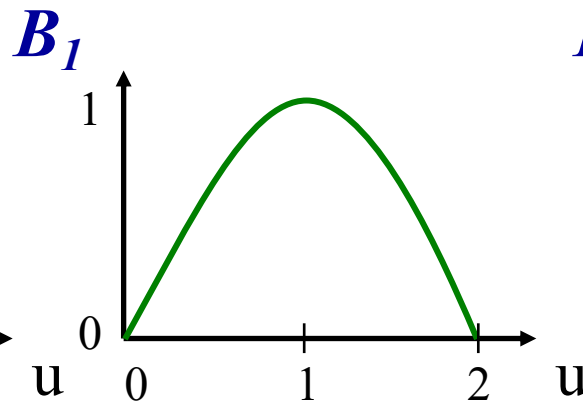
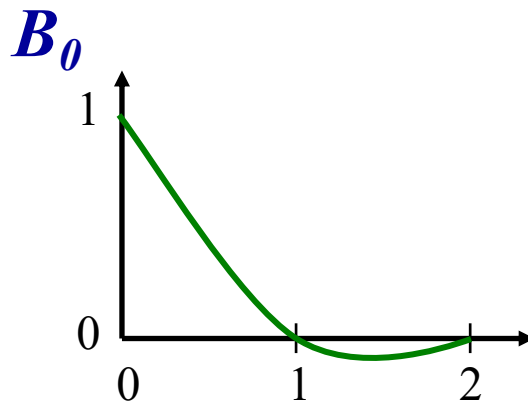
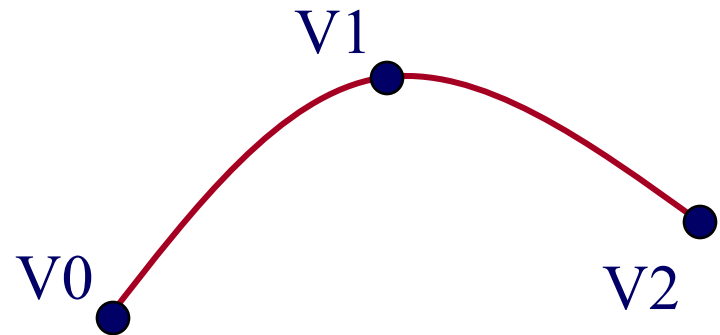


Parametric curves

What $B(u)$ functions should we use?

$$x(u) = \sum_{i=0}^n B_i(u) * Vi_x$$

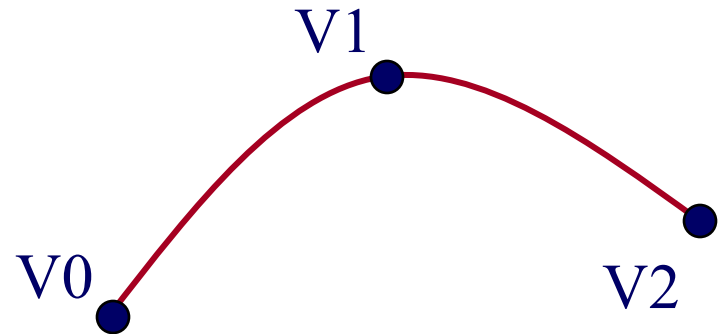
$$y(u) = \sum_{i=0}^n B_i(u) * Vi_y$$



Parametric Polynomial Curves

- Polynomial blending functions:

$$B_i(u) = \sum_{j=0}^m a_j u^j$$

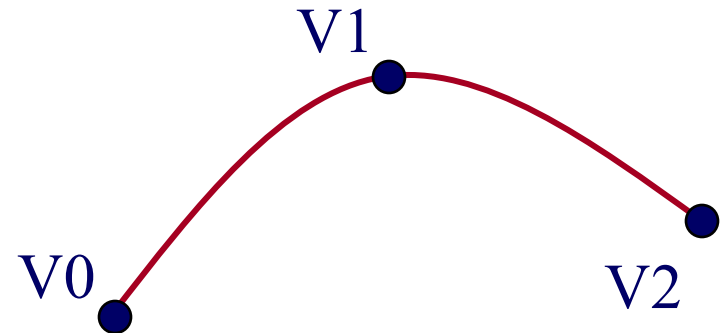


- Advantages of polynomials
 - Easy to compute
 - Infinitely continuous
 - Easy to derive curve properties

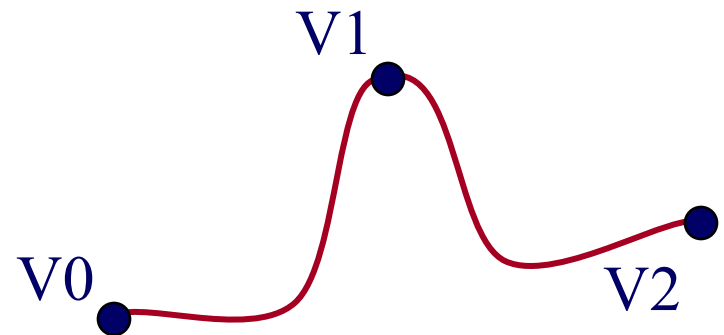
Parametric Polynomial Curves

- Polynomial blending functions:

$$B_i(u) = \sum_{j=0}^m a_j u^j$$



- What degree polynomial?
 - Easy to compute
 - Easy to control
 - Expressive



Piecewise Parametric Polynomial Curves



- **Splines:**

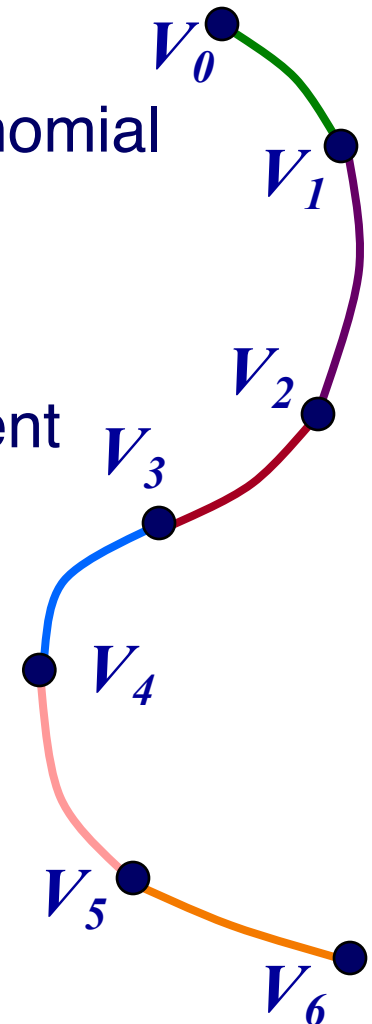
- Split curve into segments
- Each segment defined by low-order polynomial blending subset of control vertices

- **Motivation:**

- Same blending functions for every segment
- Prove properties from blending functions
- Provides **local control** & efficiency

- **Challenges**

- How choose blending functions?
- How determine properties?



Cubic Splines



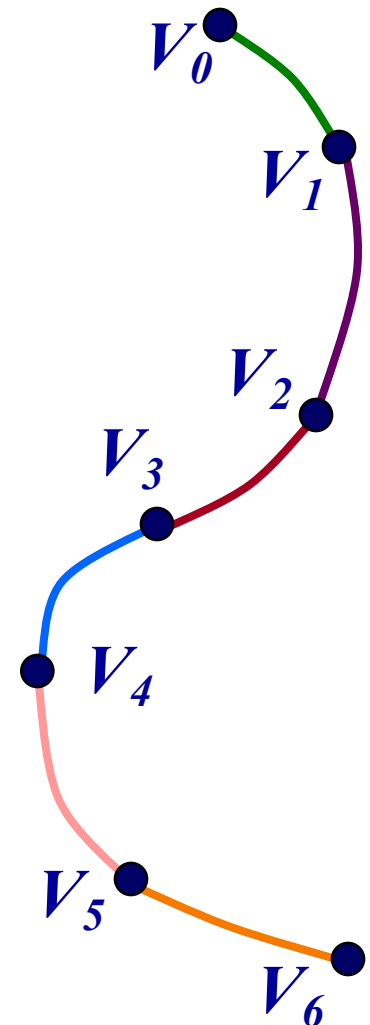
- Some properties we might like to have:

- Local control
- Continuity
- Interpolation?
- Convex hull?

$$B_i(u) = \sum_{j=0}^m a_j u^j$$

Blending functions determine properties

Properties determine blending functions



Outline

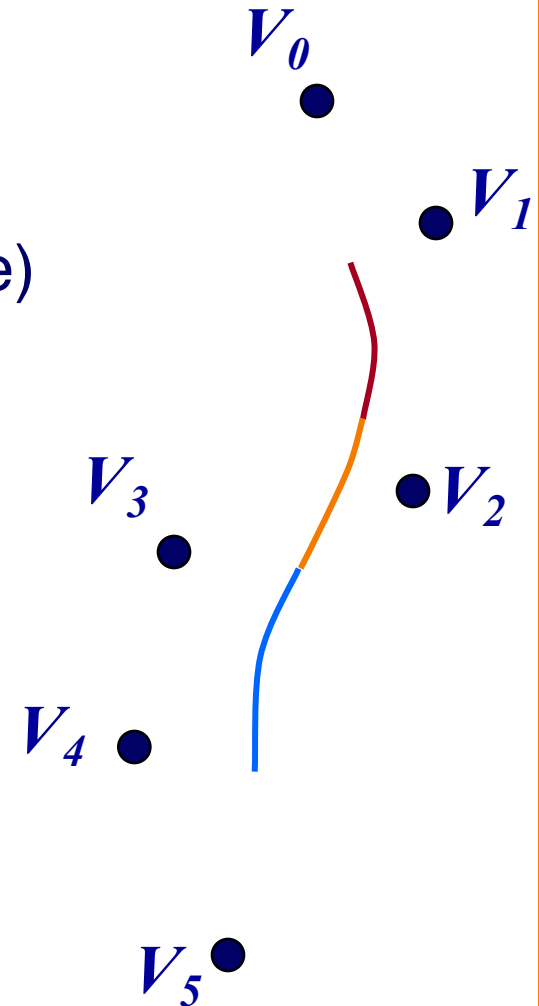


- Parametric curves
 - Cubic B-Spline
 - Cubic Bézier
- Parametric surfaces
 - Bi-cubic B-Spline
 - Bi-cubic Bézier

Cubic B-Splines



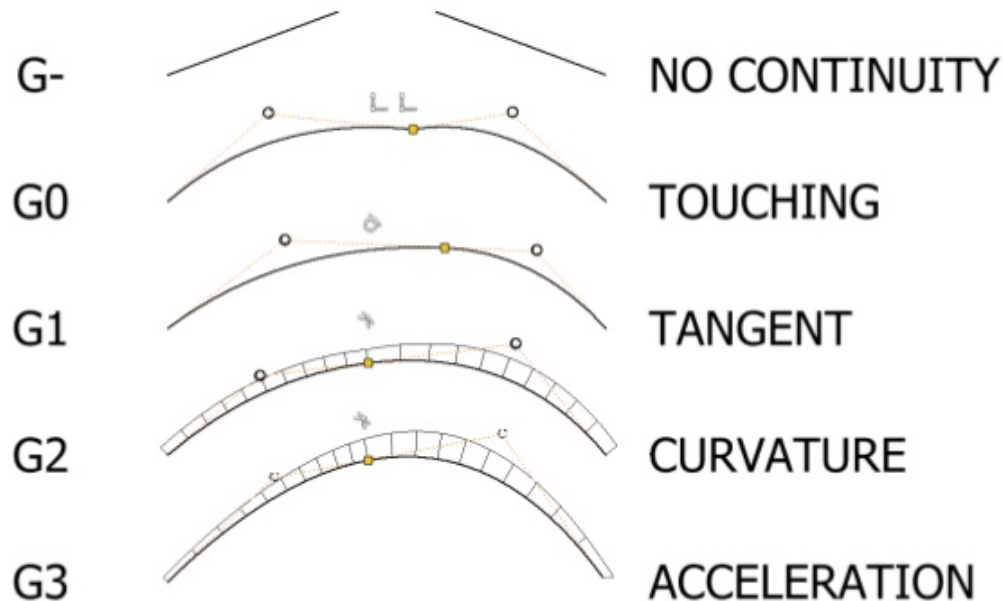
- Properties:
 - Local control
 - C^2 continuity at joints
(infinitely continuous within each piece)
 - Approximating
 - Convex hull



