Parametric Surfaces

COS 426

3D Object Representations

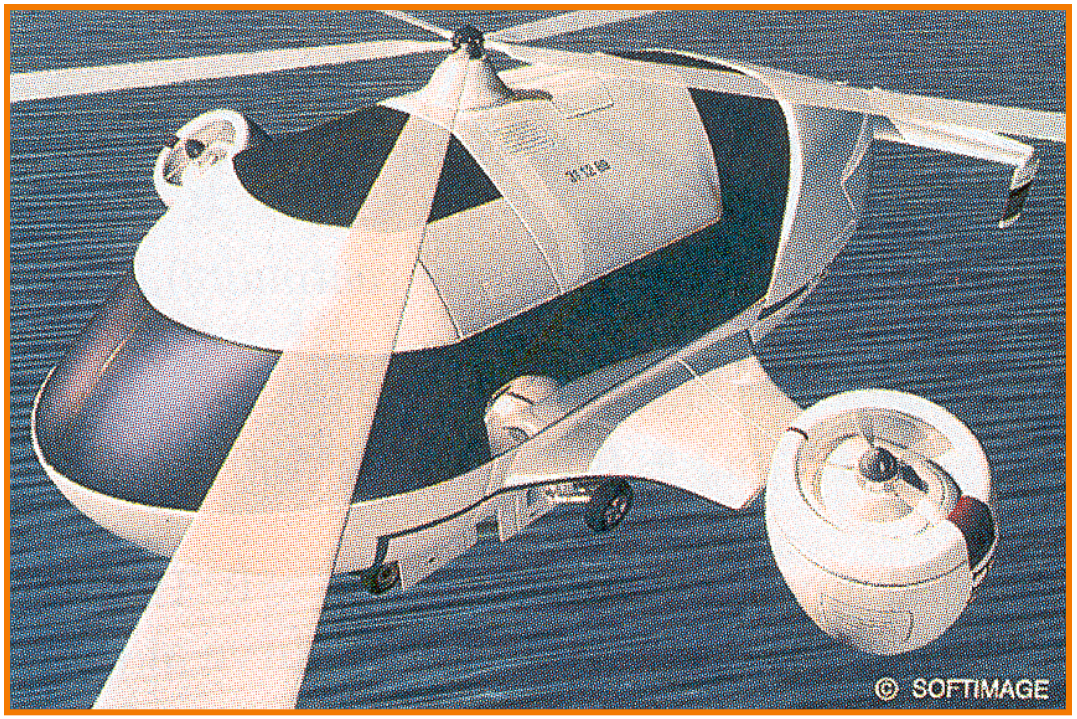
- Raw data
  - Voxels
  - Point cloud
  - Range image
  - Polygons
- Surfaces
  - Mesh
  - Subdivision
  - Parametric
  - Implicit
- Solids
  - Octree
  - BSP tree
  - CSG
  - Sweep
- High-level structures
  - Scene graph
  - Application specific

3D Object Representations

- Raw data
  - Voxels
  - Point cloud
  - Range image
  - Polygons
- Surfaces
  - Mesh
  - Subdivision
  - ➢ Parametric
  - Implicit
- Solids
  - Octree
  - BSP tree
  - CSG
  - Sweep
- High-level structures
  - Scene graph
  - Application specific

Parametric Surfaces

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



H&B Figure 10.46

Outline

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

Outline

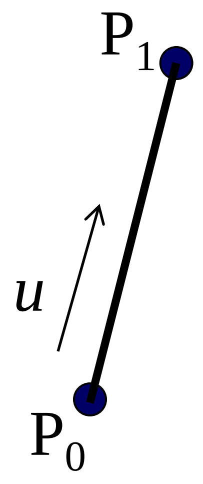
- ➢ Parametric curves
  - Cubic B-Spline
  - Cubic Bezier
- Parametric surfaces
  - Bi-cubic B-Spline
  - Bi-cubic Bezier

Parametric Curves

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

- Example: line segment

$$f_x(u) = (1-u)x_0 + ux_1$$
$$f_y(u) = (1-u)y_0 + uy_1$$



H&B Figure 10.10

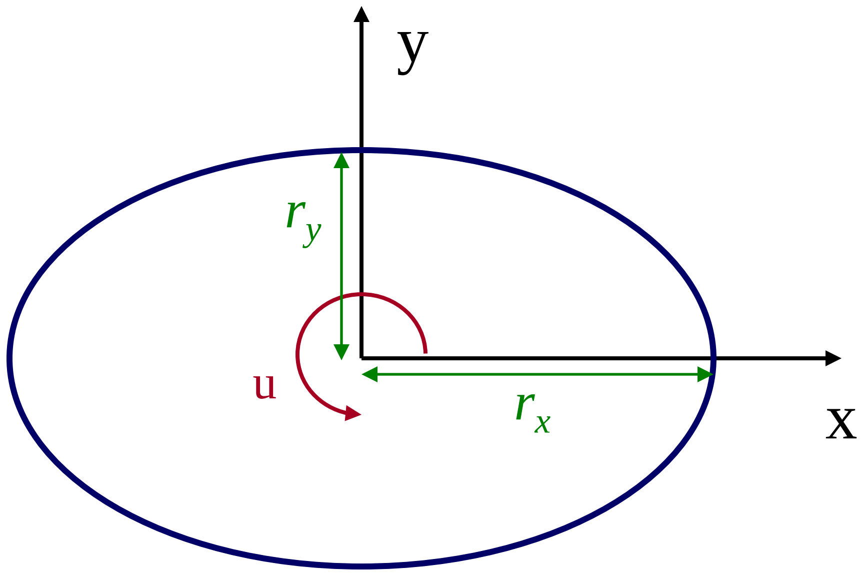
Parametric Curves

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

- Example: ellipse

$$f_x(u) = r_x \cos \frac{u}{2\pi}$$

$$f_y(u) = r_y \sin \frac{u}{2\pi}$$

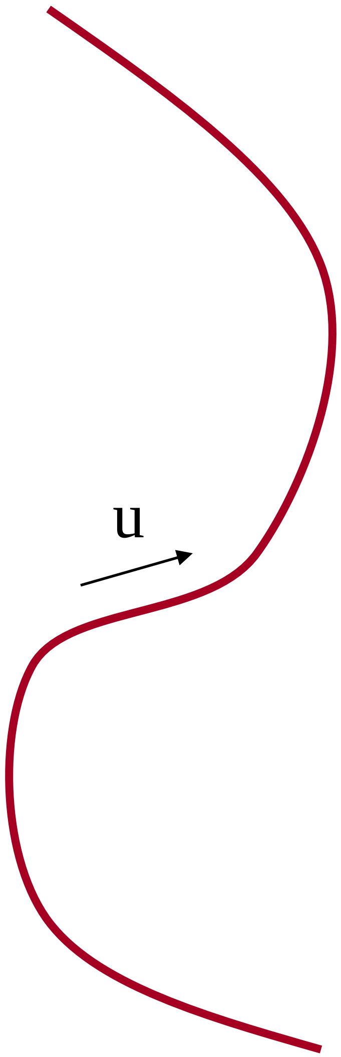


H&B Figure 10.10

Parametric curves

How to easily define arbitrary curves?

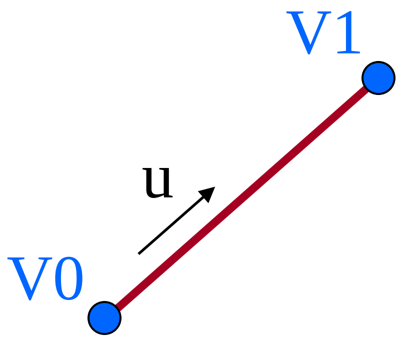
$$x = f_x(u)$$
$$y = f_y(u)$$



Parametric curves

How to easily define arbitrary curves?

$$x = f_x(u)$$
$$y = f_y(u)$$



Use functions that "blend" control points

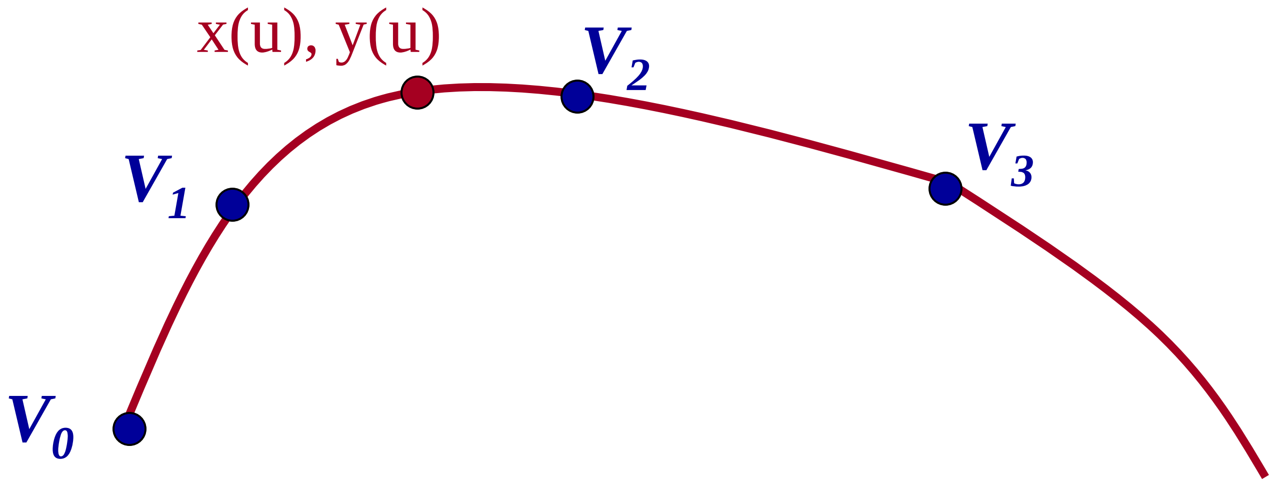
$$x = f_x(u) = V0_x*(1 - u) + V1_x*u$$
$$y = f_y(u) = V0_y*(1 - u) + V1_y*u$$

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?

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

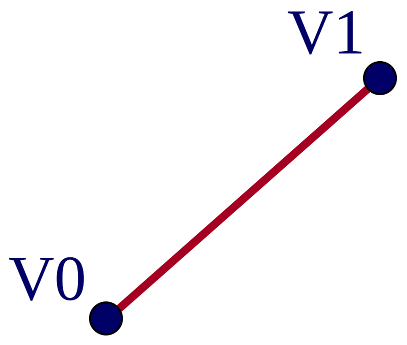
$$y(u) = \sum_{i=0}^{n} B_i(u) * Vi_y$$

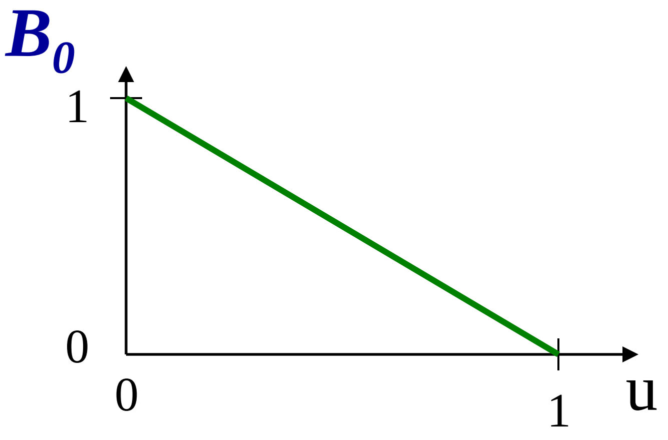
Parametric curves

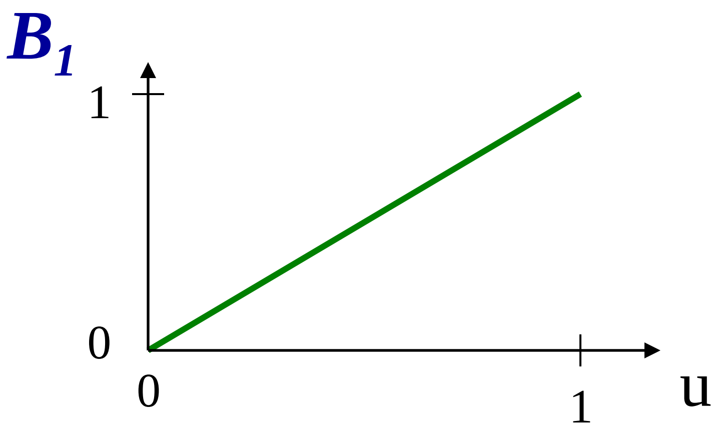
What B(u) functions should we use?

