Advanced Computer Graphics
CS 563: Curves and Curved Surfaces

William DiSanto

Computer Science Dept.

Worcester Polytechnic Institute (WPI)

Overview

- Advantages/Disadvantages
- Implicit Surfaces
- Parametric Paths
- Parametric Surfaces
- NURBS
- Demos

Why Use Curves?

- Compact Representation
  - Control Points, Knots, Weights
- Usually Scale/Rotation/Translation Invariant
- Smooth well Defined Derivatives
  - Good for paths
- Curves can represent triangle mesh exactly
- Support surfaces of arbitrary dimension

Disadvantages to Curved Surfaces

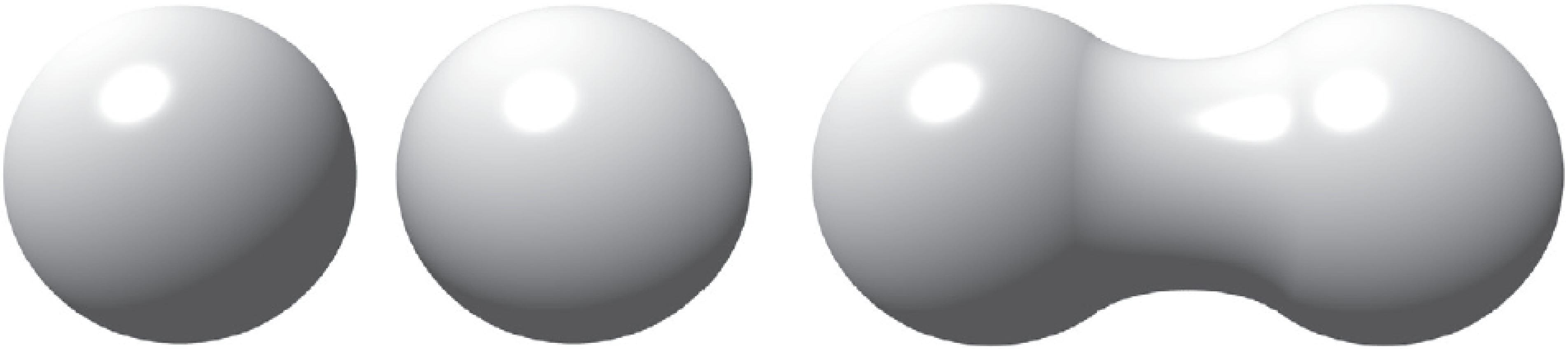
- Can be difficult to implement
- Expensive to render
- Hard to fit
- Few SW packages support diverse features needed
- Often reduced to tri-mesh anyway
- Subdivision surfaces may suffice
- Extraordinarily unintuitive to manipulate
  - Hard for artists to edit in a precise manner
  - Some artists prefer since its reliably scalable (scan in)

Implicit Surfaces

$$f(x, y, z) = x^2 + y^2 + z^2 - 1$$

$$f(p) = \sum_{i=0}^{n-1} h(r_i)$$

$$h(r) = \left(1 - \frac{r^2}{R^2}\right)^3$$

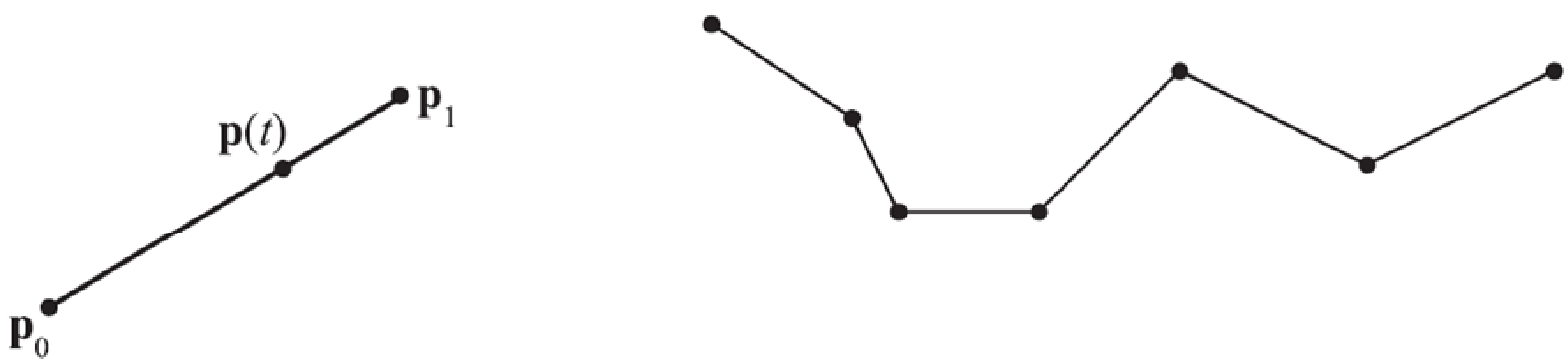


Blending

Parametric Curves

- Each coordinate is expressed as and explicit function of some independent parameters