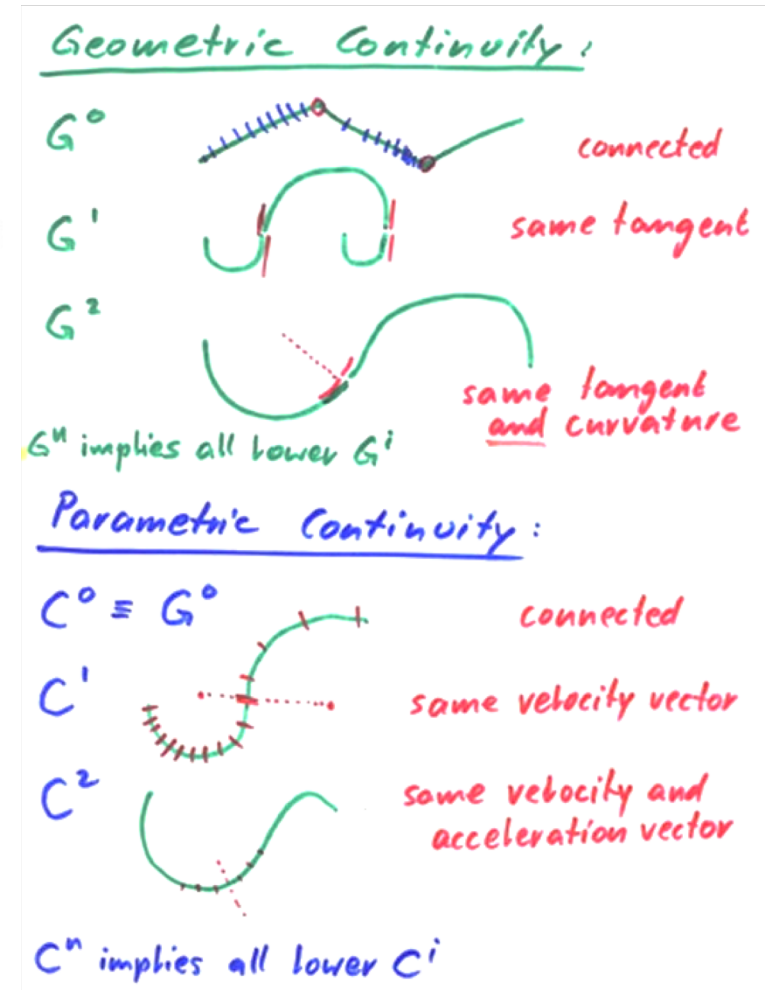
Cubic B-Splines



Notes on continuity



[from Robert Duvall]



[from Carlo Séquin]

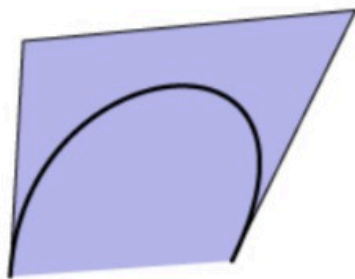
Cubic B-Splines



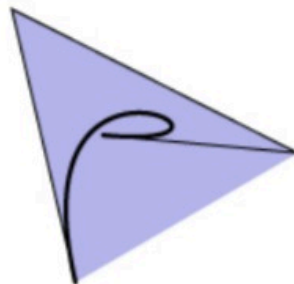
Notes on local control:



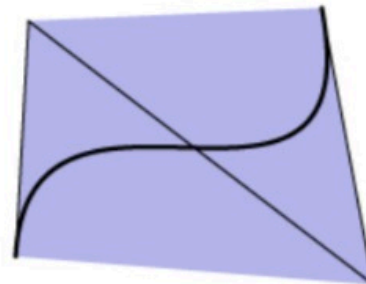
and convex hull property:



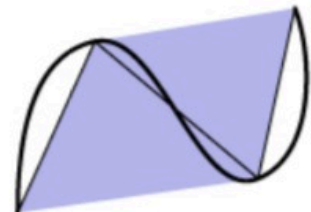
YES



YES



YES



NO

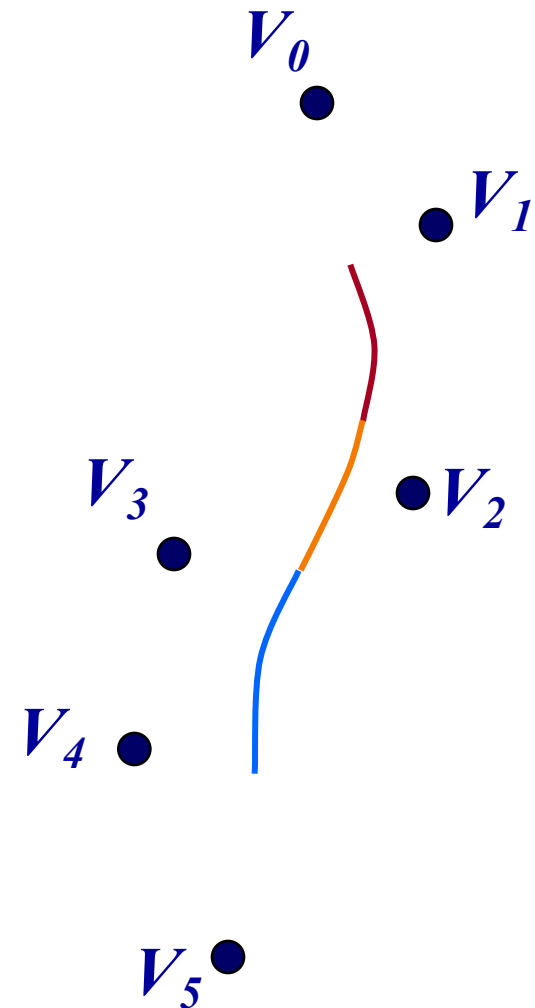
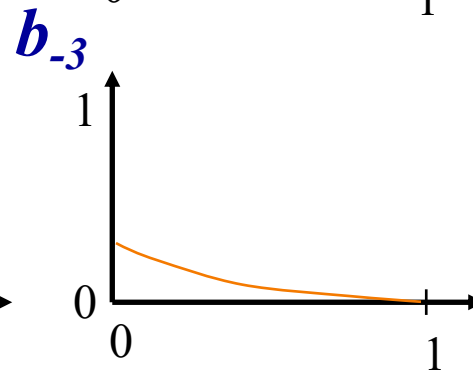
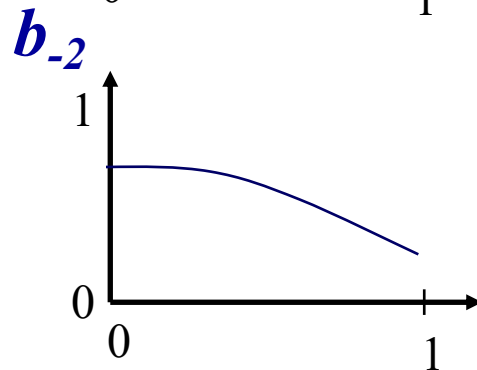
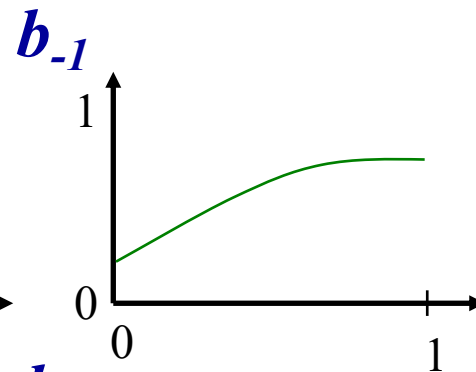
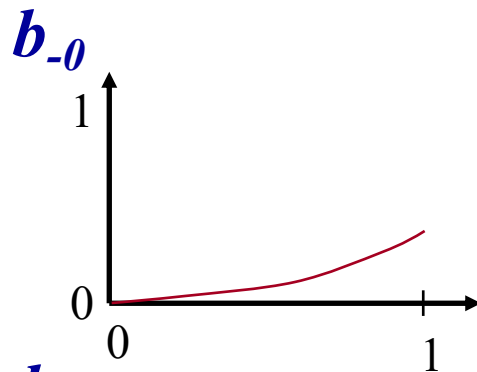
[from Steve Marschner]

Cubic B-Spline Blending Functions



Blending functions:

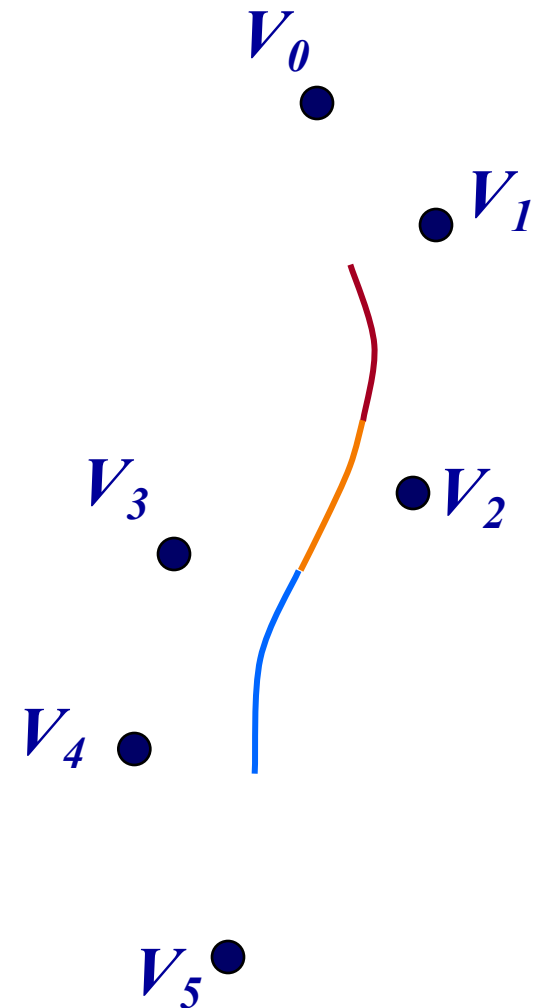
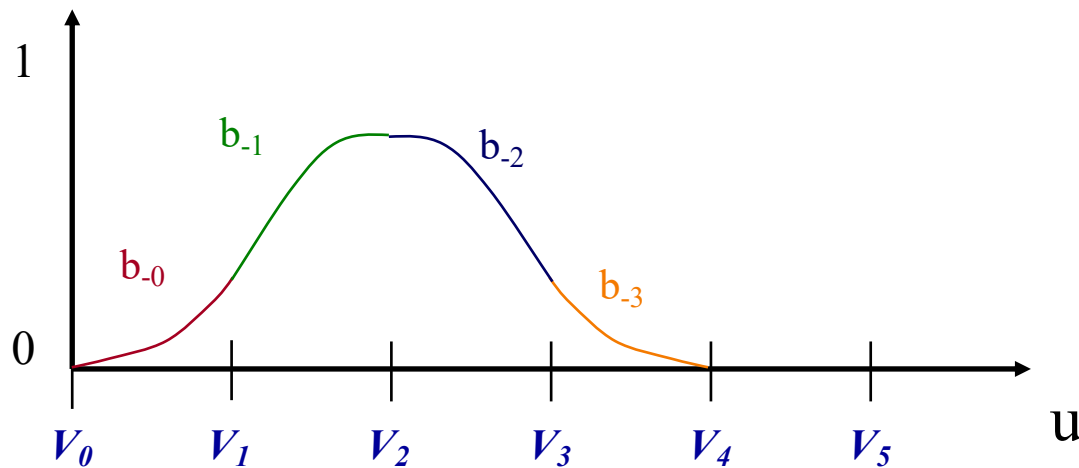
$$B_i(u) = \sum_{j=0}^m a_j u^j$$



Cubic B-Spline Blending Functions



- How derive blending functions?
 - Cubic polynomials
 - Local control
 - C^2 continuity
 - Convex hull