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

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





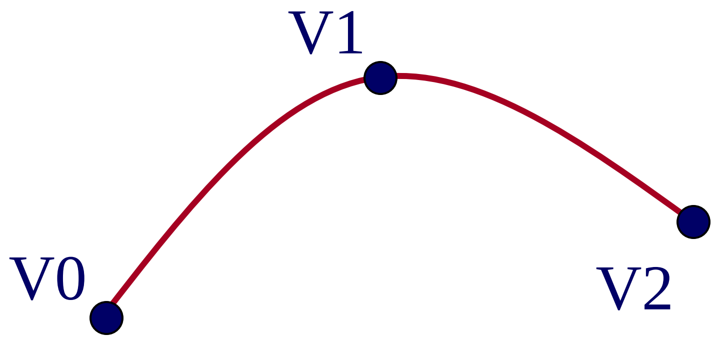


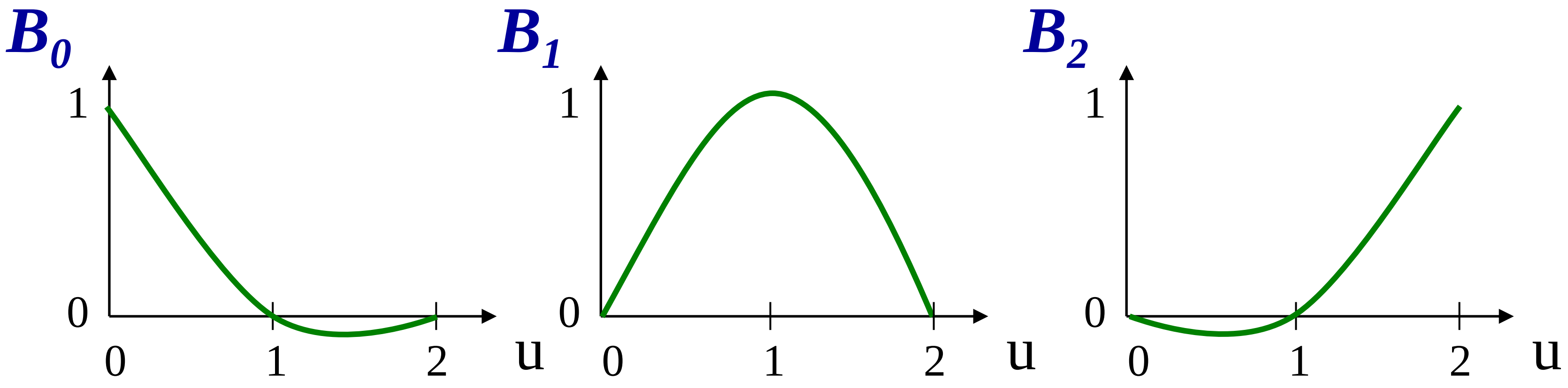
Parametric curves

What B(u) functions should we use?

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

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

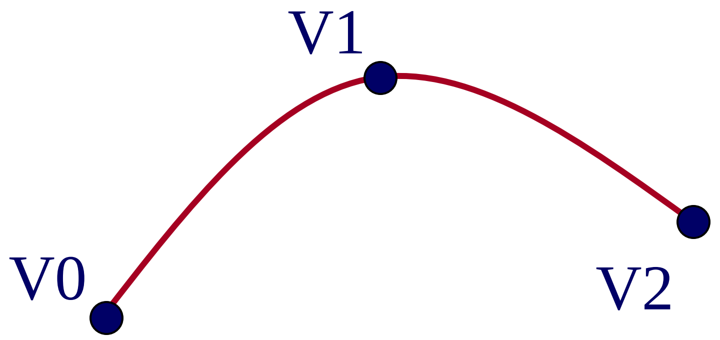




Parametric Polynomial Curves

- Polynomial blending functions:

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

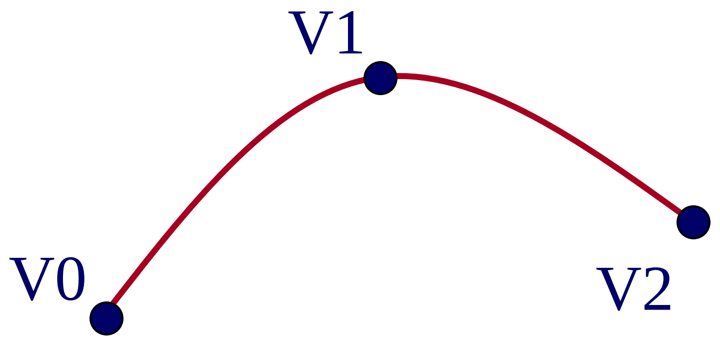


- Advantages of polynomials
  - Easy to compute
  - Infinitely continuous
  - Easy to derive curve properties

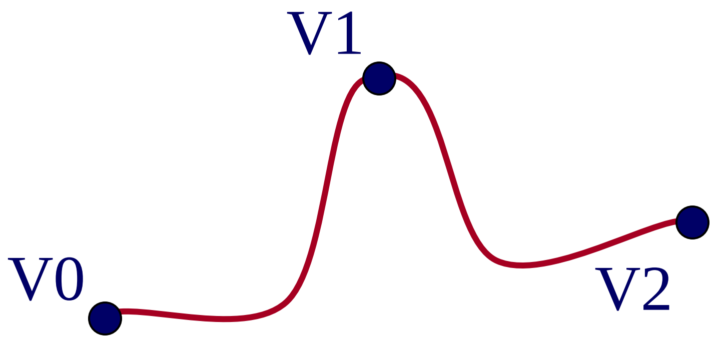
Parametric Polynomial Curves

- Polynomial blending functions:

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

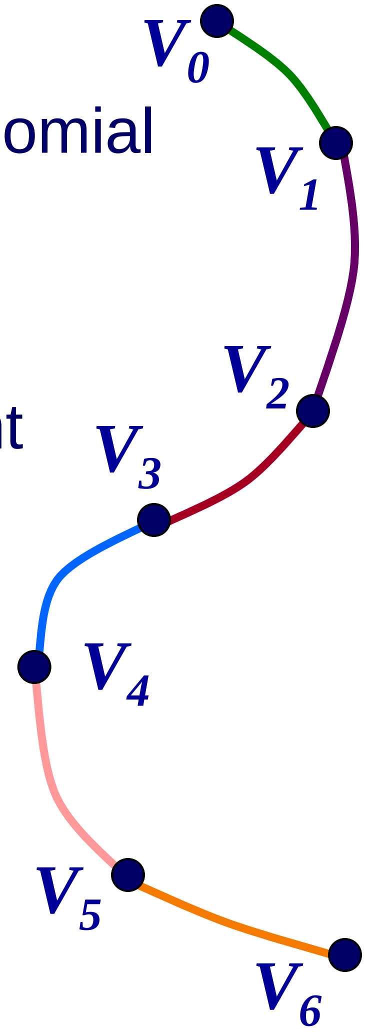


- What degree polynomial?
  - Easy to compute
  - Easy to control
  - Expressive



Piecewise Parametric Polynomial Curves

- Splines:
  - Split curve into segments
  - Each segment defined by low-order polynomial blending subset of control vertices
- Motivation:
  - Same blending function for every segment
  - Prove properties from blending functions
  - Provides control & efficiency
- Challenges
  - How choose blending functions?
  - How determine properties?



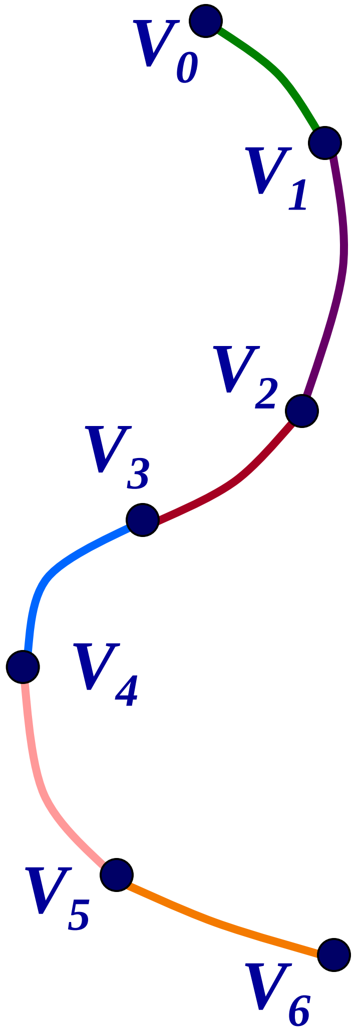
Cubic Splines

- Some properties we might like to have:
  - Local control
  - Interpolation
  - Continuity
  - Convex hull

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

Blending functions determine properties

Properties determine blending functions

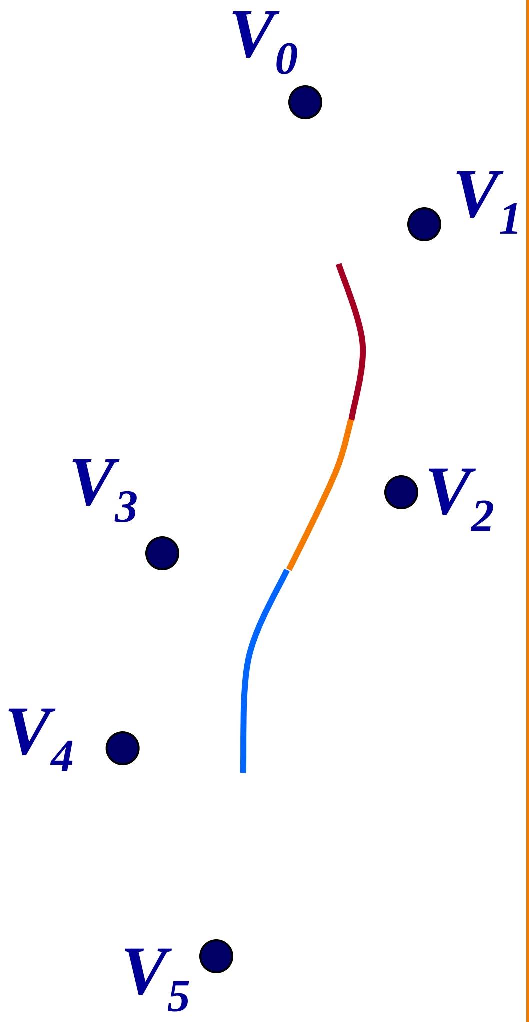


Outline

- Parametric curves
  - ➢ Cubic B-Spline
  - Cubic Bezier
- Parametric surfaces
  - Bi-cubic B-Spline
  - Bi-cubic Bezier

Cubic B-Splines

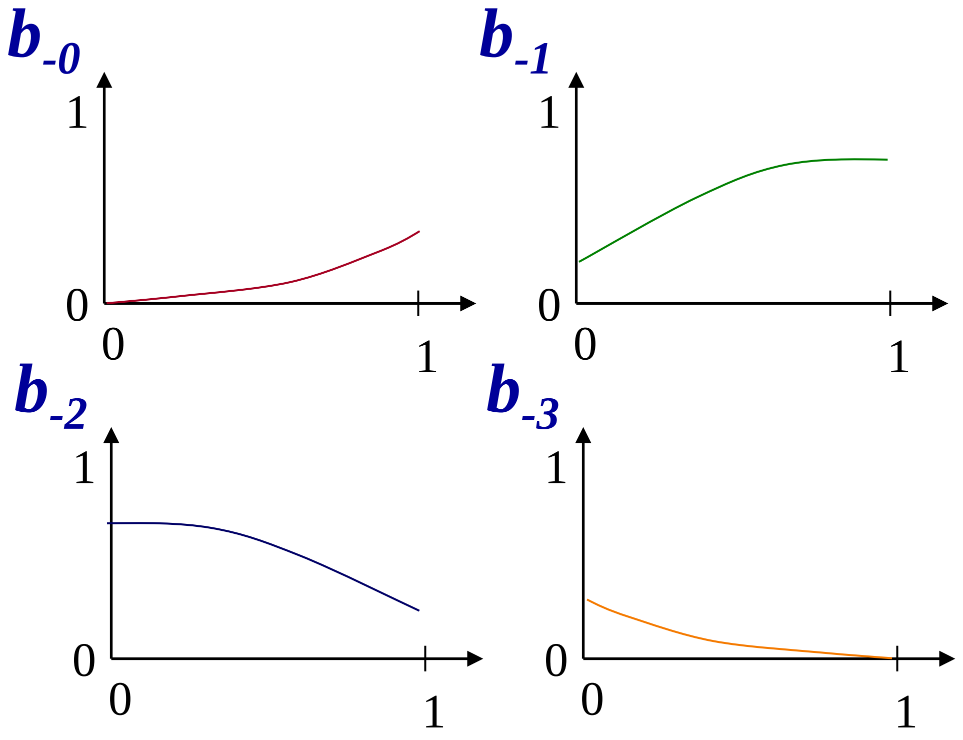
- Properties:
  - Local control
  - C^2 continuity
  - Approximating
  - Convex hull