$$p(t) = p_0 + t(p_1 - p_0) = (1 - t)\, p_0 + p_1$$



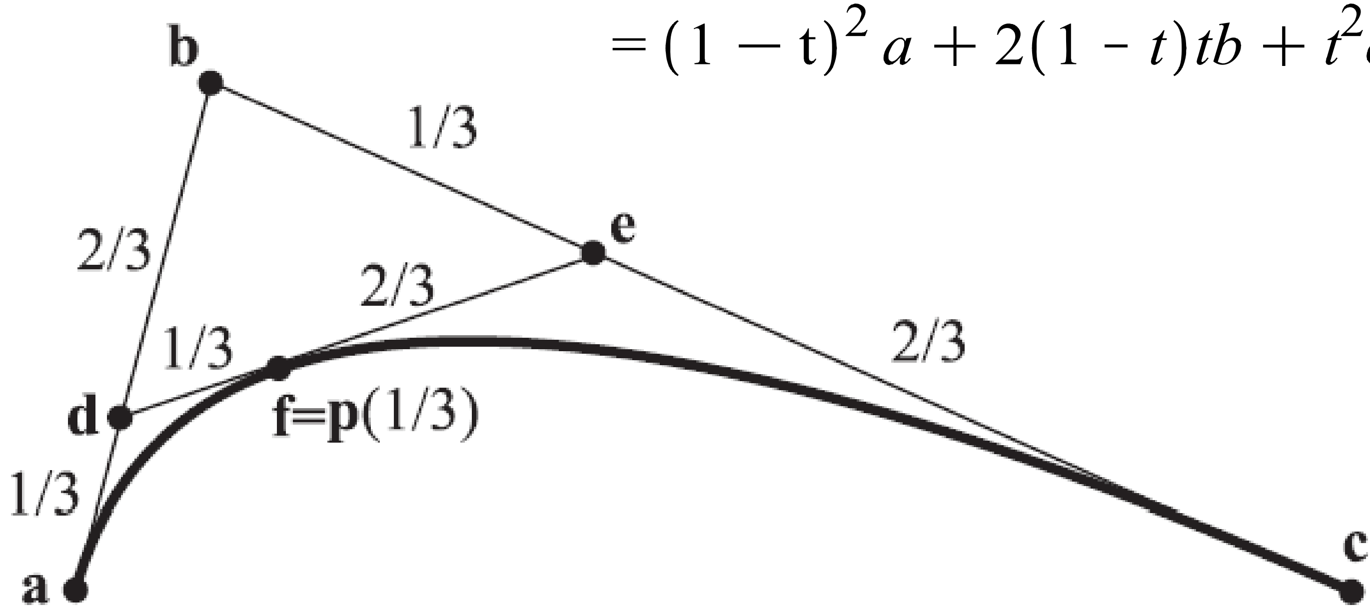
Parametric Linear Curve

Bézier Curves

- Curves from repeatedly linear interpolation on control points
- Curve will have continuity = # of control points – 2
- Can be expressed as a recurrence over control points