Cubic B-Spline Blending Functions



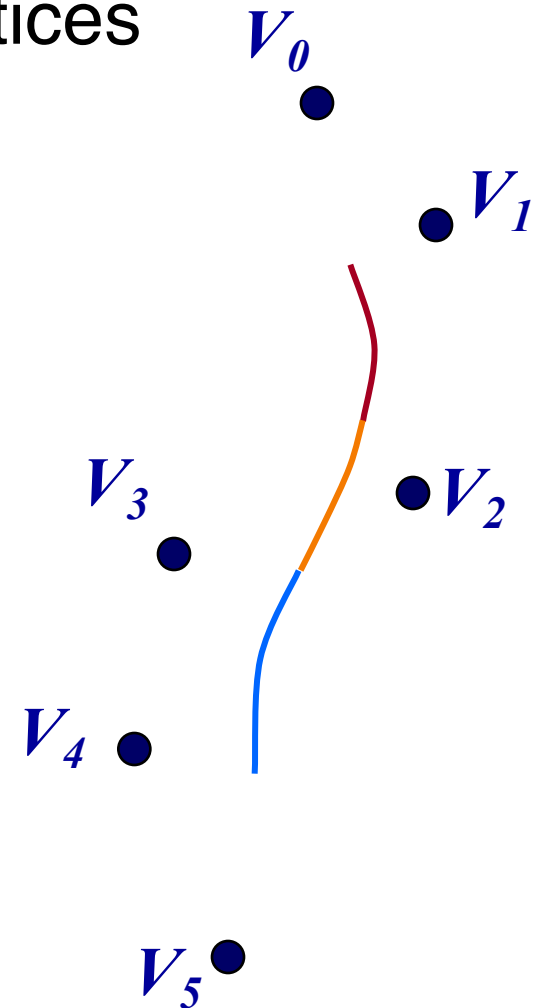
- Four cubic polynomials for four vertices
 - 16 variables (degrees of freedom)
 - Variables are a_i, b_i, c_i, d_i for four blending functions

$$b_{-0}(u) = a_0 u^3 + b_0 u^2 + c_0 u^1 + d_0$$

$$b_{-1}(u) = a_1 u^3 + b_1 u^2 + c_1 u^1 + d_1$$

$$b_{-2}(u) = a_2 u^3 + b_2 u^2 + c_2 u^1 + d_2$$

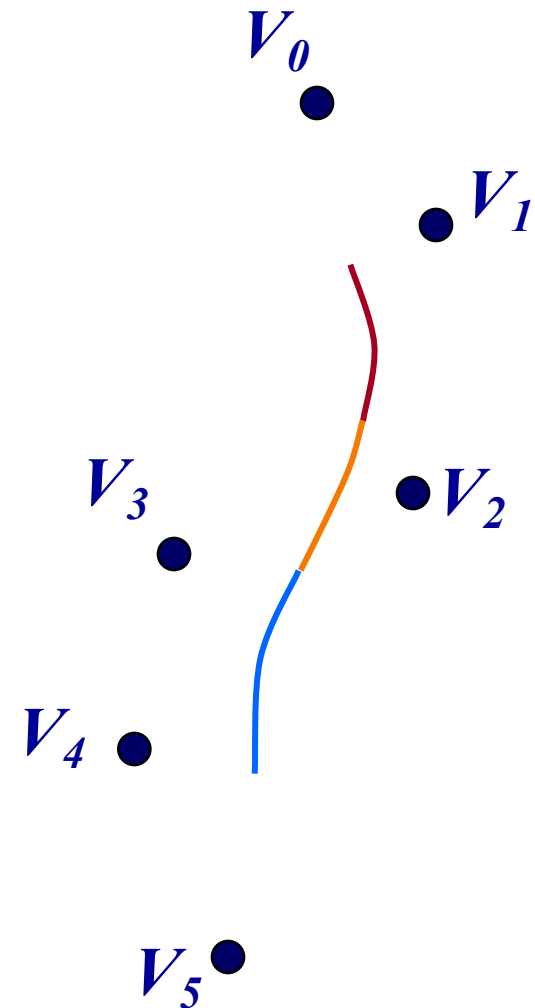
$$b_{-3}(u) = a_3 u^3 + b_3 u^2 + c_3 u^1 + d_3$$



Cubic B-Spline Blending Functions



- C^2 continuity implies 15 constraints
 - Position of two curves same
 - Derivative of two curves same
 - Second derivatives same



Cubic B-Spline Blending Functions



Fifteen continuity constraints:

$0 = b_{-0}(0)$	$0 = b_{-0}'(0)$	$0 = b_{-0}''(0)$
$b_{-0}(1) = b_{-1}(0)$	$b_{-0}'(1) = b_{-1}'(0)$	$b_{-0}''(1) = b_{-1}''(0)$
$b_{-1}(1) = b_{-2}(0)$	$b_{-1}'(1) = b_{-2}'(0)$	$b_{-1}''(1) = b_{-2}''(0)$
$b_{-2}(1) = b_{-3}(0)$	$b_{-2}'(1) = b_{-3}'(0)$	$b_{-2}''(1) = b_{-3}''(0)$
$b_{-3}(1) = 0$	$b_{-3}'(1) = 0$	$b_{-3}''(1) = 0$

One more convenient constraint:

$$b_{-0}(0) + b_{-1}(0) + b_{-2}(0) + b_{-3}(0) = 1$$

Cubic B-Spline Blending Functions



- Solving the system of equations yields:

$$b_{-3}(u) = -\frac{1}{6}u^3 + \frac{1}{2}u^2 - \frac{1}{2}u + \frac{1}{6}$$

$$b_{-2}(u) = \frac{1}{2}u^3 - u^2 + \frac{2}{3}$$

$$b_{-1}(u) = -\frac{1}{2}u^3 + \frac{1}{2}u^2 + \frac{1}{2}u + \frac{1}{6}$$

$$b_{-0}(u) = \frac{1}{6}u^3$$

Cubic B-Spline Blending Functions



- In matrix form:

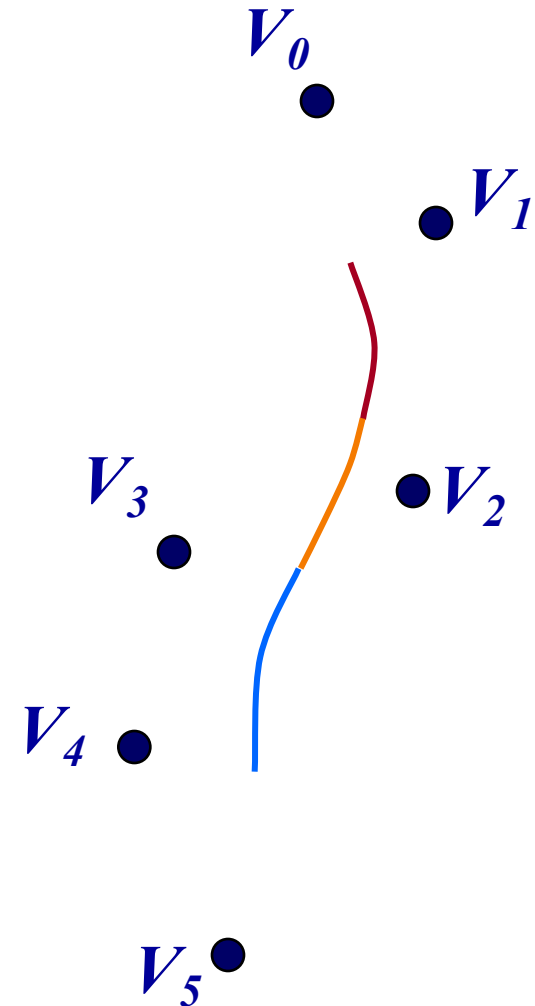
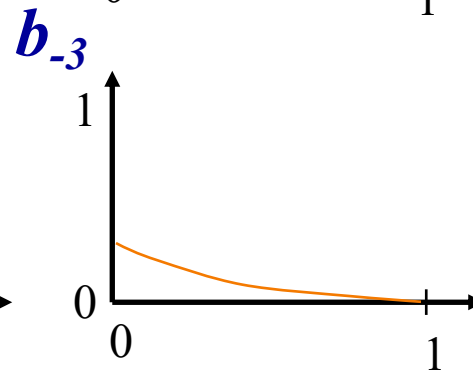
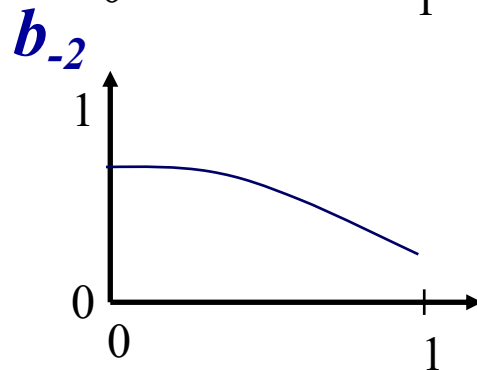
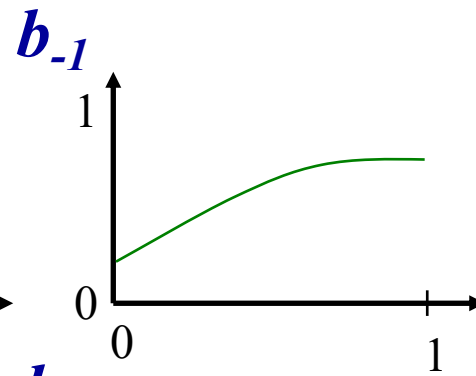
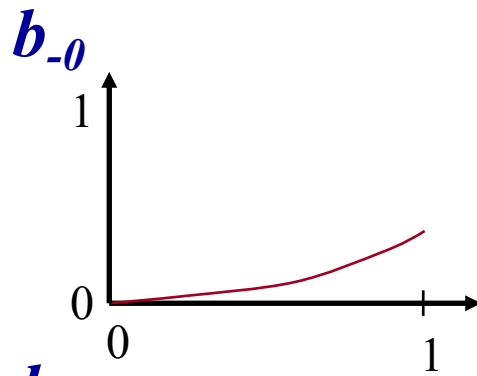
$$Q(u) = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \frac{1}{6} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix} \begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix}$$

Cubic B-Spline Blending Functions



In plot form:

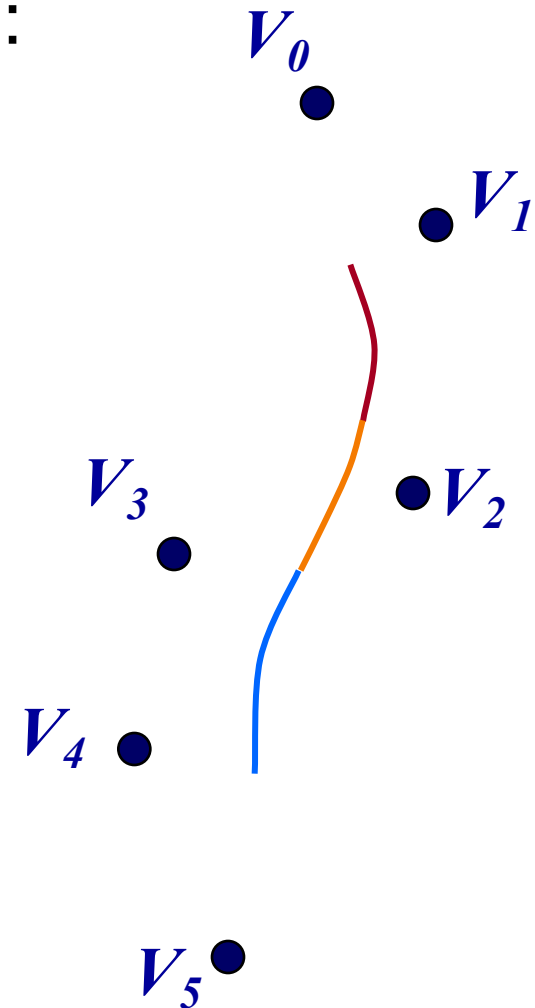
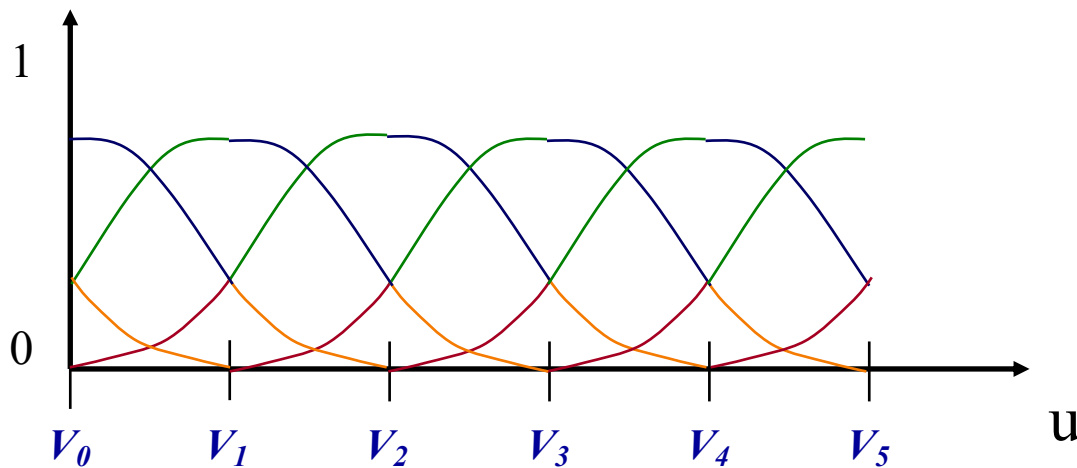
$$B_i(u) = \sum_{j=0}^m a_j u^j$$