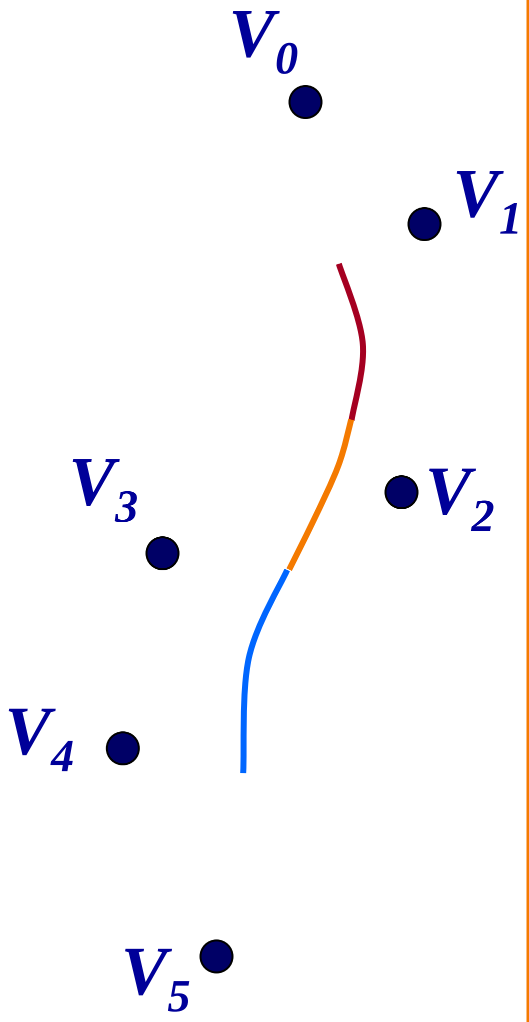


Cubic B-Spline Blending Functions

Blending functions:

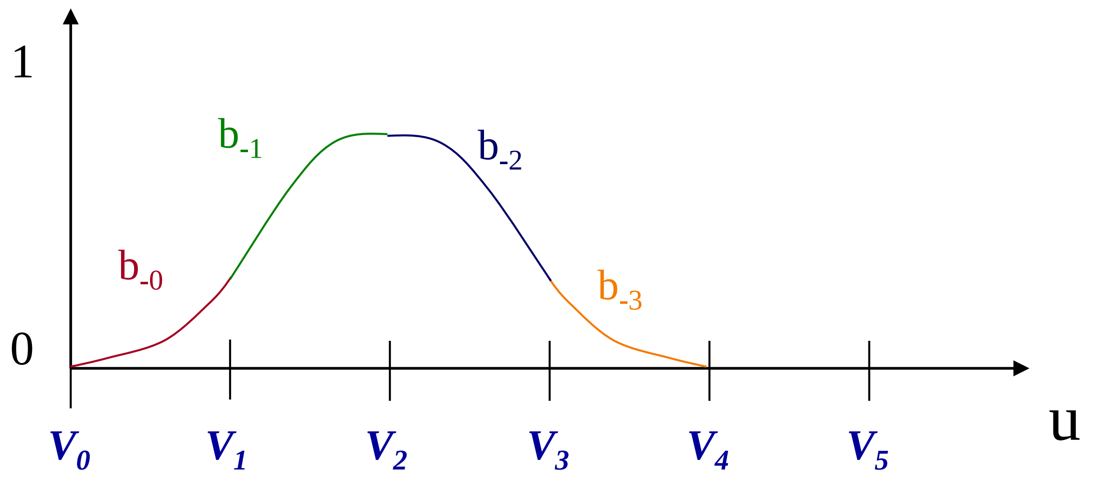
$$B_i(u) = \sum_{j=0}^{m} a_j u^j$$

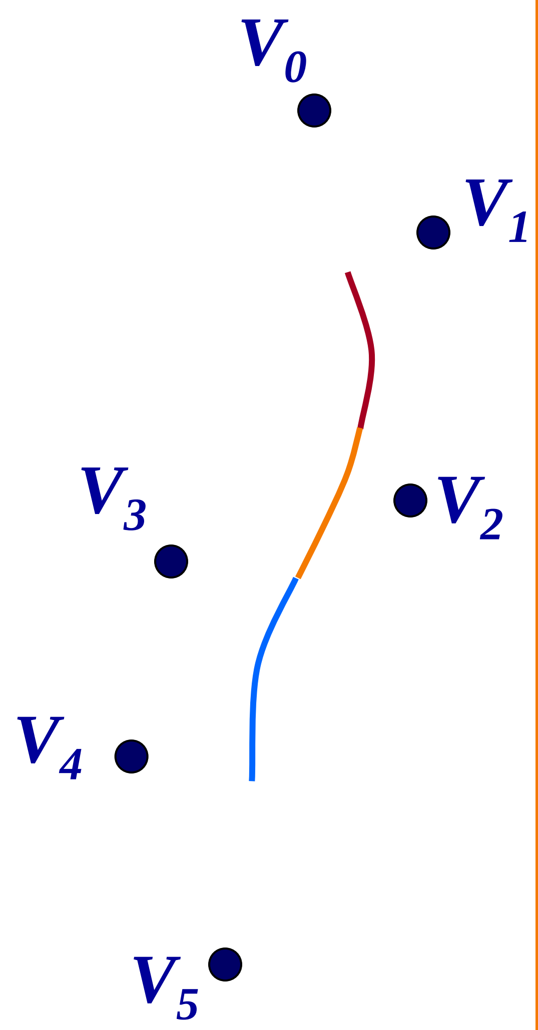




Cubic B-Spline Blending Functions

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

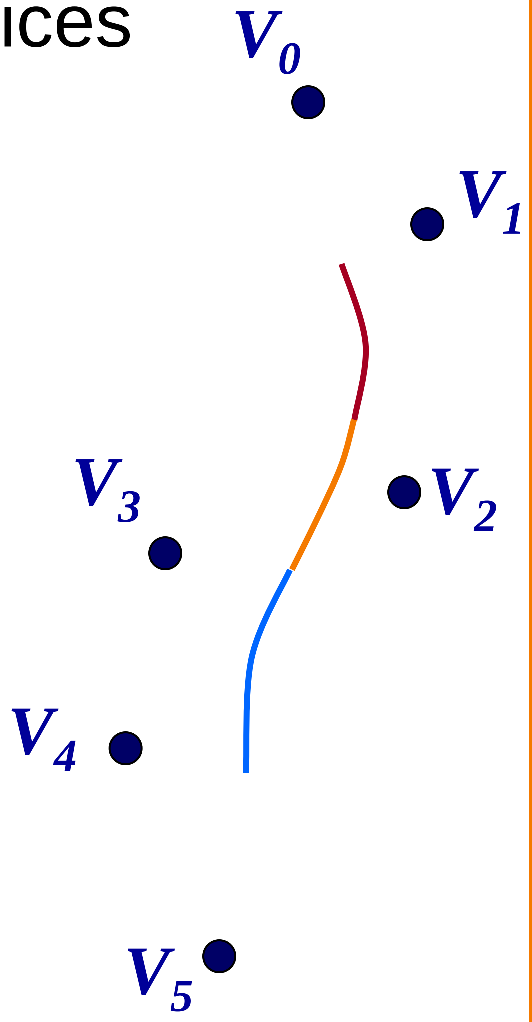




Cubic B-Spline Blending Functions

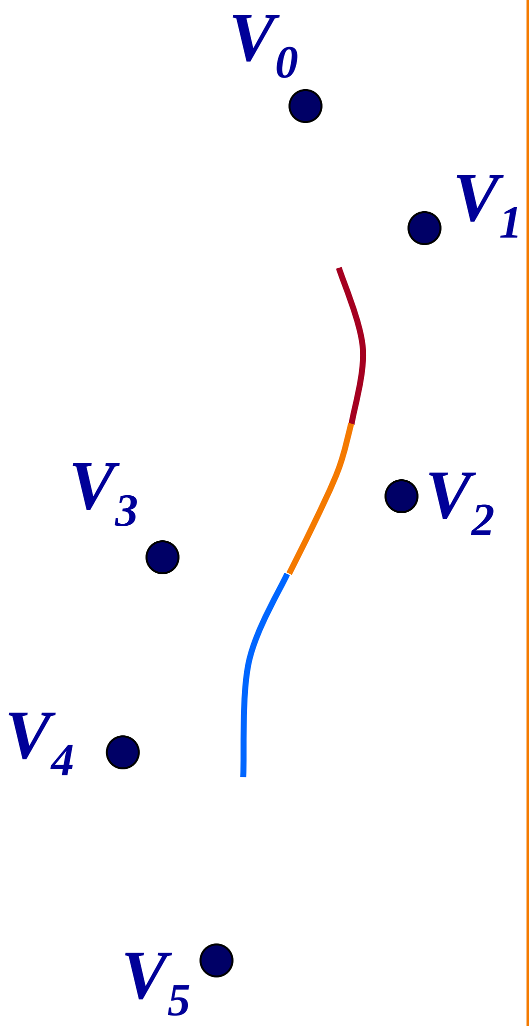
- Four cubic polynomials for four vertices
  - 16 variables (degrees of freedom)
  - Variables are a_i, b_i, c_i, d_i for four blending functions

$$
\boxed{\begin{aligned}
b_{-0}(u) &= a_0u^3 + b_0u^2 + c_0u^1 + d_0 \\
b_{-1}(u) &= a_1u^3 + b_1u^2 + c_1u^1 + d_1 \\
b_{-2}(u) &= a_2u^3 + b_2u^2 + c_2u^1 + d_2 \\
b_{-3}(u) &= a_3u^3 + b_3u^2 + c_3u^1 + d_3
\end{aligned}}
$$



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:

$$\begin{aligned} 0 &= b_{-0}(0) \\ b_{-0}(1) &= b_{-1}(0) \\ b_{-1}(1) &= b_{-2}(0) \\ b_{-2}(1) &= b_{-3}(0) \\ b_{-3}(1) &= 0 \end{aligned} \qquad \begin{aligned} 0 &= b_{-0}'(0) \\ b_{-0}'(1) &= b_{-1}'(0) \\ b_{-1}'(1) &= b_{-2}'(0) \\ b_{-2}'(1) &= b_{-3}'(0) \\ b_{-3}'(1) &= 0 \end{aligned} \qquad \begin{aligned} 0 &= b_{-0}''(0) \\ b_{-0}''(1) &= b_{-1}''(0) \\ b_{-1}''(1) &= b_{-2}''(0) \\ b_{-2}''(1) &= b_{-3}''(0) \\ b_{-3}''(1) &= 0 \end{aligned}$$

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:

$$
\begin{aligned}
b_{-3}(u) &= -\tfrac{1}{6}u^3 + \tfrac{1}{2}u^2 - \tfrac{1}{2}u + \tfrac{1}{6} \\
b_{-2}(u) &= \tfrac{1}{2}u^3 - u^2 + \tfrac{2}{3} \\
b_{-1}(u) &= -\tfrac{1}{2}u^3 + \tfrac{1}{2}u^2 + \tfrac{1}{2}u + \tfrac{1}{6} \\
b_{-0}(u) &= \tfrac{1}{6}u^3
\end{aligned}
$$