Cubic B-Spline Blending Functions



- Blending functions imply properties:
 - Local control
 - Approximating
 - C^2 continuity
 - Convex hull



Try online at <http://bl.ocks.org/mbostock/4342190>

Outline

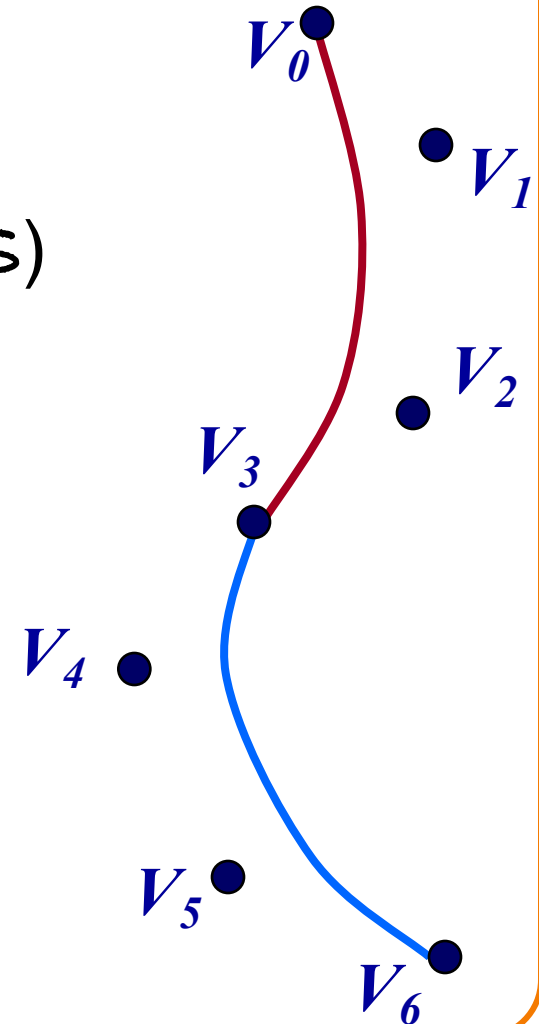


- Parametric curves
 - Cubic B-Spline
 - Cubic Bézier
- Parametric surfaces
 - Bi-cubic B-Spline
 - Bi-cubic Bézier

Bézier Curves

- Developed around 1960 by both
 - Pierre Bézier (Renault)
 - Paul de Casteljau (Citroen)
- Today: graphic design (e.g. *FONTS*)
- Properties:
 - Local control
 - Continuity depends on control points
 - Interpolating (every third for cubic)

Blending functions determine properties

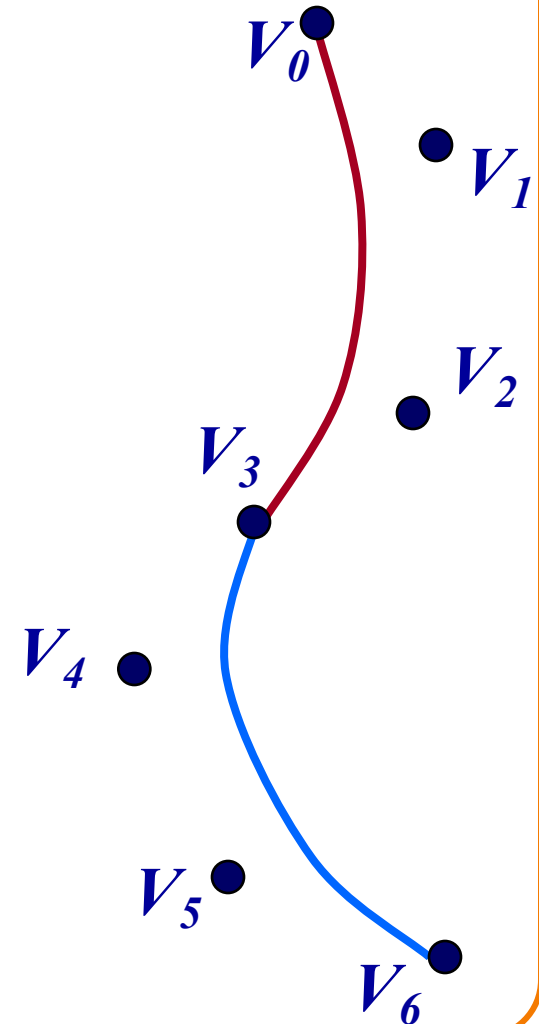
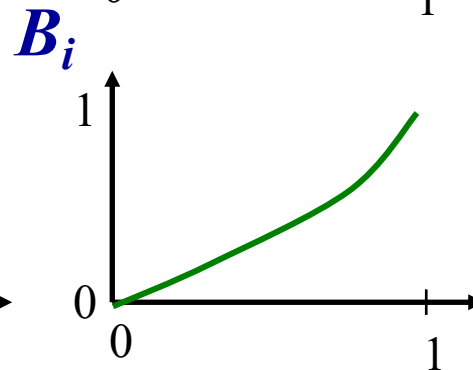
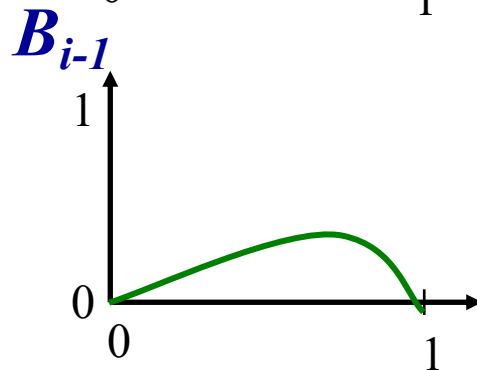
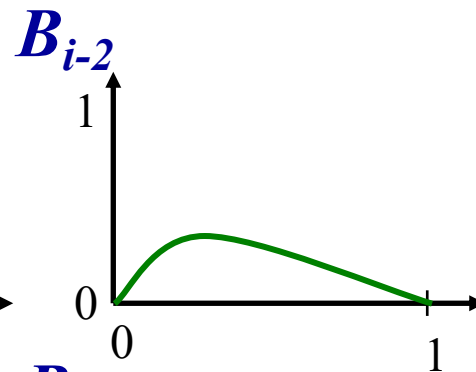
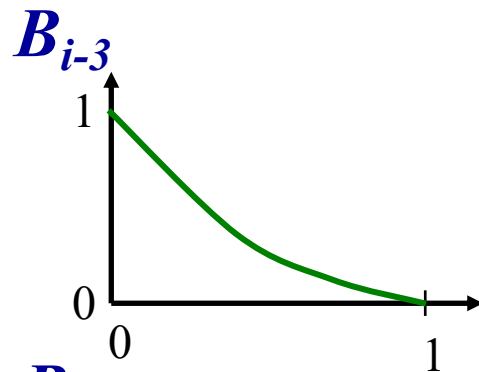


Cubic Bézier Curves



Blending functions:

$$B_i(u) = \sum_{j=0}^m a_j u^j$$



Cubic Bézier Curves



Bézier curves in matrix form:

$$\begin{aligned} Q(u) &= \sum_{i=0}^n V_i \binom{n}{i} u^i (1-u)^{n-i} \\ &= (1-u)^3 V_0 + 3u(1-u)^2 V_1 + 3u^2(1-u) V_2 + u^3 V_3 \\ &= \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix} \end{aligned}$$

$\mathbf{M}_{\text{Bézier}}$

Basic properties of Bézier Curves



- Endpoint interpolation:

$$Q(0) = V_0$$

$$Q(1) = V_n$$

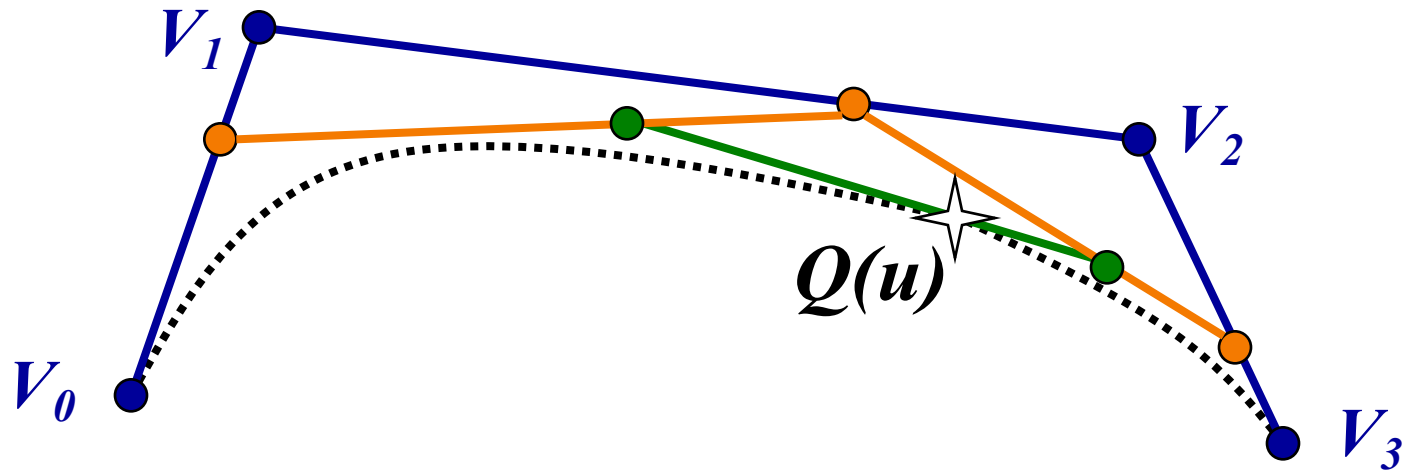
- Convex hull:
 - Curve is contained within convex hull of control polygon

- Symmetry

$$Q(u) \text{ defined by } \{V_0, \dots, V_n\} \equiv Q(1-u) \text{ defined by } \{V_n, \dots, V_0\}$$

Bézier Curves

- Curve $Q(u)$ can also be defined by nested interpolation:



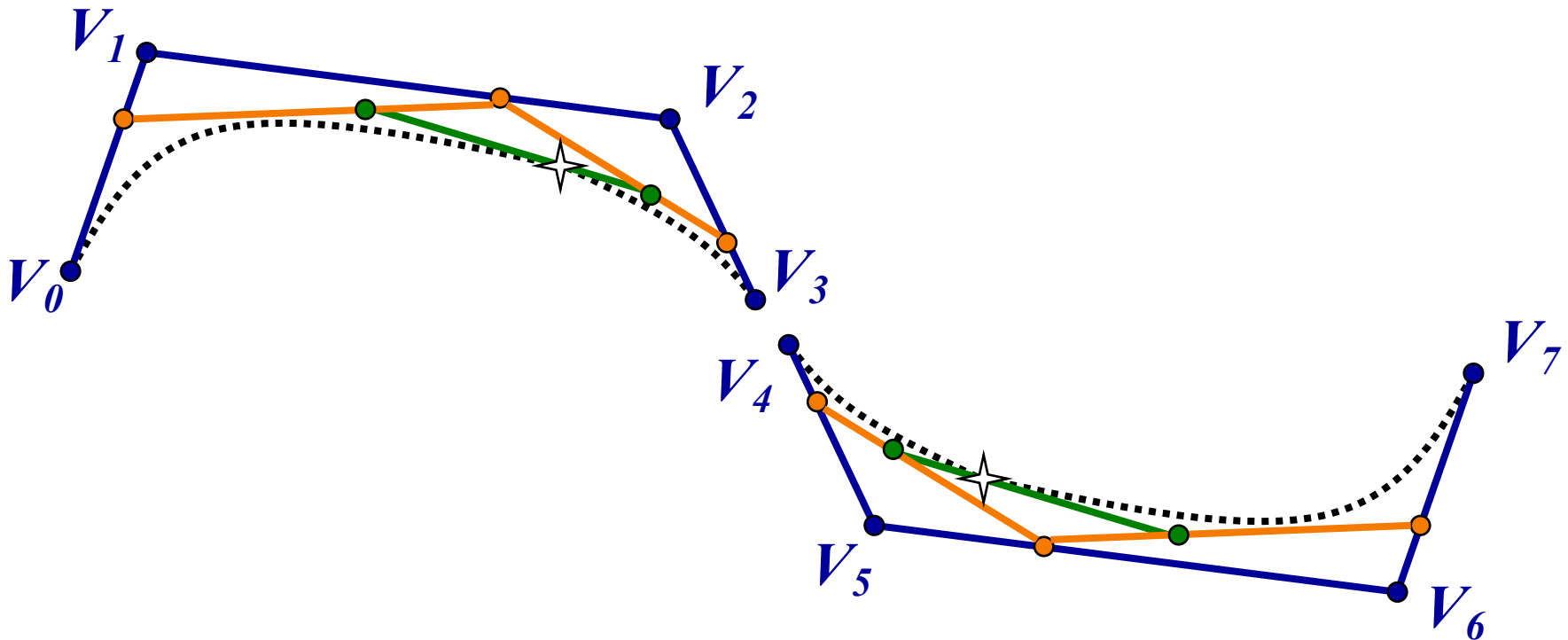
V_i are control points

$\{V_0, V_1, \dots, V_n\}$ is control polygon

Enforcing Bézier Curve Continuity



- C^0 : $V_3 = V_4$
- C^1 : $V_5 - V_4 = V_3 - V_2$
- C^2 : $V_6 - 2V_5 + V_4 = V_3 - 2V_2 + V_1$



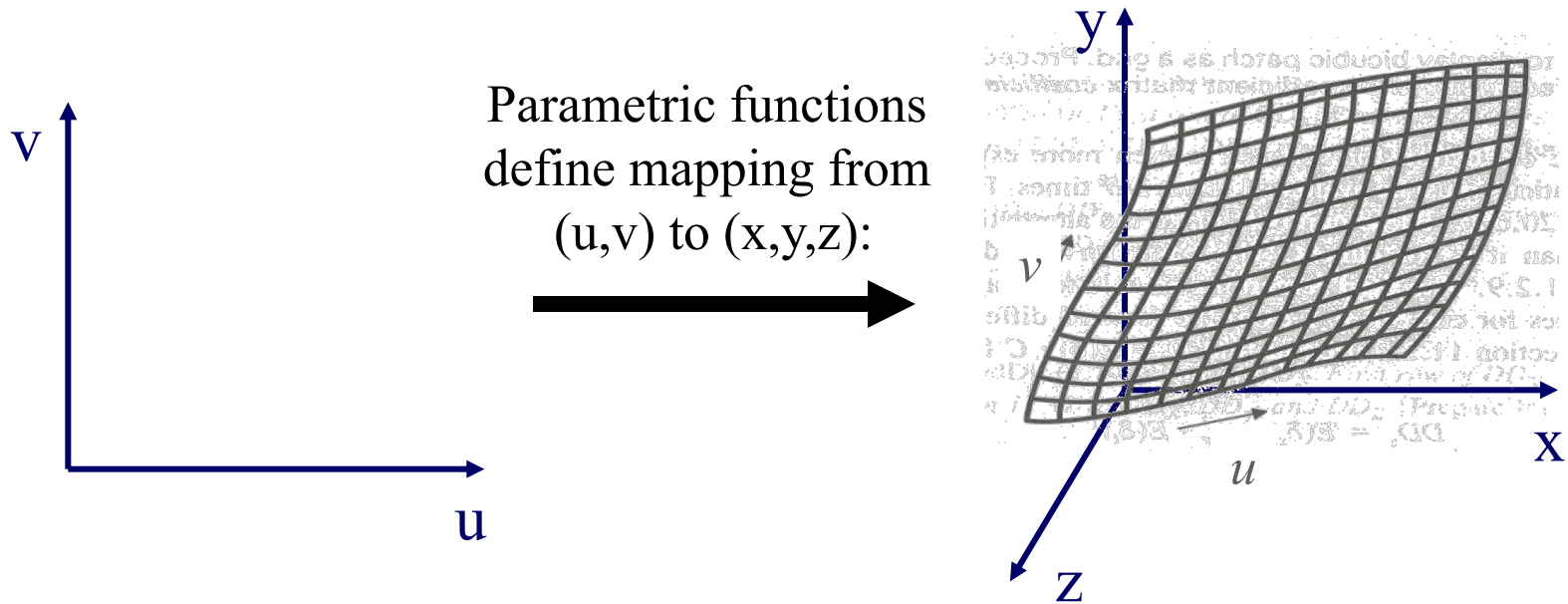
Outline



- Parametric curves
 - Cubic B-Spline
 - Cubic Bézier
- Parametric surfaces
 - Bi-cubic B-Spline
 - Bi-cubic Bézier

Parametric Surfaces

- Defined by parametric functions:
 - $x = f_x(u,v)$
 - $y = f_y(u,v)$
 - $z = f_z(u,v)$

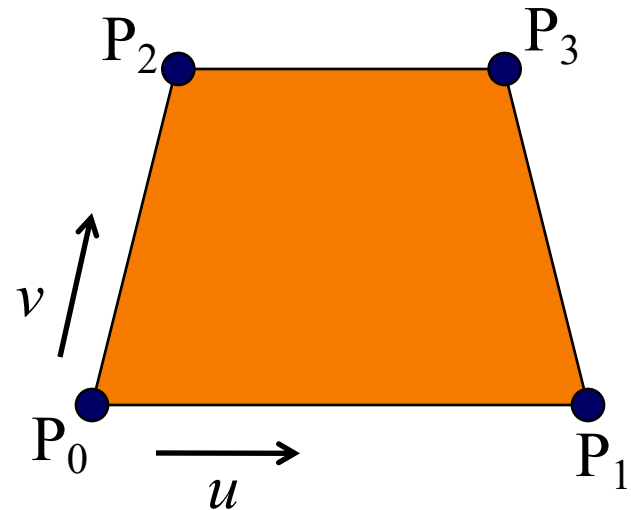


Parametric Surfaces

- Defined by parametric functions:

- $x = f_x(u,v)$
- $y = f_y(u,v)$
- $z = f_z(u,v)$

- Example: quadrilateral



$$f_x(u,v) = (1-v)((1-u)x_0 + ux_1) + v((1-u)x_2 + ux_3)$$

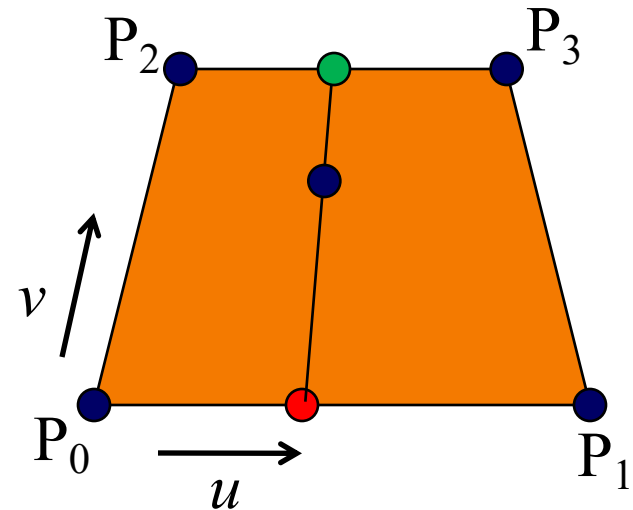
$$f_y(u,v) = (1-v)((1-u)y_0 + uy_1) + v((1-u)y_2 + uy_3)$$

$$f_z(u,v) = (1-v)((1-u)z_0 + uz_1) + v((1-u)z_2 + uz_3)$$

Parametric Surfaces

- Defined by parametric functions:

- $x = f_x(u, v)$
- $y = f_y(u, v)$
- $z = f_z(u, v)$



- Example: quadrilateral

$$f_x(u, v) = (1-v)((1-u)x_0 + ux_1) + v((1-u)x_2 + ux_3)$$

$$f_y(u, v) = (1-v)((1-u)y_0 + uy_1) + v((1-u)y_2 + uy_3)$$

$$f_z(u, v) = (1-v)((1-u)z_0 + uz_1) + v((1-u)z_2 + uz_3)$$

Parametric Surfaces

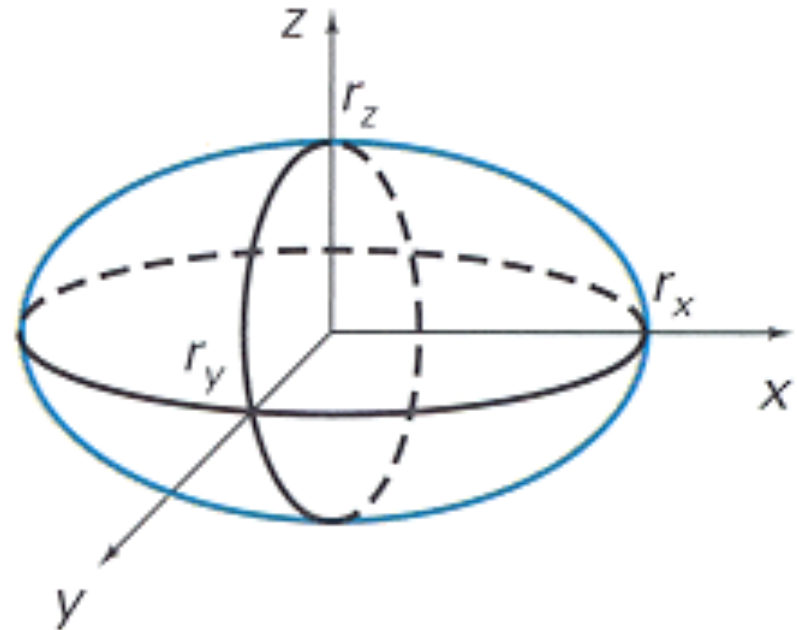
- Defined by parametric functions:
 - $x = f_x(u,v)$
 - $y = f_y(u,v)$
 - $z = f_z(u,v)$

- Example: ellipsoid

$$f_x(u,v) = r_x \cos v \cos u$$

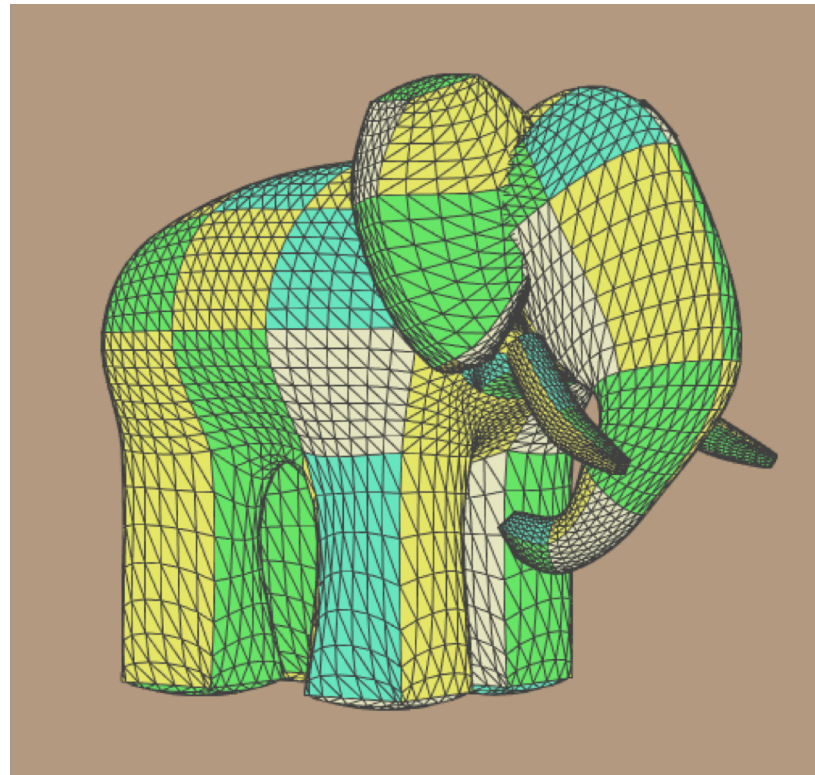
$$f_y(u,v) = r_y \cos v \sin u$$

$$f_z(u,v) = r_z \sin v$$



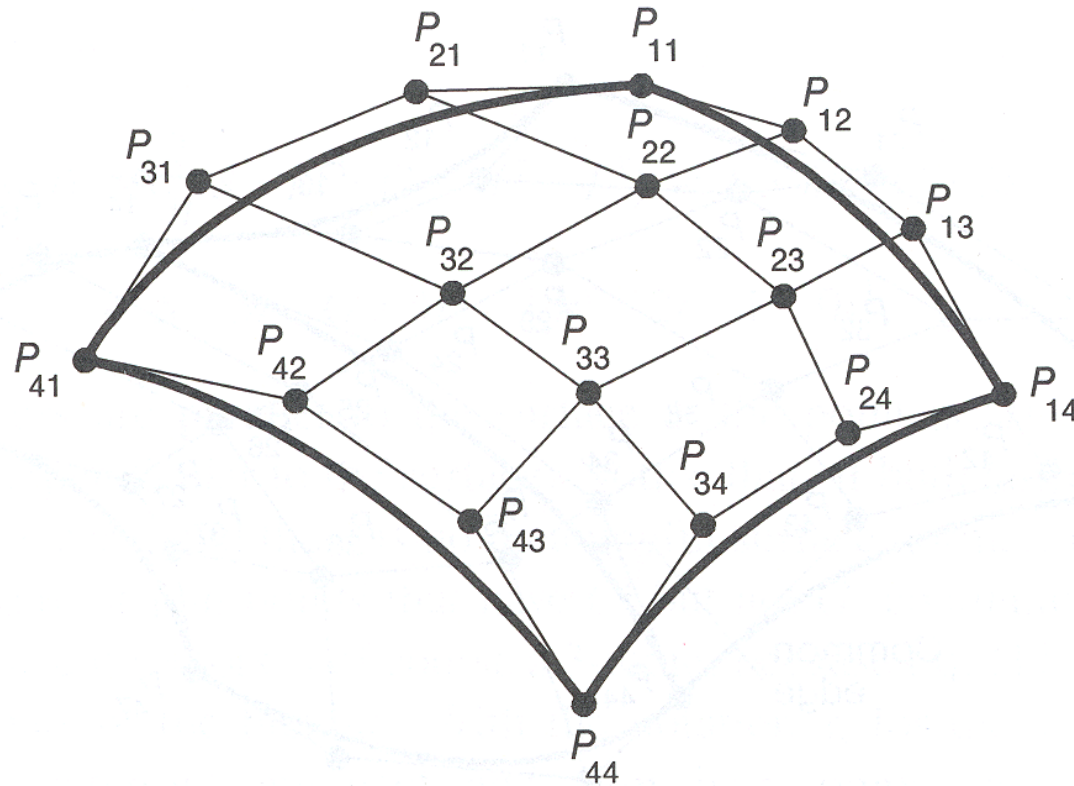
Parametric Surfaces

To model arbitrary shapes, surface is partitioned into parametric patches



Parametric Patches

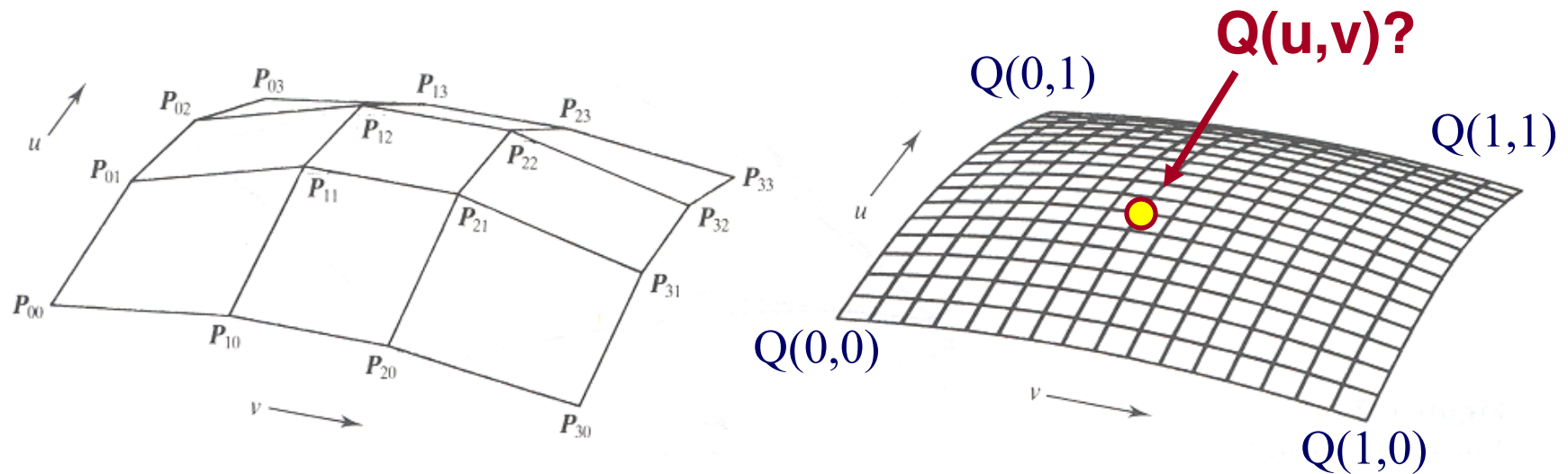
- Each patch is defined by blending control points



Same ideas as parametric curves!

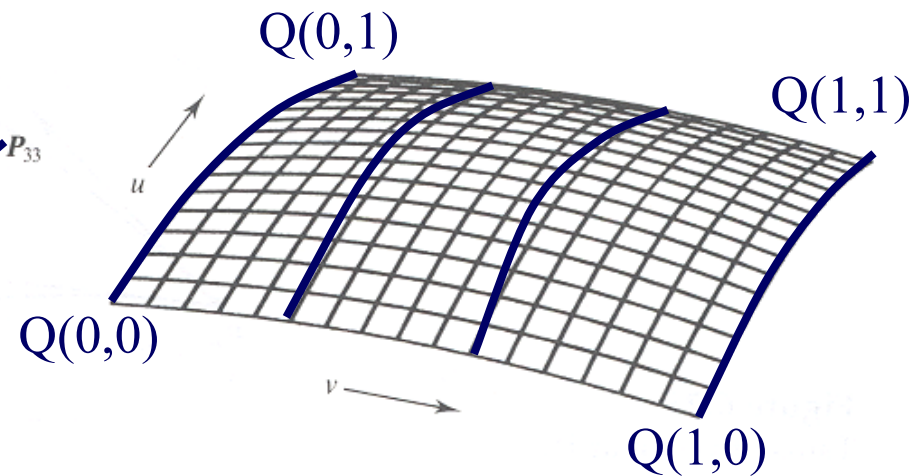
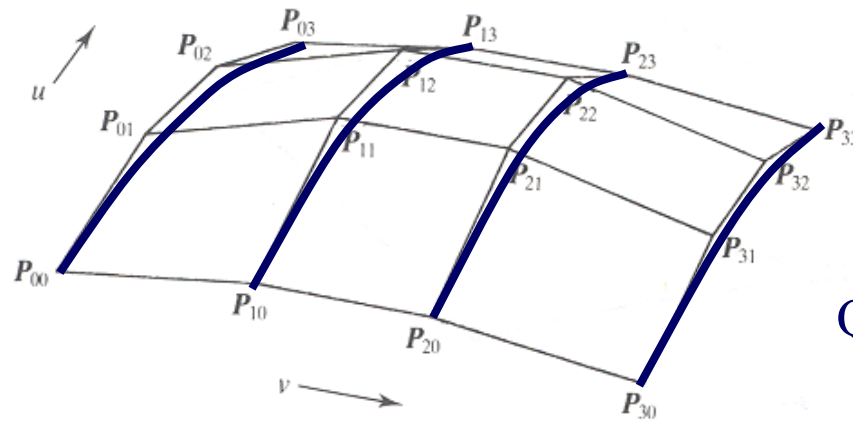
Parametric Patches

- Point $Q(u,v)$ on the patch is the tensor product of parametric curves defined by the control points



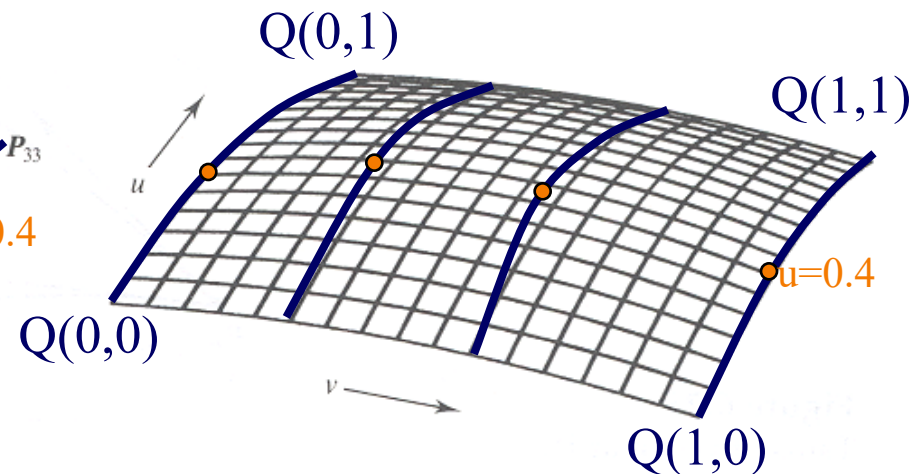
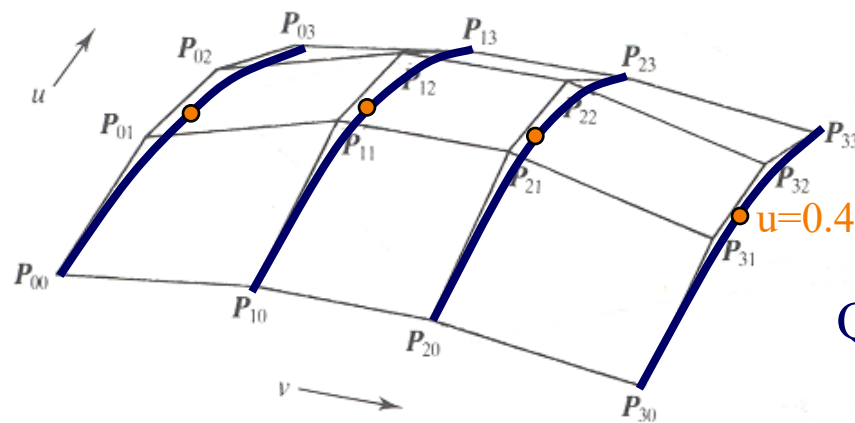
Parametric Patches

- Point $Q(u,v)$ on the patch is the tensor product of parametric curves defined by the control points



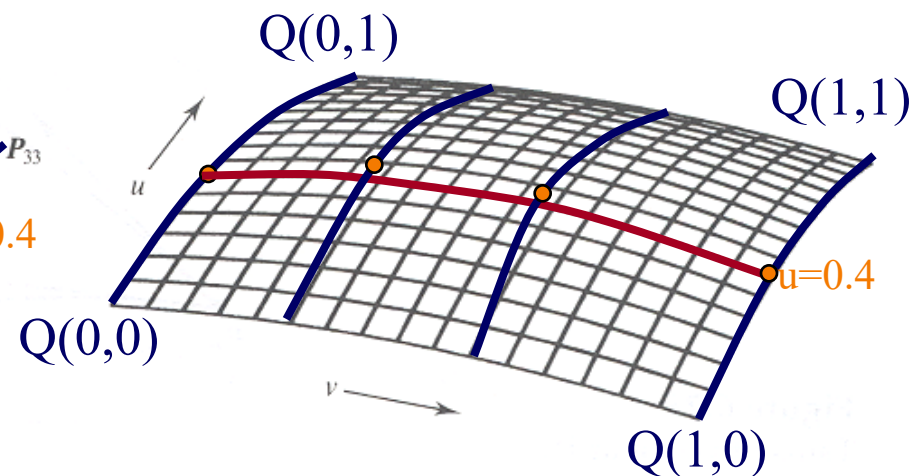
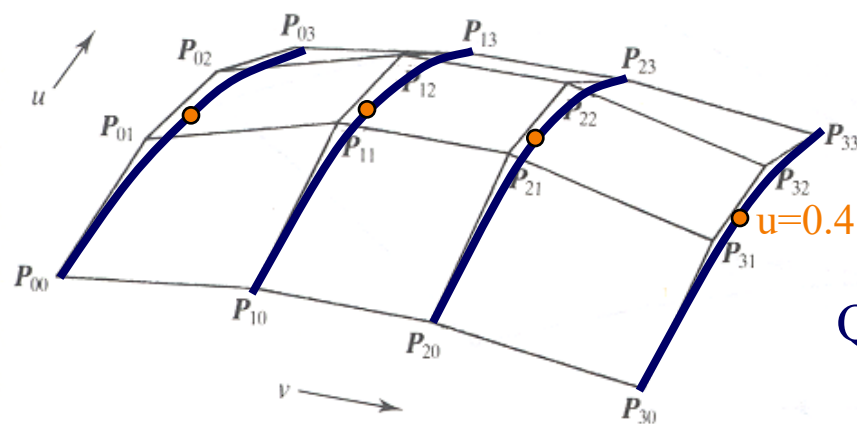
Parametric Patches

- Point $Q(u,v)$ on the patch is the tensor product of parametric curves defined by the control points