Cubic B-Spline Blending Functions

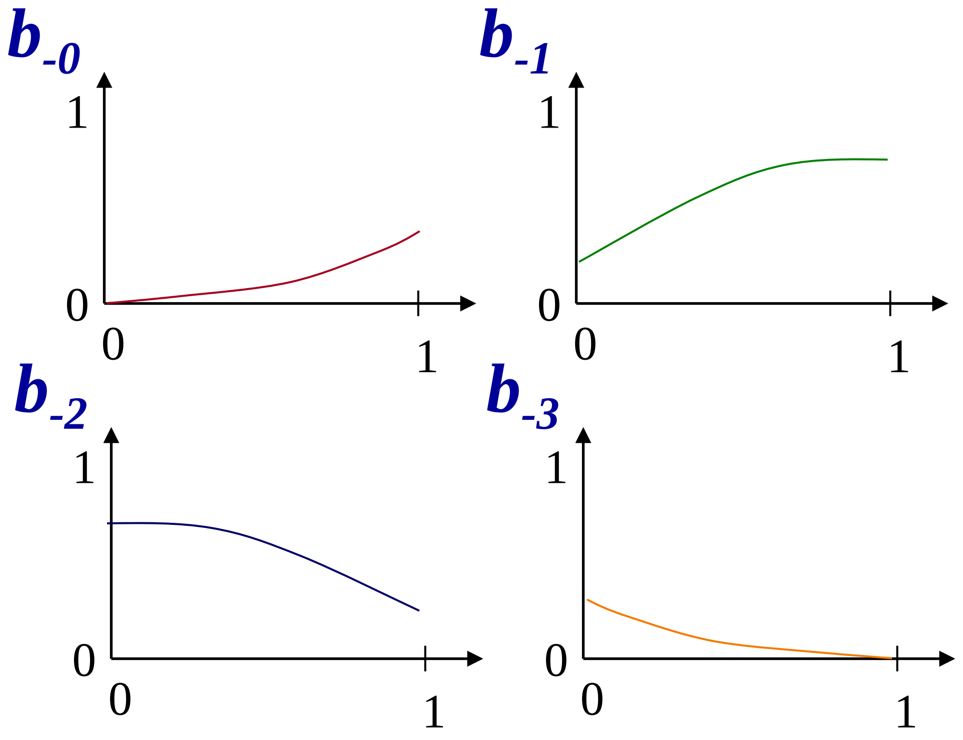
- In matrix form:

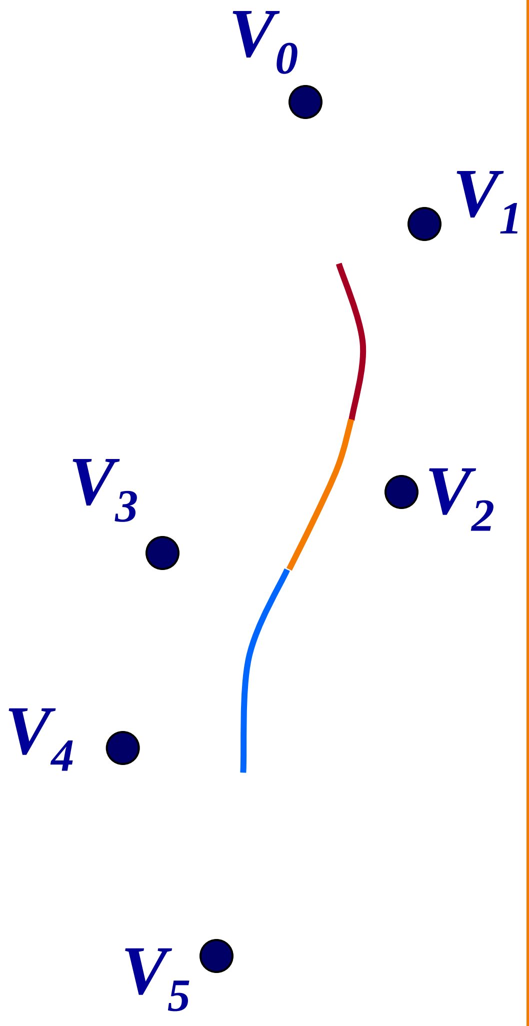
$$Q(u) = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \frac{1}{6} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix} \begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix}$$

Cubic B-Spline Blending Functions

In plot form:

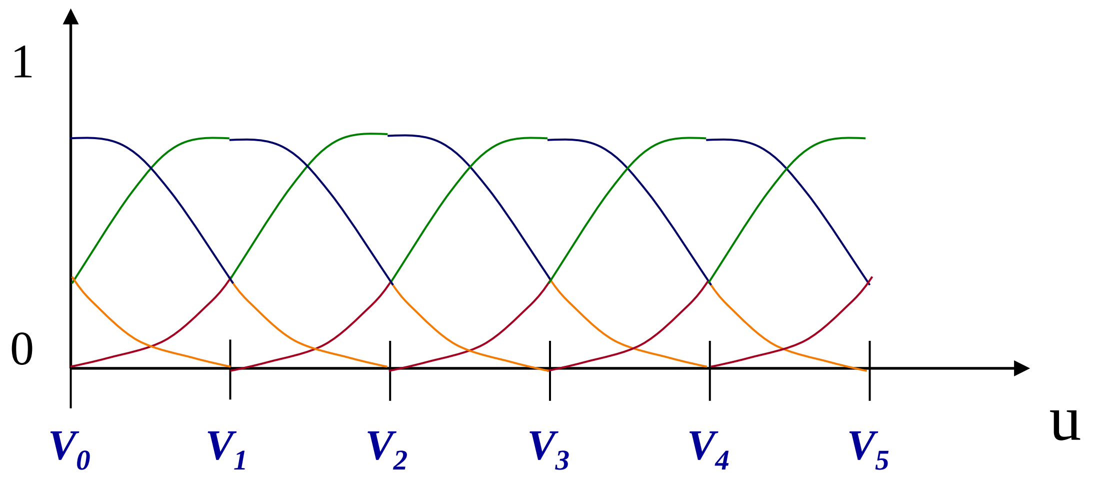
$$B_i(u) = \sum_{j=0}^{m} a_j u^j$$

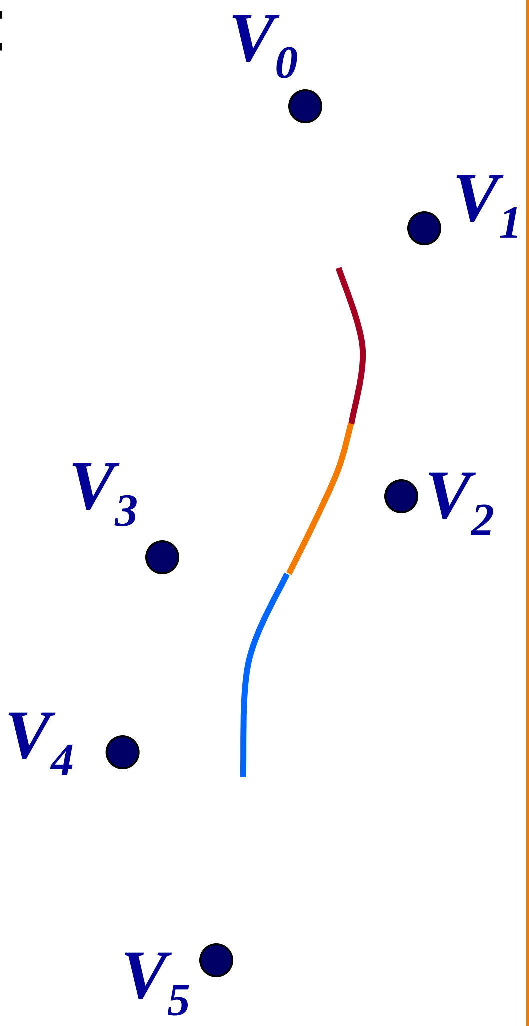




Cubic B-Spline Blending Functions

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





Outline

- Parametric curves
  - Cubic B-Spline
  - ➢ Cubic Bezier
- Parametric surfaces
  - Bi-cubic B-Spline
  - Bi-cubic Bezier

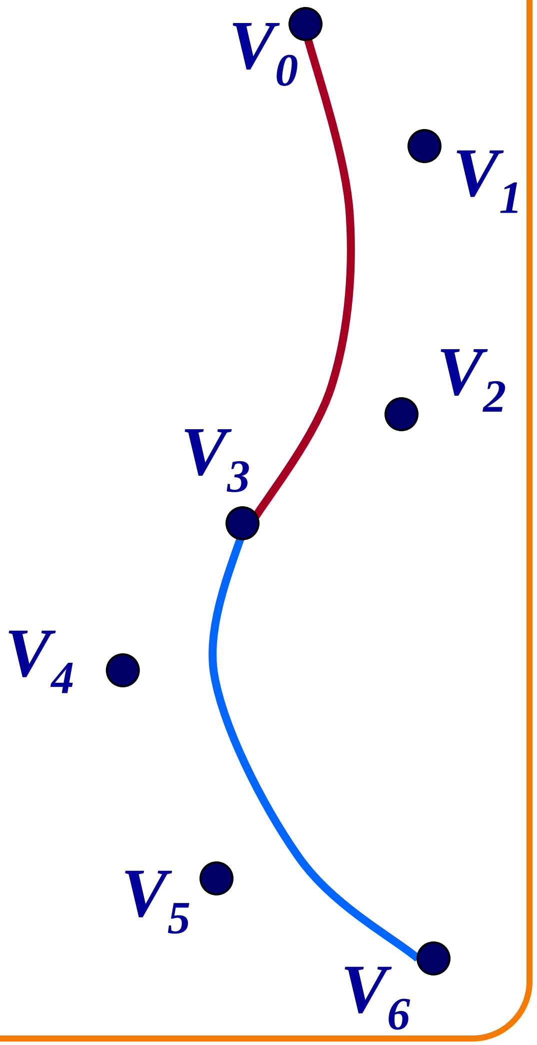
Cubic Bezier

- Developed around 1960 by both
  - Pierre Bézier (Renault)
  - Paul de Casteljau (Citroen)

- Properties:
  - Local control
  - Continuity depends on control points
  - Interpolating (every third)

Properties determine blending functions

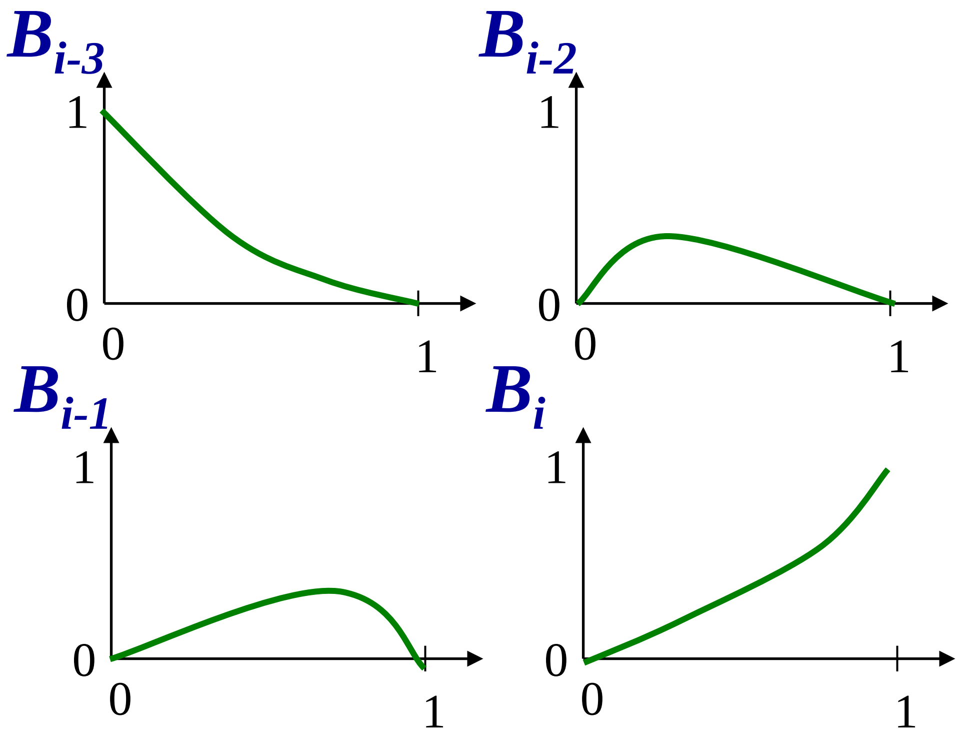
Blending functions determine properties

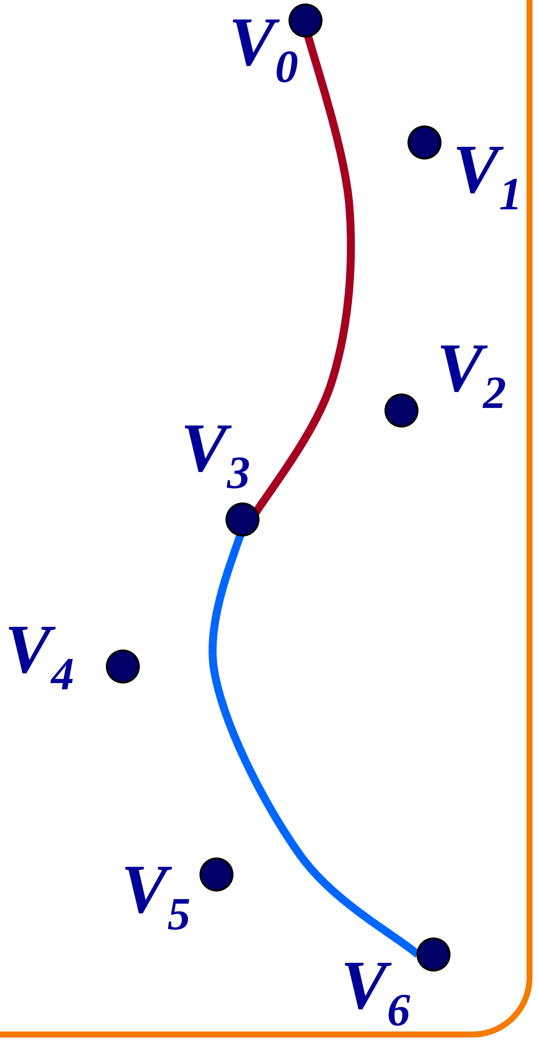


Cubic Bezier curves

Blending functions:

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





Cubic Bezier Curves

Bézier curves in matrix form:

$$Q(u) = \sum_{i=0}^{n} V_i \binom{n}{i} u^i (1-u)^{n-i}$$

$$= (1-u)^3 V_0 + 3u(1-u)^2 V_1 + 3u^2(1-u)V_2 + u^3 V_3$$

$$= \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \underbrace{\begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}}_{\mathbf{M}_{\mathbf{Bezier}}} \begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \end{pmatrix}$$

Basic properties of Bézier curves

- Endpoint interpolation:

$$Q(0) = V_0$$

$$Q(1) = V_n$$

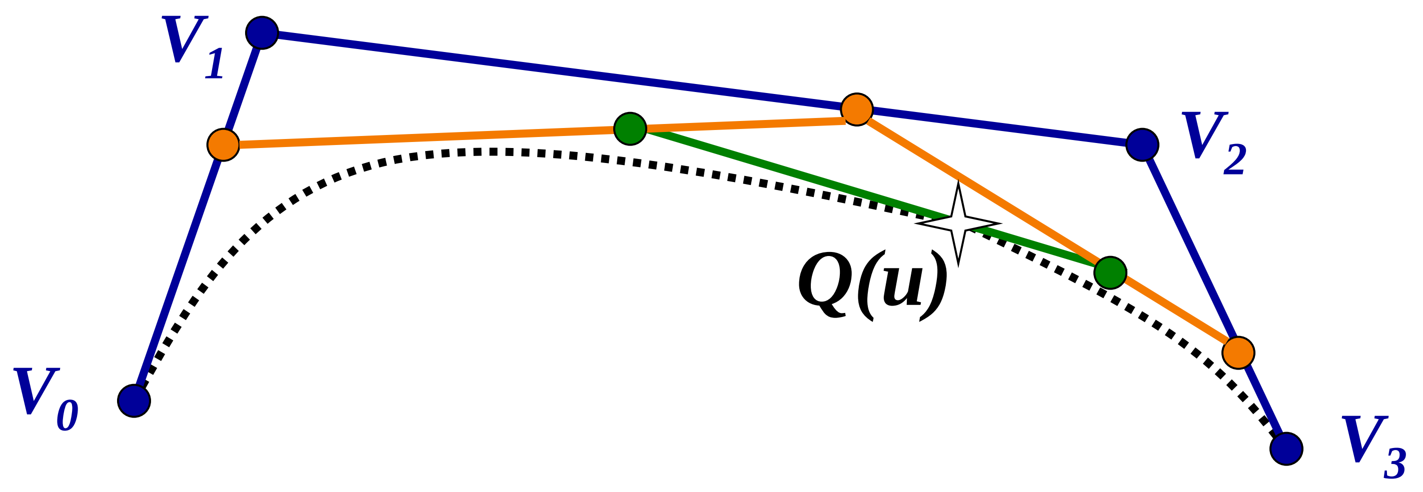
- Convex hull:
  - Curve is contained within convex hull of control polygon

- Symmetry

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

Bézier curves

- Curve $\boldsymbol{Q(u)}$ can also be defined by nested interpolation:



V_i are *control points*

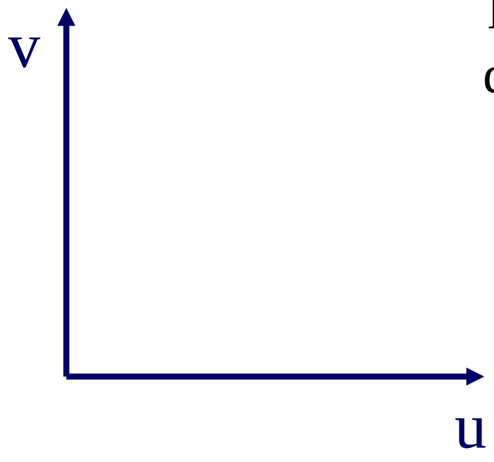
$\{V_0, V_1, \ldots, V_n\}$ is *control polygon*

Outline

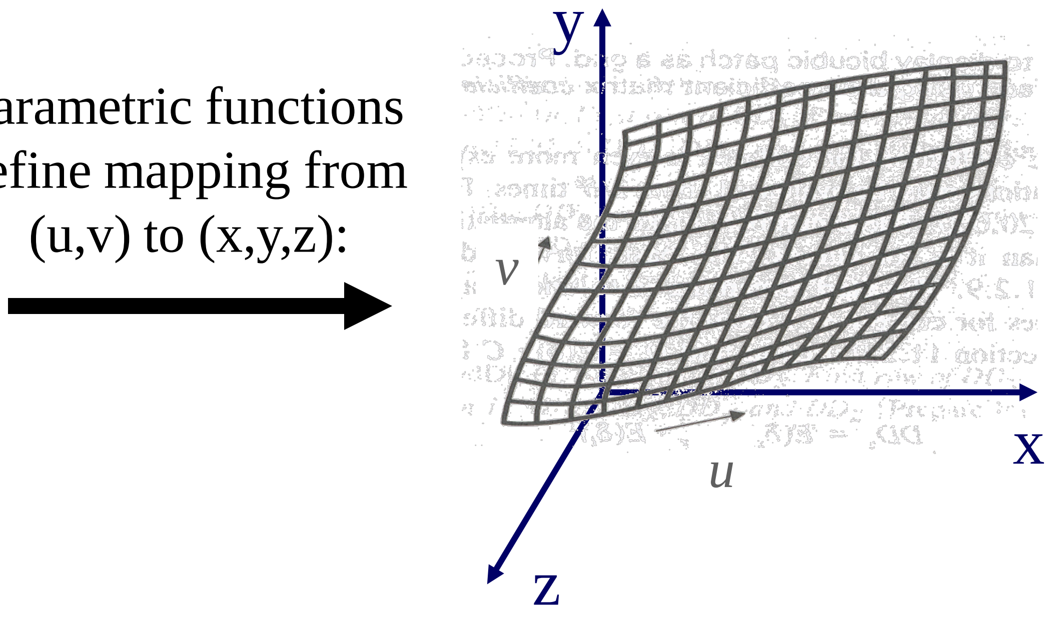
- Parametric curves
  - Cubic B-Spline
  - Cubic Bezier
- ➢ Parametric surfaces
  - Bi-cubic B-Spline
  - Bi-cubic Bezier

Parametric Surfaces

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



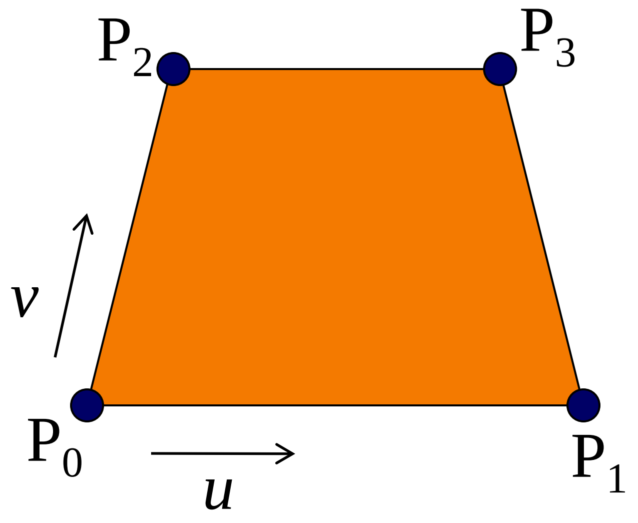
Parametric functions define mapping from (u,v) to (x,y,z):



FvDFH Figure 11.42

Parametric Surfaces

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



- Example: quadrilateral

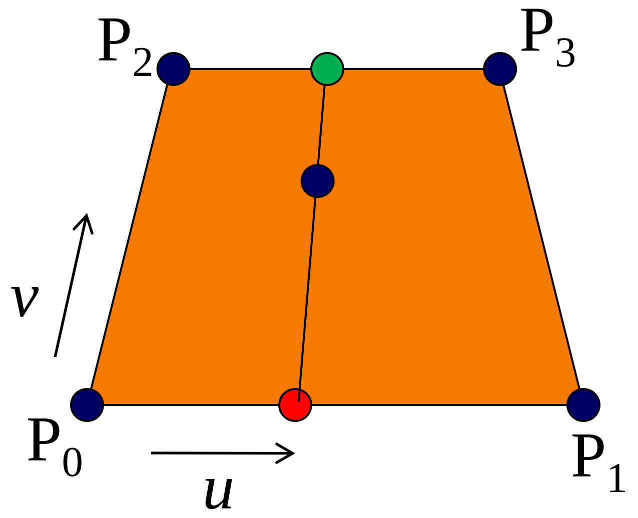
$$f_x(u,v) = (1-v)((1-u)x_0 + ux_1) + v((1-u)x_2 + ux_3)$$

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

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

Parametric Surfaces

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



- Example: quadrilateral

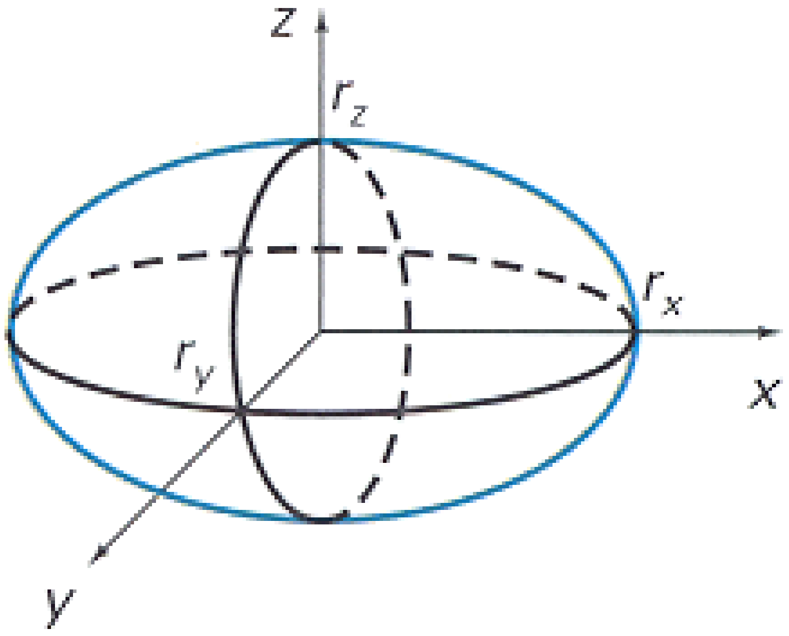
$$f_x(u,v) = (1-v)\boxed{((1-u)x_0 + ux_1)} + v\boxed{((1-u)x_2 + ux_3)}$$
$$f_y(u,v) = (1-v)\boxed{((1-u)y_0 + uy_1)} + v\boxed{((1-u)y_2 + uy_3)}$$
$$f_z(u,v) = (1-v)\boxed{((1-u)z_0 + uz_1)} + v\boxed{((1-u)z_2 + uz_3)}$$

Parametric Surfaces

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

- Example: ellipsoid

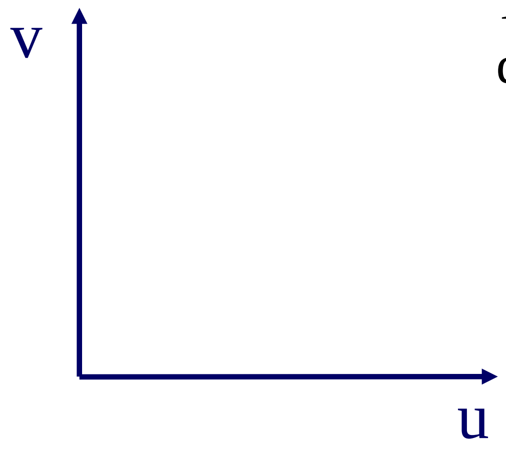
$$f_x(u,v) = r_x \cos v \cos u$$
$$f_y(u,v) = r_y \cos v \sin u$$
$$f_z(u,v) = r_z \sin v$$



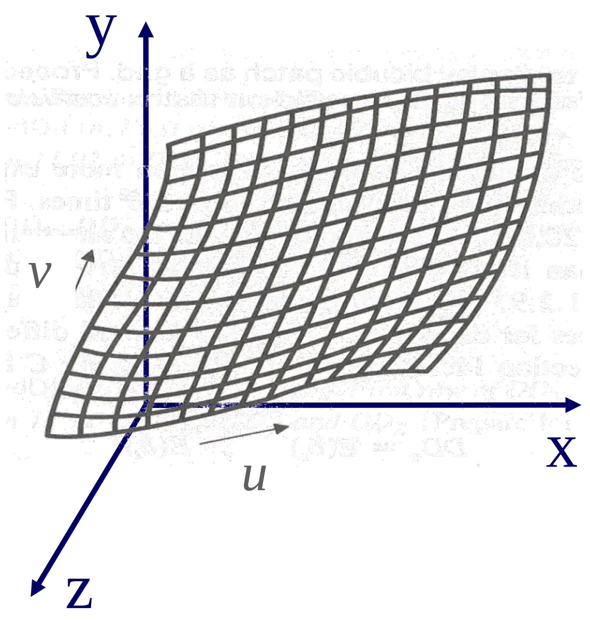
H&B Figure 10.10

Parametric Surfaces

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



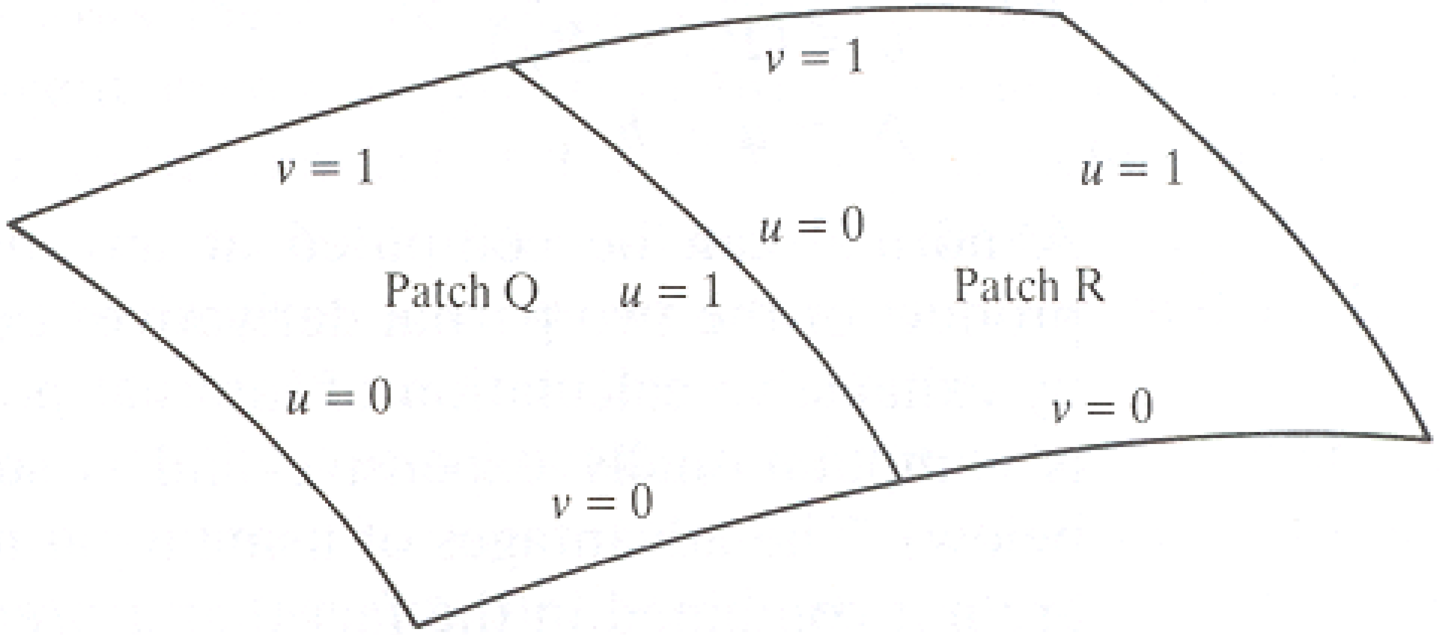
Parametric functions define mapping from (u,v) to (x,y,z):



FvDFH Figure 11.42

Piecewise Polynomial Parametric Surfaces

- Surface is partitioned into parametric patches:

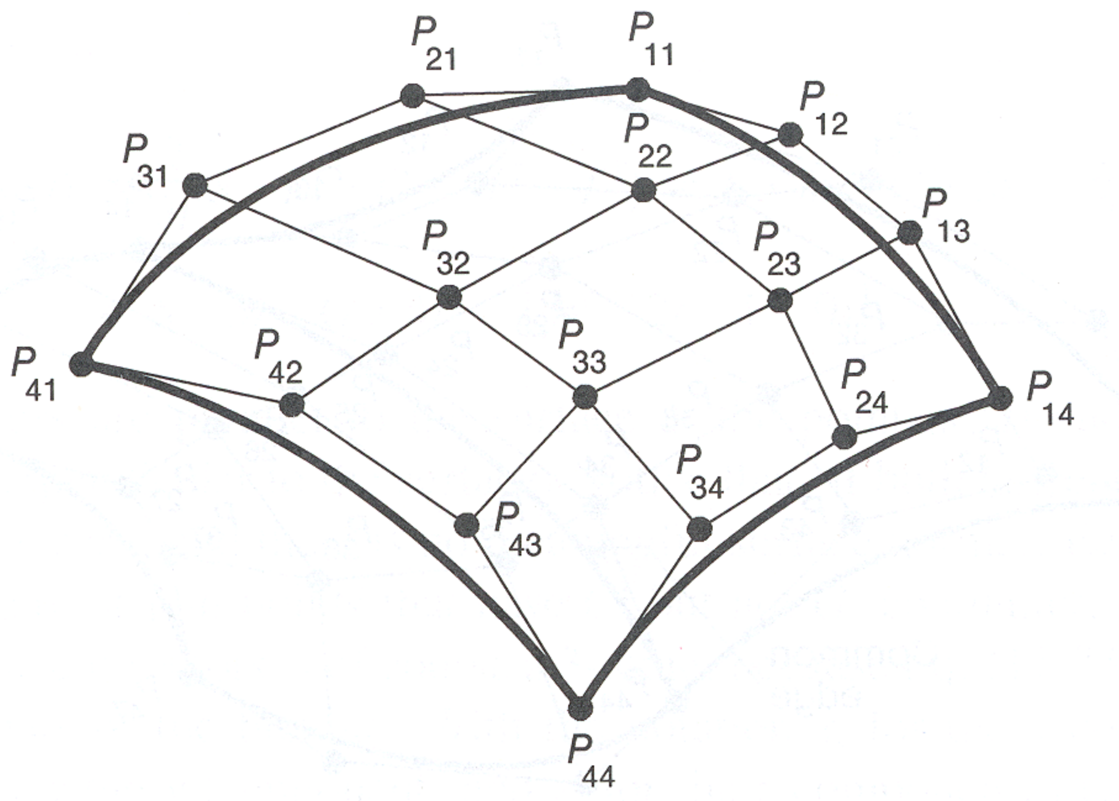


Same ideas as parametric splines!

Watt Figure 6.25

Parametric Patches

- Each patch is defined by blending control points

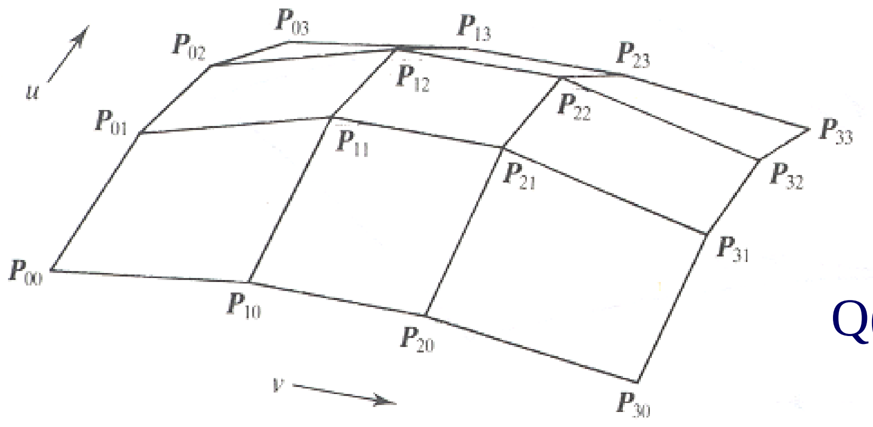


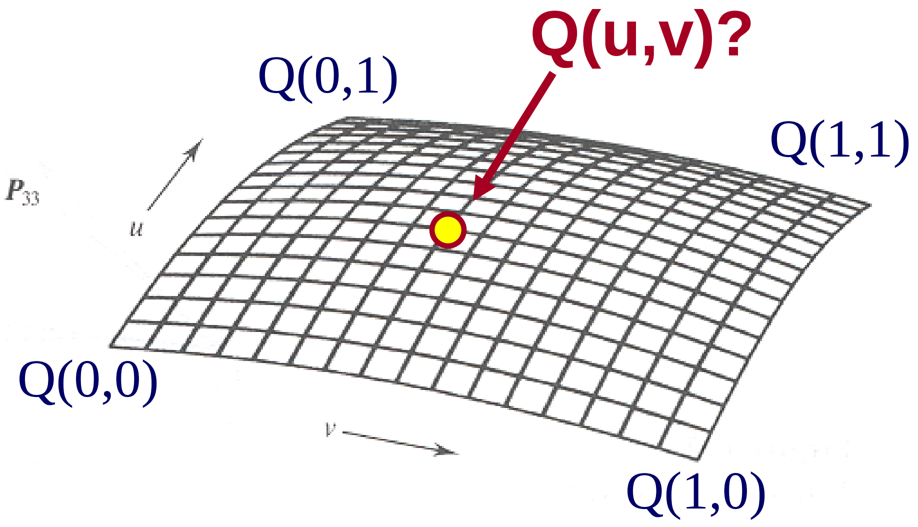
Same ideas as parametric curves!