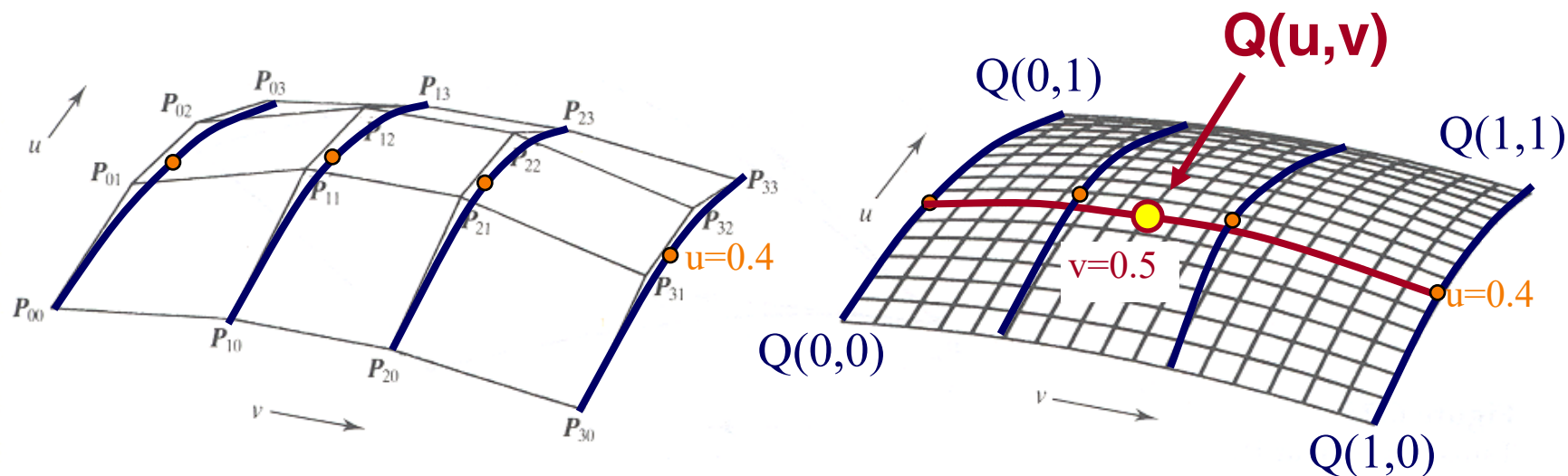
Parametric Patches

- Point $Q(u,v)$ on the patch is the tensor product of parametric curves defined by the control points



Parametric Patches

- Point $Q(u,v)$ on the patch is the tensor product of parametric curves defined by the control points





Parametric Bicubic Patches

Point $Q(u,v)$ on any patch is defined by combining control points with polynomial blending functions:

$$Q(u,v) = \mathbf{U} \mathbf{M} \begin{bmatrix} P_{1,1} & P_{1,2} & P_{1,3} & P_{1,4} \\ P_{2,1} & P_{2,2} & P_{2,3} & P_{2,4} \\ P_{3,1} & P_{3,2} & P_{3,3} & P_{3,4} \\ P_{4,1} & P_{4,2} & P_{4,3} & P_{4,4} \end{bmatrix} \mathbf{M}^T \mathbf{V}^T$$

$$\mathbf{U} = [u^3 \quad u^2 \quad u \quad 1] \quad \mathbf{V} = [v^3 \quad v^2 \quad v \quad 1]$$

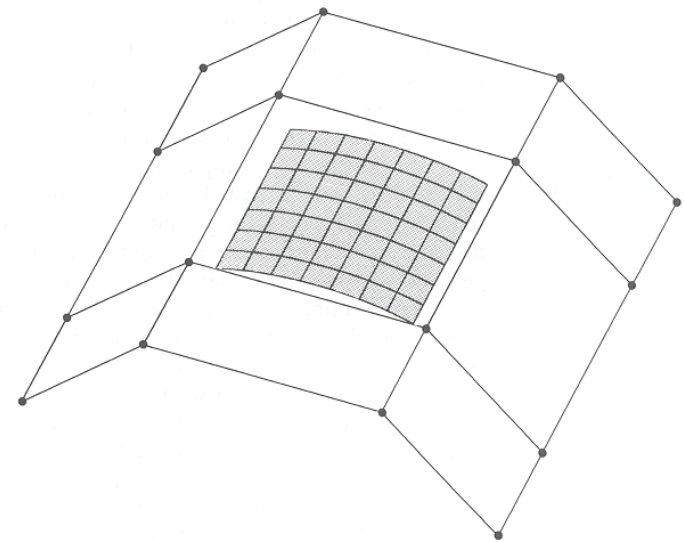
Where $\mathbf{M}$ is a matrix describing the blending functions for a parametric cubic curve (e.g., Bézier, B-spline, etc.)

B-Spline Patches



$$Q(u, v) = \mathbf{U} \mathbf{M}_{\text{B-Spline}} \begin{bmatrix} P_{1,1} & P_{1,2} & P_{1,3} & P_{1,4} \\ P_{2,1} & P_{2,2} & P_{2,3} & P_{2,4} \\ P_{3,1} & P_{3,2} & P_{3,3} & P_{3,4} \\ P_{4,1} & P_{4,2} & P_{4,3} & P_{4,4} \end{bmatrix} \mathbf{M}_{\text{B-Spline}}^T \mathbf{V}$$

$$\mathbf{M}_{\text{B-Spline}} = \begin{bmatrix} -1/6 & 1/2 & -1/2 & 1/6 \\ 1/2 & -1 & 1/2 & 0 \\ -1/2 & 0 & 1/2 & 0 \\ 1/6 & 2/3 & 1/6 & 0 \end{bmatrix}$$



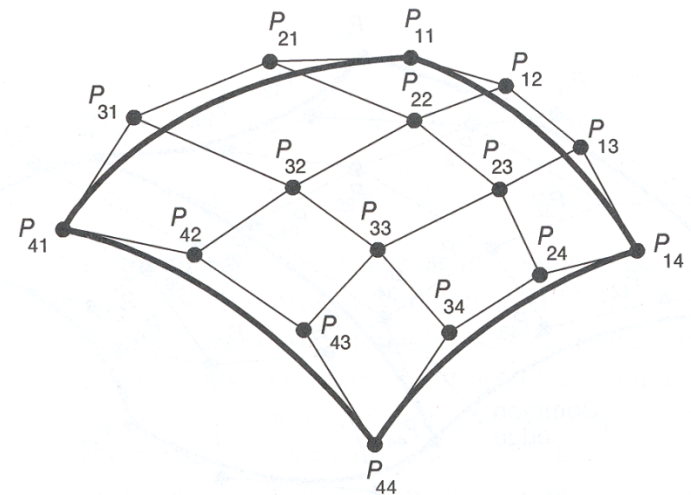
Watt Figure 6.28

Bézier Patches



$$Q(u, v) = \mathbf{U} \mathbf{M}_{\text{Bezier}} \begin{bmatrix} P_{1,1} & P_{1,2} & P_{1,3} & P_{1,4} \\ P_{2,1} & P_{2,2} & P_{2,3} & P_{2,4} \\ P_{3,1} & P_{3,2} & P_{3,3} & P_{3,4} \\ P_{4,1} & P_{4,2} & P_{4,3} & P_{4,4} \end{bmatrix} \mathbf{M}_{\text{Bezier}}^T \mathbf{V}$$

$$\mathbf{M}_{\text{Bezier}} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

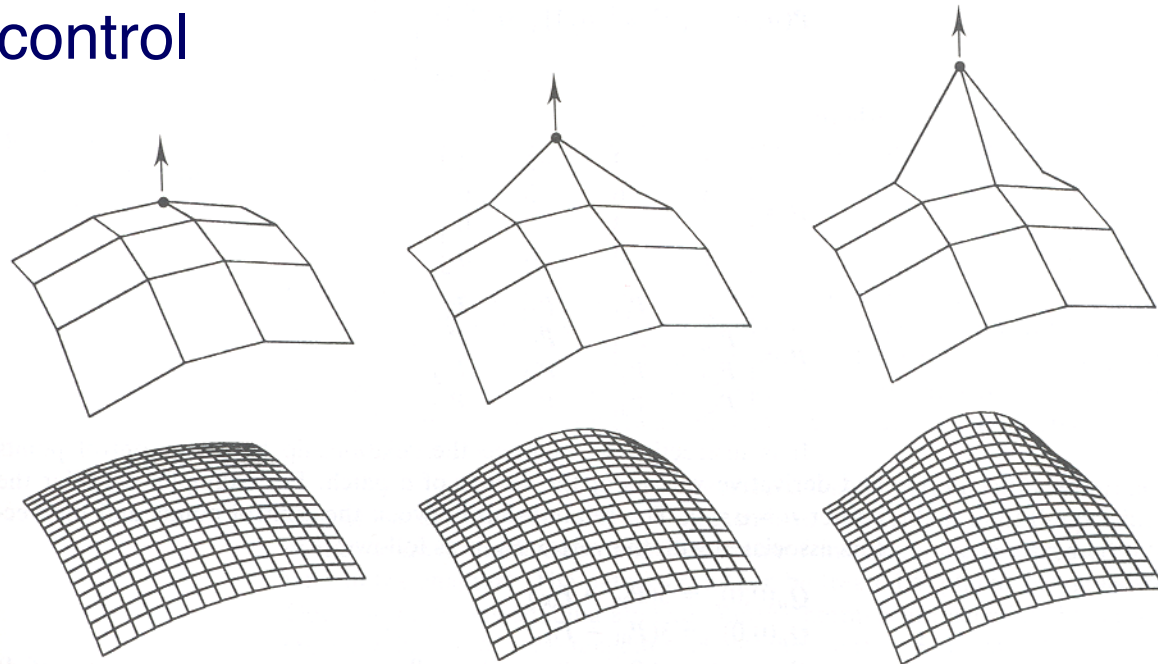


FvDFH Figure 11.42

Bézier Patches



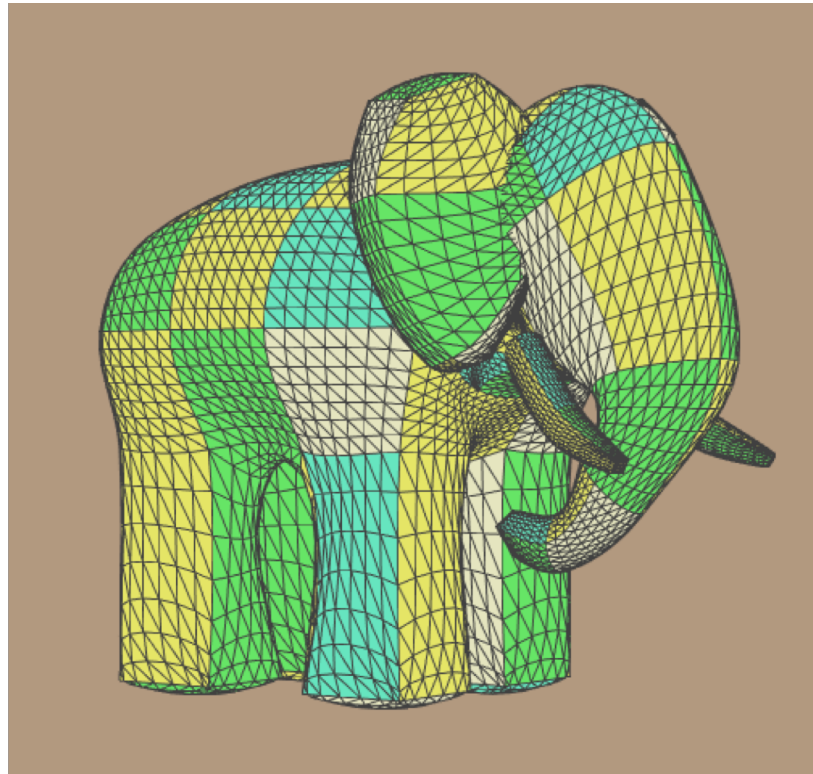
- Properties:
 - Interpolates four corner points
 - Convex hull
 - Local control