FvDFH Figure 11.44

Parametric Patches

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

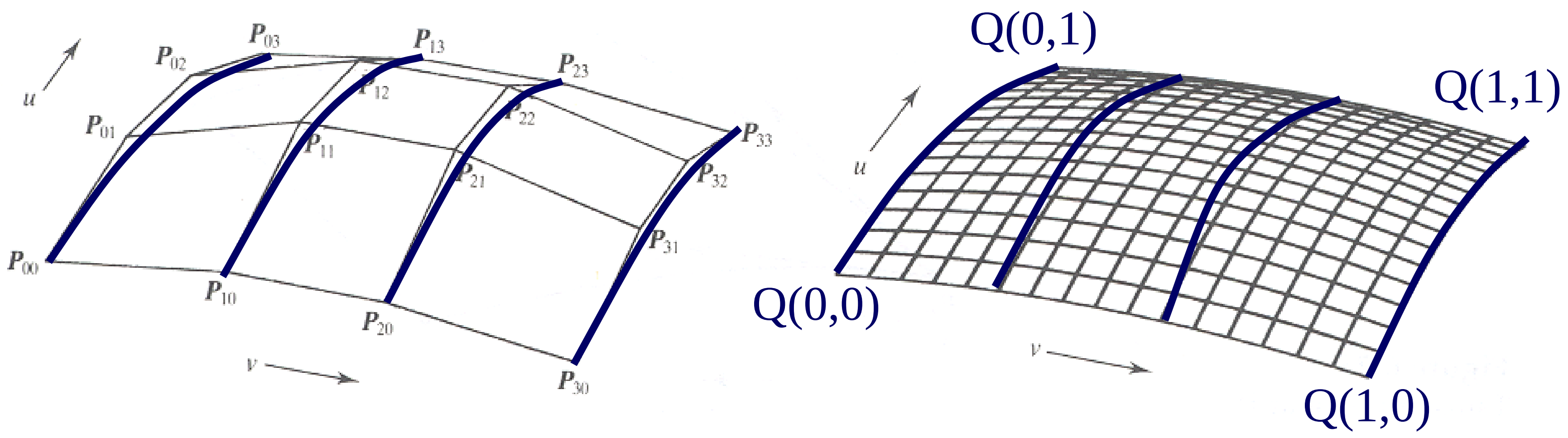




Watt Figure 6.21

Parametric Patches

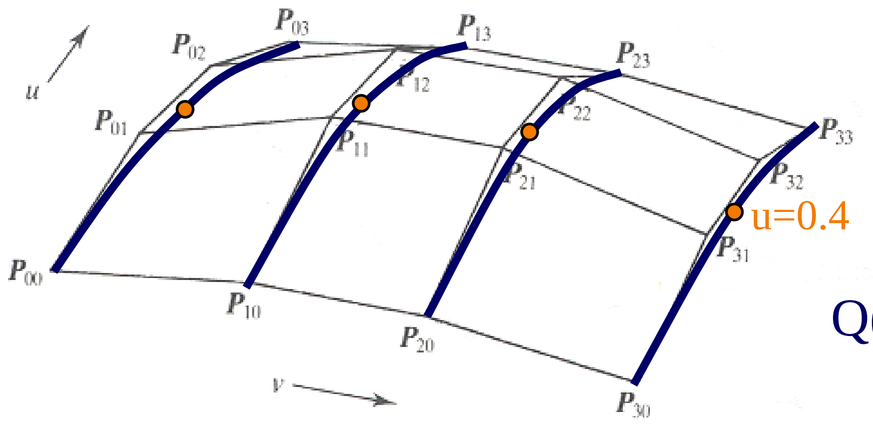
- Point Q(u,v) on the patch is the tensor product of parametric curves defined by the control points

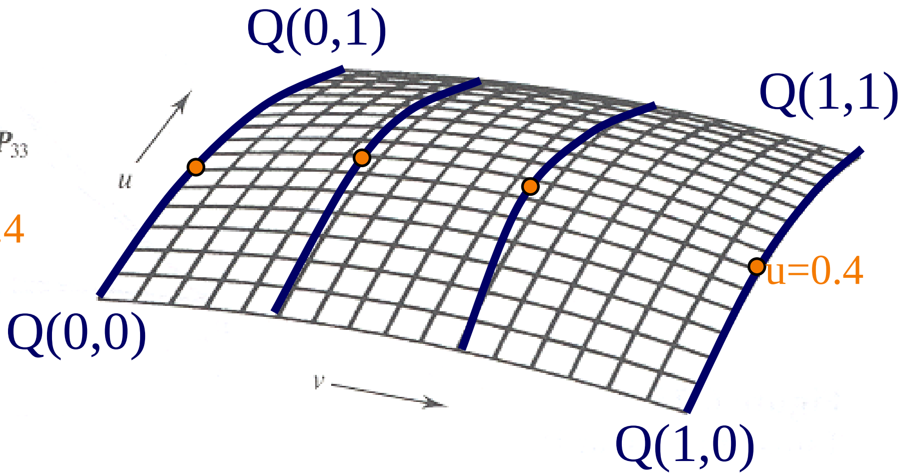


Watt Figure 6.21

Parametric Patches

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

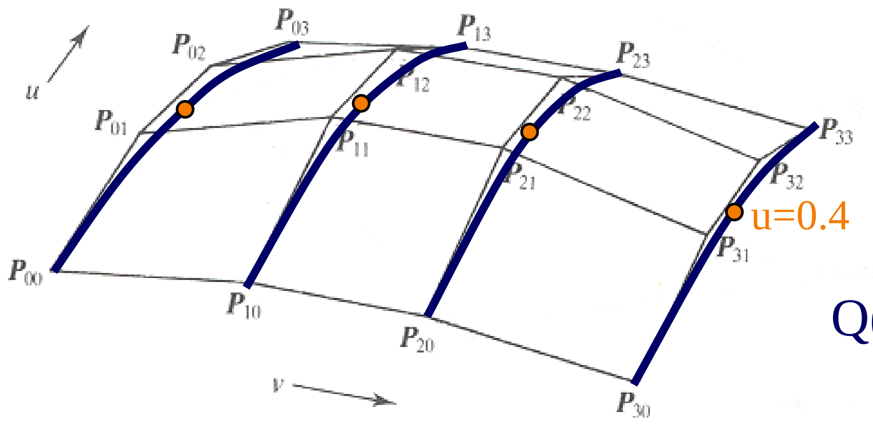


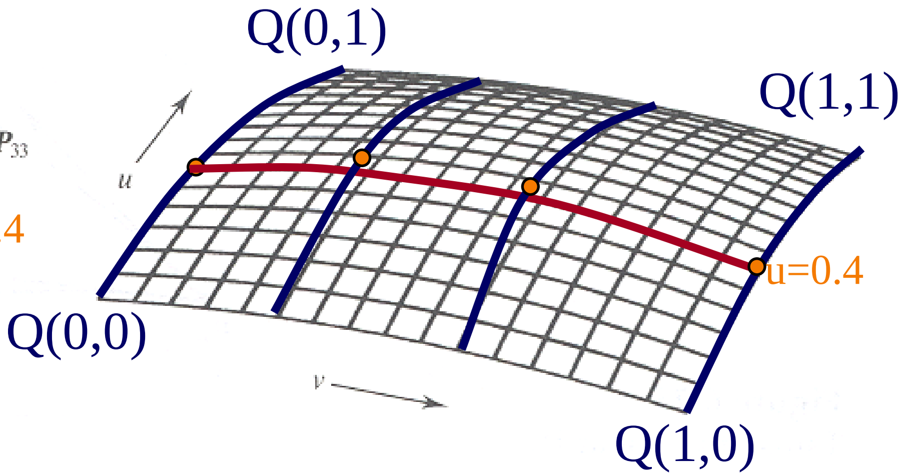


Watt Figure 6.21

Parametric Patches

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

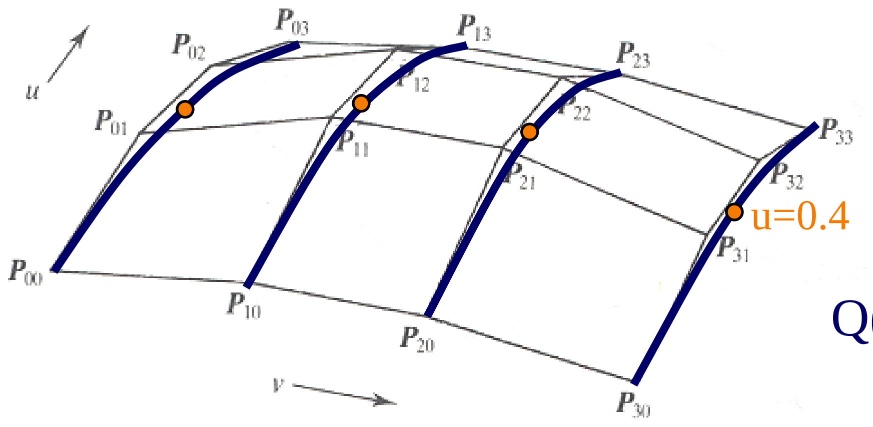


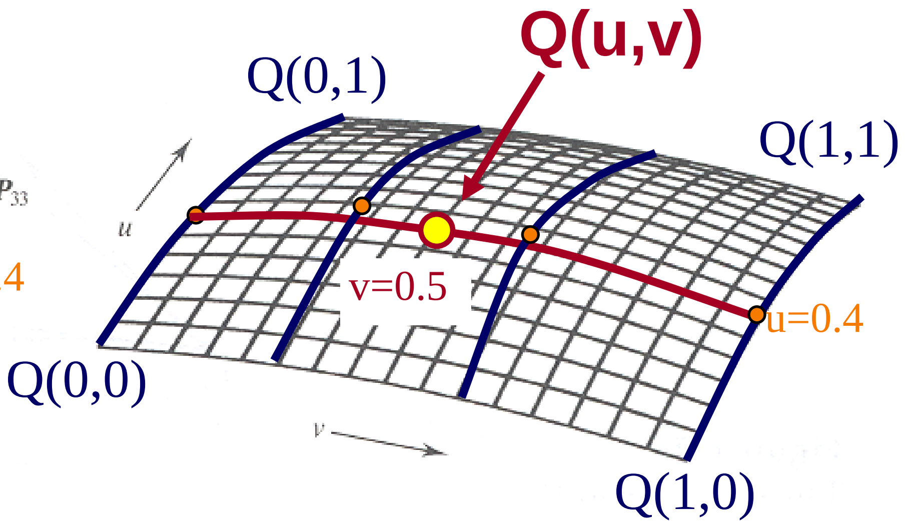


Watt Figure 6.21

Parametric Patches

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





Watt Figure 6.21

Parametric Bicubic Patches

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

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

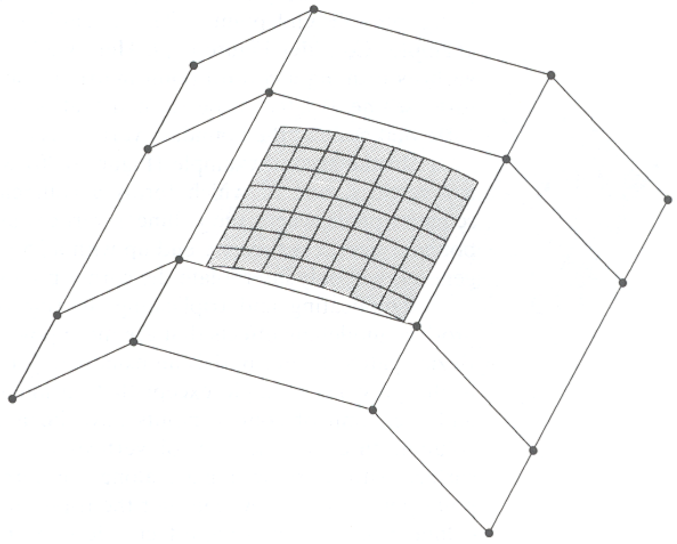
$$\mathbf{U} = [u^3 \quad u^2 \quad u \quad 1] \qquad \mathbf{V} = [v^3 \quad v^2 \quad v \quad 1]$$

Where M is a matrix describing the blending functions for a parametric cubic curve (e.g., Bezier, B-spline, etc.)

B-Spline Patches

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

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

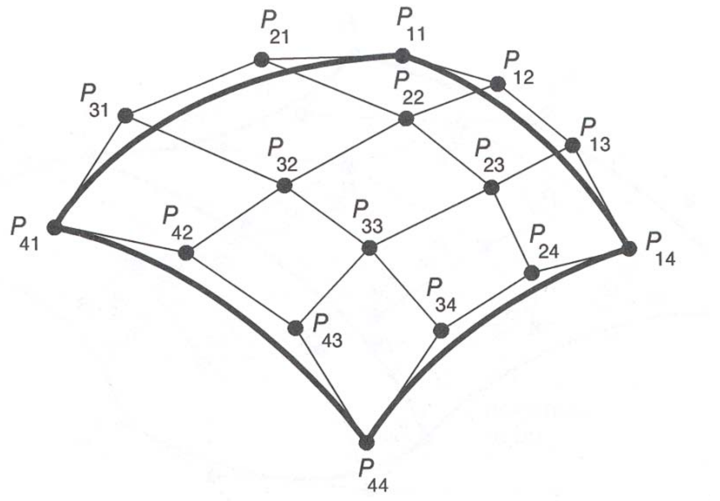


Watt Figure 6.28

Bezier Patches

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

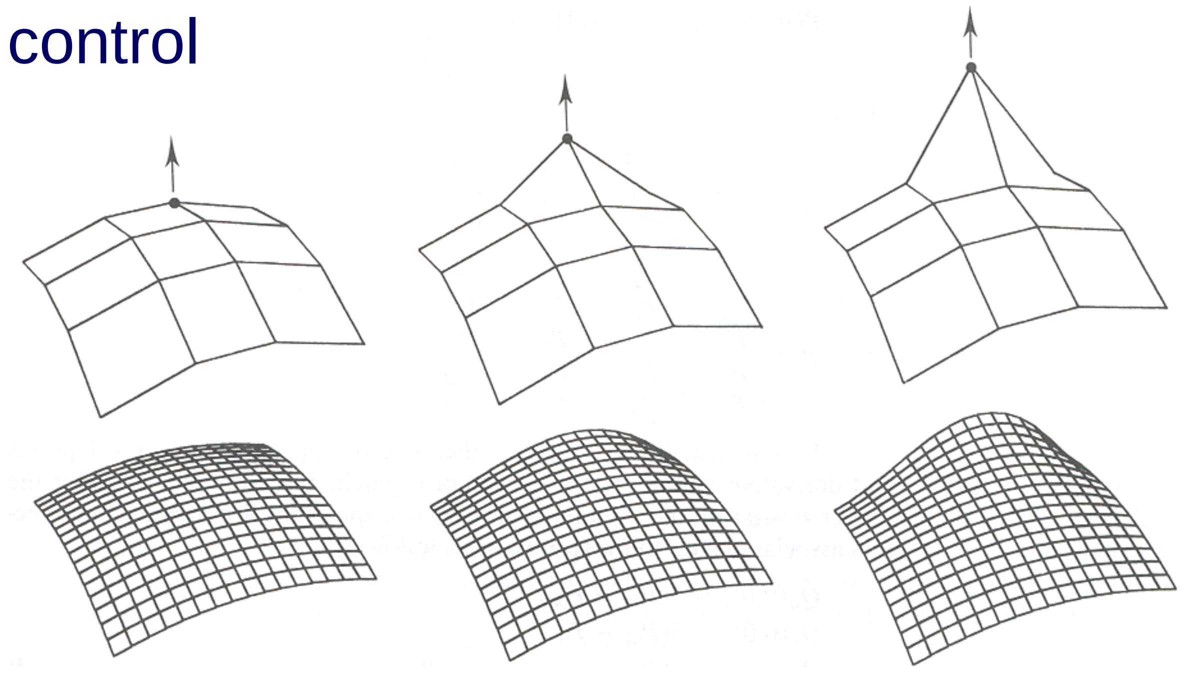
$$\mathbf{M}_{\mathbf{Bezier}} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$



FvDFH Figure 11.42

Bezier Patches

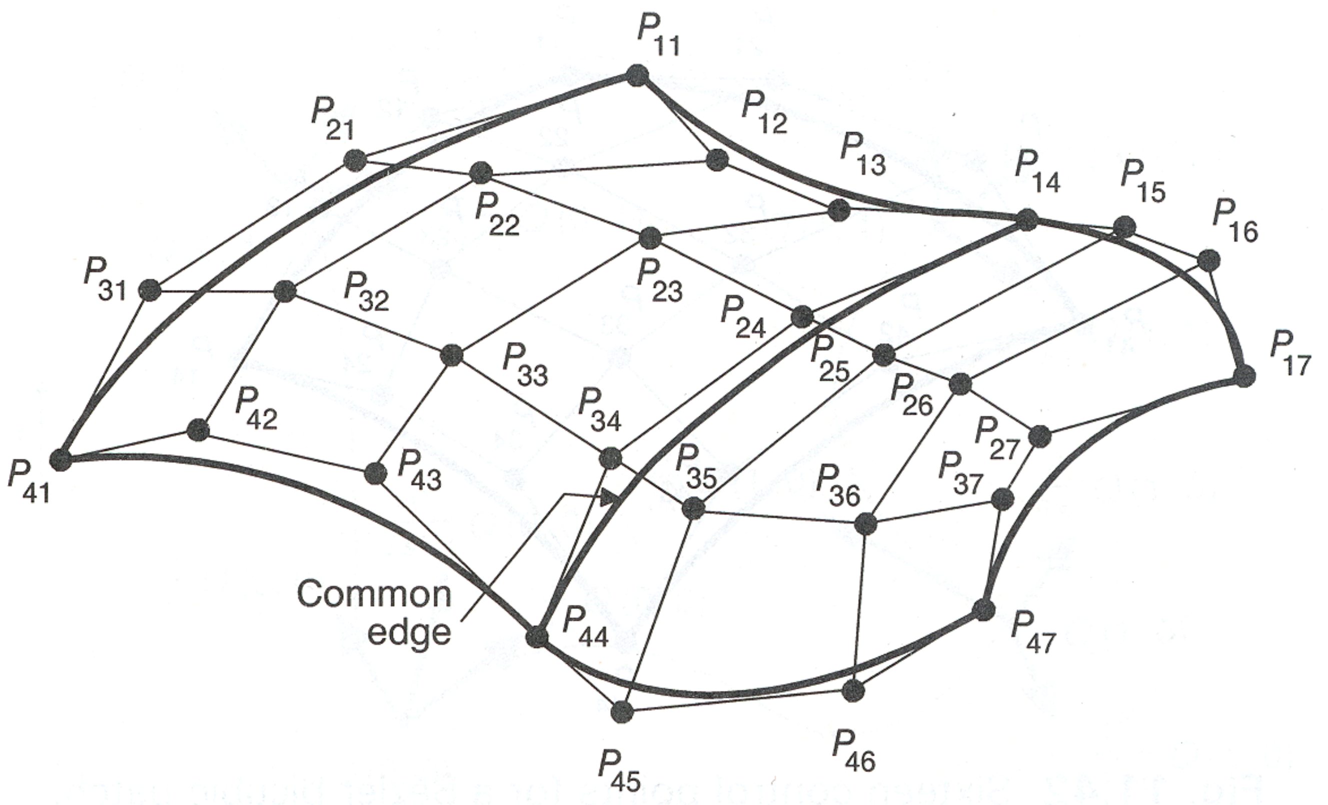
- Properties:
  - Interpolates four corner points
  - Convex hull
  - Local control



Watt Figure 6.22

Bezier Surfaces

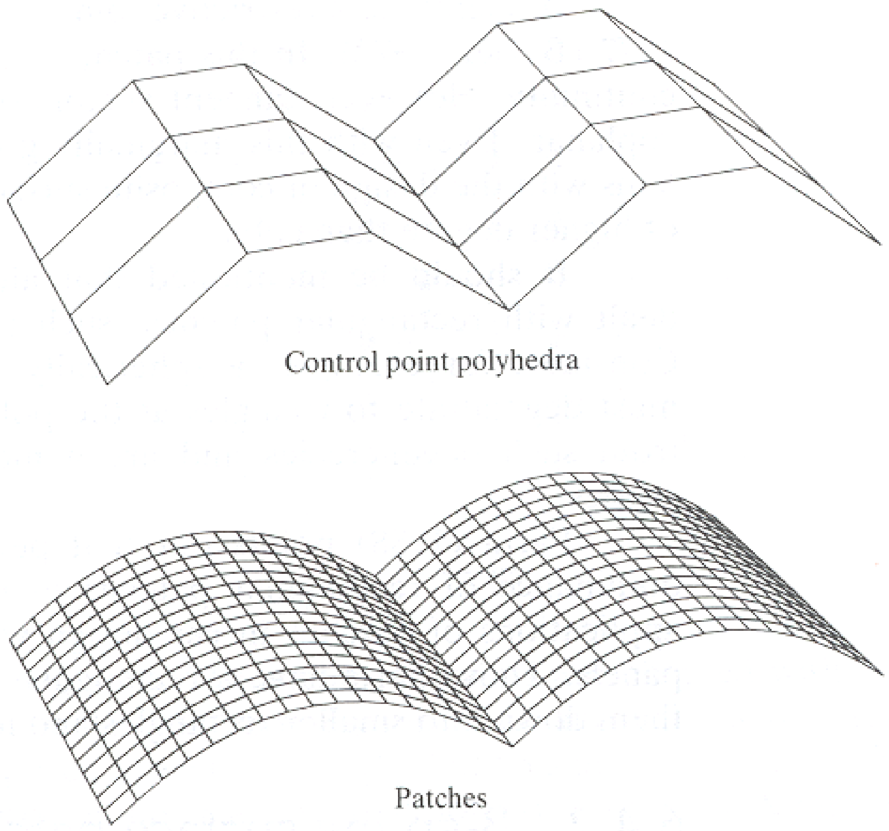
- Continuity constraints are similar to the ones for Bezier splines



FvDFH Figure 11.43

Bezier Surfaces

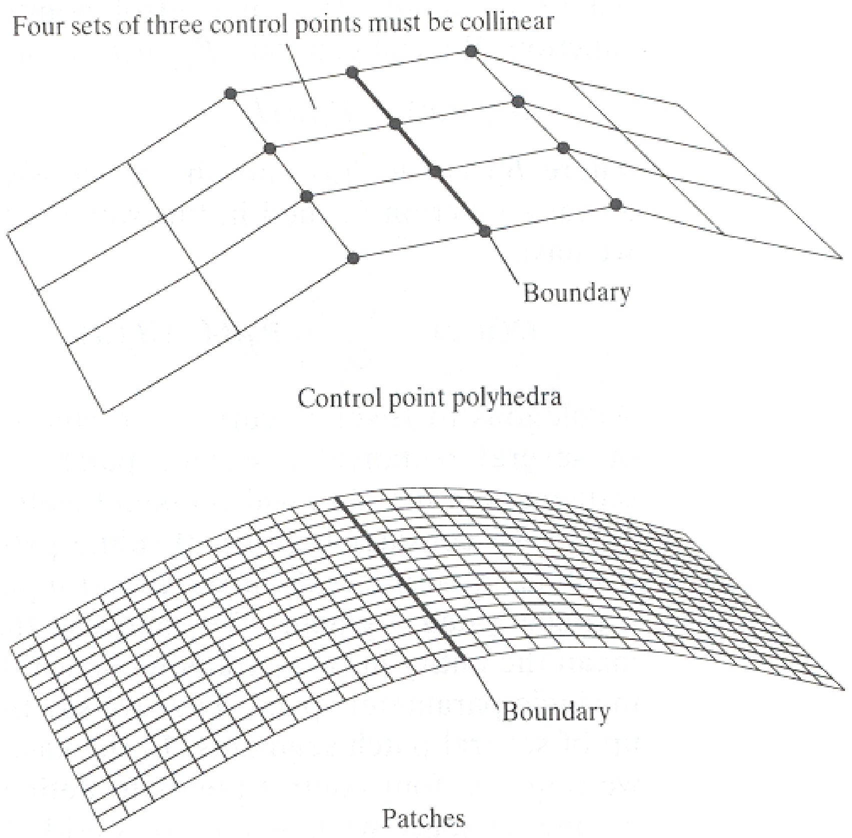
- C^0 continuity requires aligning boundary curves



Watt Figure 6.26a

Bezier Surfaces

- C^1 continuity requires aligning boundary curves and derivatives



Watt Figure 6.26b

Parametric Surfaces

- Advantages:
  - Easy to enumerate points on surface
  - Possible to describe complex shapes
- Disadvantages:
  - Control mesh must be quadrilaterals
  - Continuity constraints difficult to maintain:
    C^0 easy, C^1 possible, C^2 hard at extraordinary vertices
  - Hard to find intersections

Comparison

Feature	Polygonal Mesh	Parametric Surface	Subdivision Surface
Accurate	No	Yes	Yes
Concise	No	Yes	Yes
Intuitive specification	No	Yes	No
Local support	Yes	Yes	Yes
Affine invariant	Yes	Yes	Yes
Arbitrary topology	Yes	No	Yes
Guaranteed continuity	No	Yes	Yes
Natural parameterization	No	Yes	No
Efficient display	Yes	Yes	Yes
Efficient intersections	No	No	No