Piecewise Polynomial Parametric Surfaces



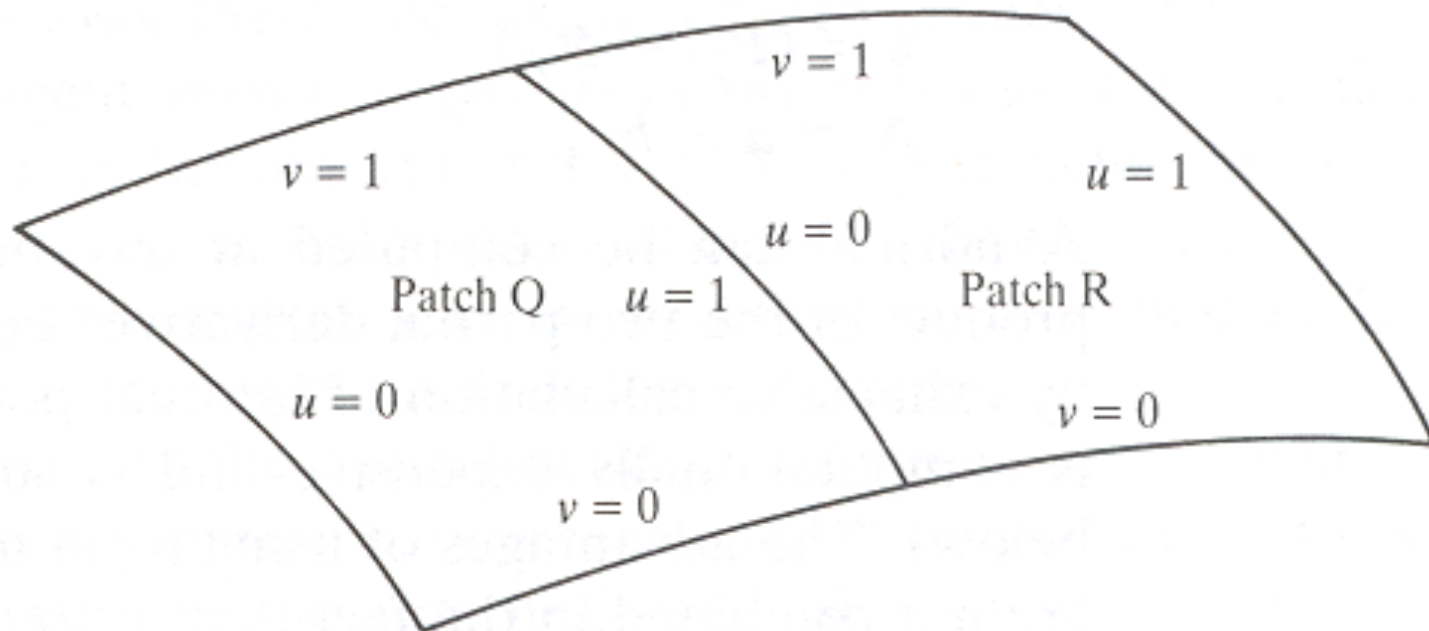
Surface is composition of many parametric patches



Piecewise Polynomial Parametric Surfaces



Must maintain continuity across seams



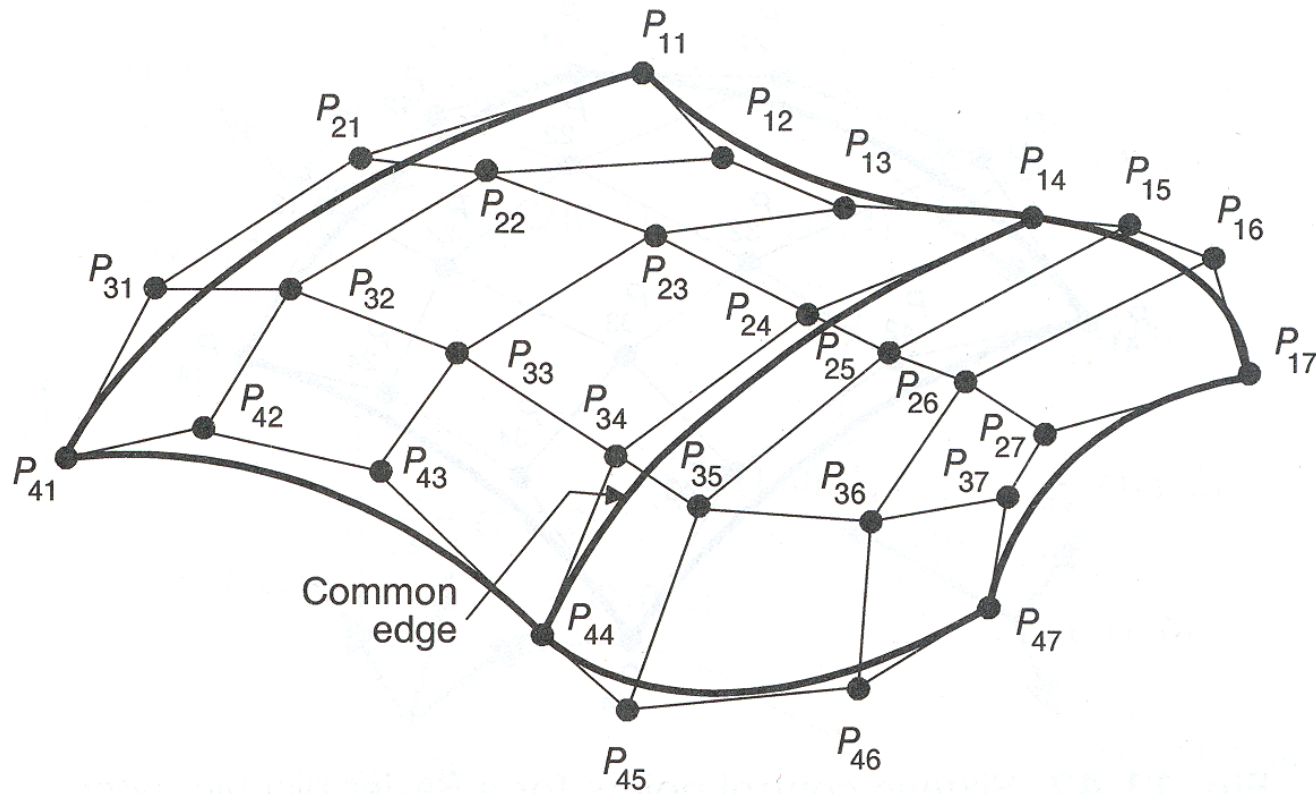
Same ideas as parametric splines!

Watt Figure 6.25

Bézier Surfaces



- Continuity constraints are similar to the ones for Bézier splines

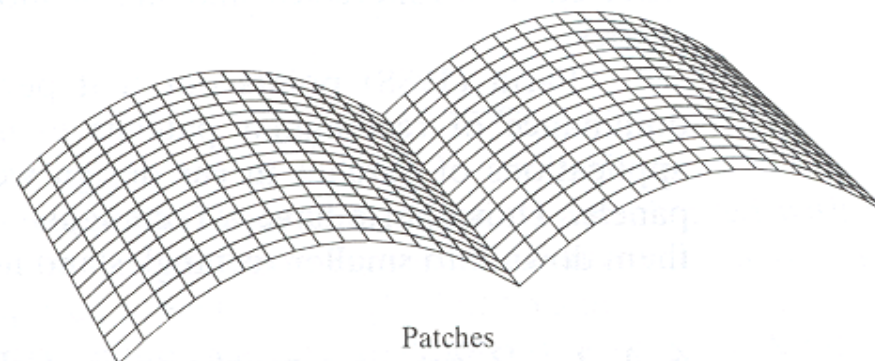
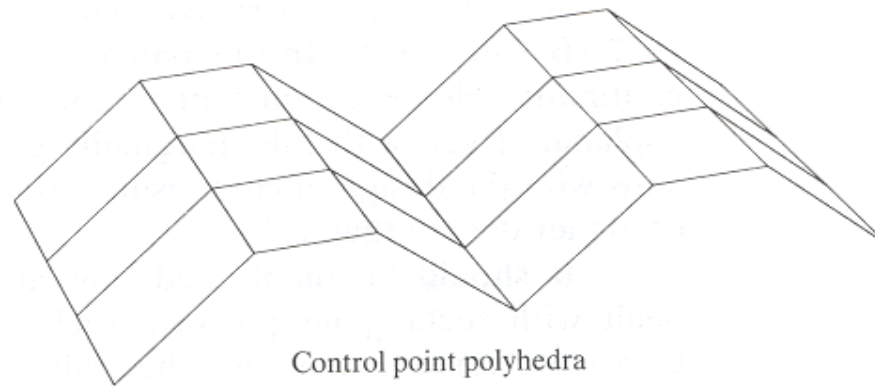


FvDFH Figure 11.43

Bézier Surfaces



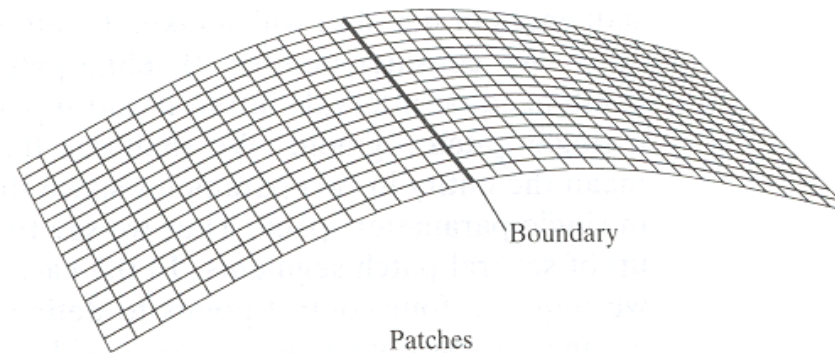
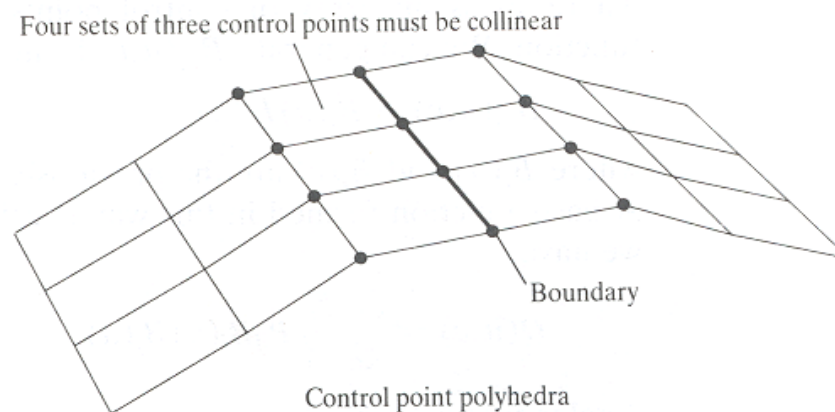
- C^0 continuity requires aligning boundary curves



Bézier Surfaces



- C^1 continuity requires aligning boundary curves and derivatives

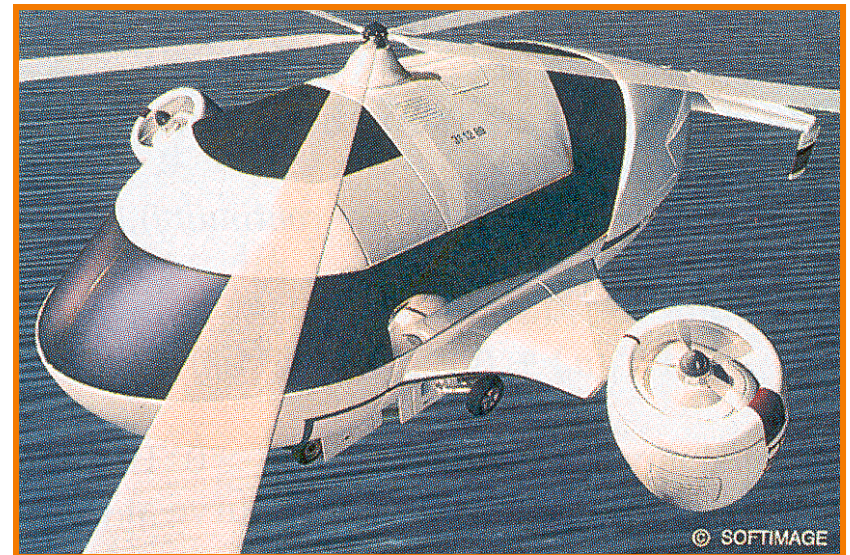


Watt Figure 6.26b

Parametric Surfaces



- Properties
 - ? Natural parameterization
 - ? Guaranteed smoothness
 - ? Intuitive editing
 - ? Concise
 - ? Accurate
 - ? Efficient display
 - ? Easy acquisition
 - ? Efficient intersections
 - ? Guaranteed validity
 - ? Arbitrary topology



Parametric Surfaces

- Properties
 - ☺ Natural parameterization
 - ☺ Guaranteed smoothness
 - ☺ Intuitive editing
 - ☺ Concise
 - ☺ Accurate
 - Efficient display
 - ☹ Easy acquisition
 - ☹ Efficient intersections
 - ☹ Guaranteed validity
 - ☹ Arbitrary topology