$$p(t) = (1-t)d + te \qquad = (1-t)[(1-t)a + tb] + t[(1-t)b + tc]$$

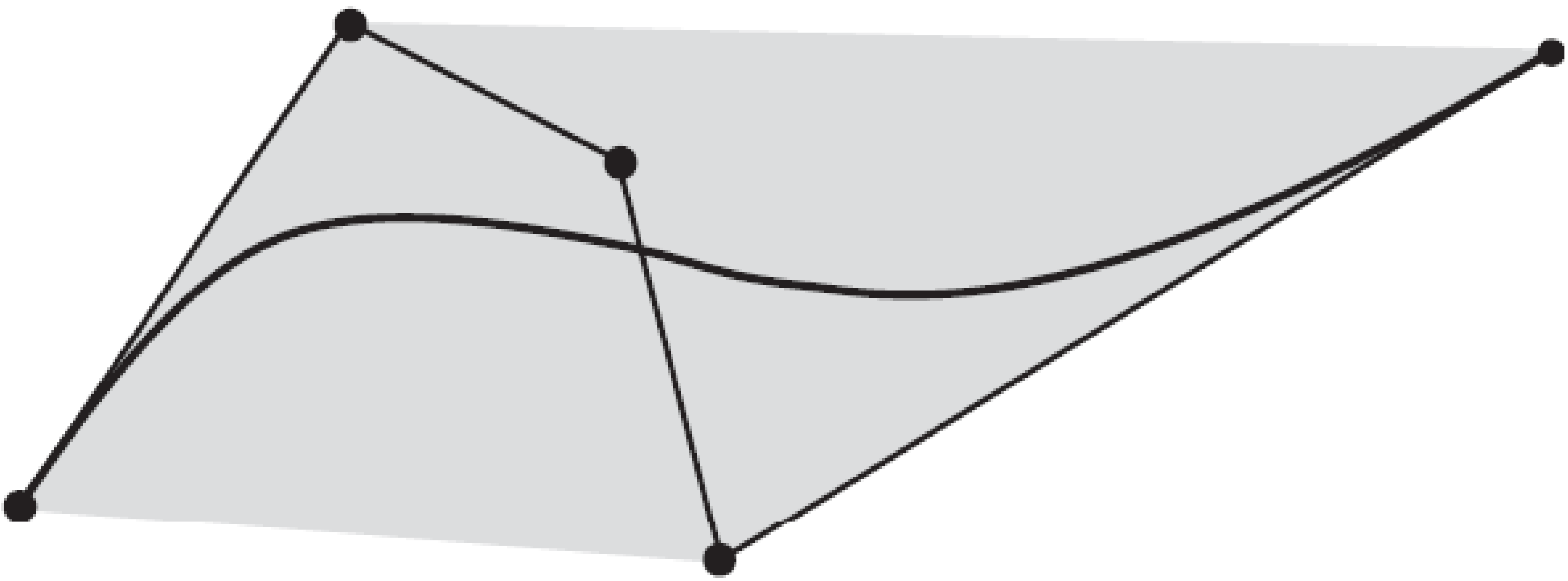
$$= (1-t)^2 a + 2(1-t)tb + t^2 c$$



Quadratic Curve from Control Points (a,b,c), t = 1/3

Bézier Curves

- Curves tend to remain in the convex hull of the control points
- Notice the entire curve is effected by every single control point
  - Except for position at t = 0.0 and t = 1.0

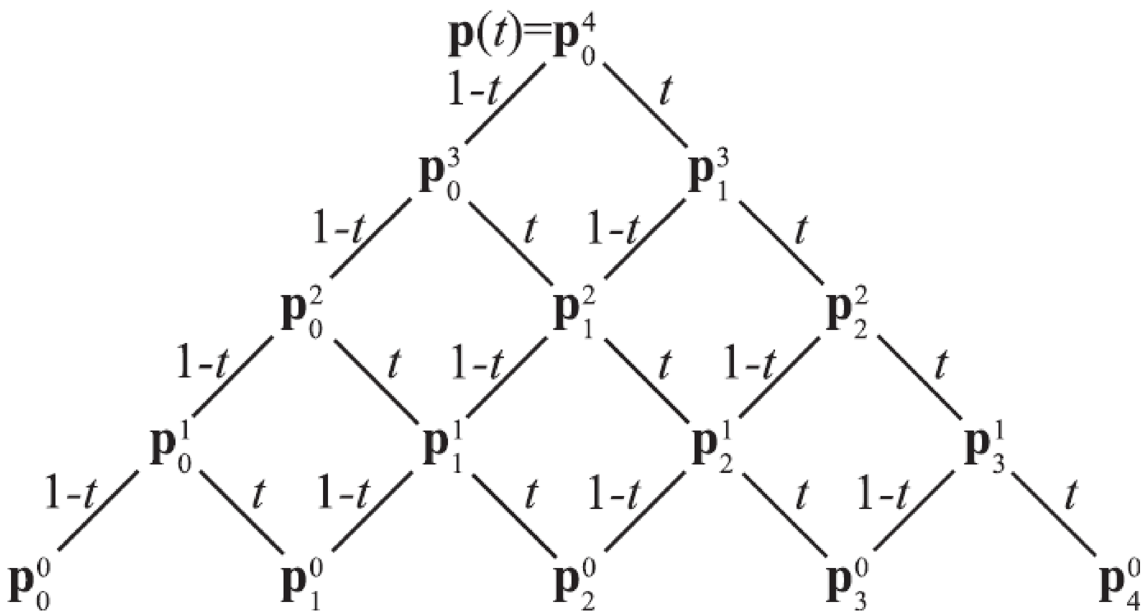


Cubic Bézier Curve

de Casteljau Algorithm

- Generate a tree of linear interpolation points originating from the point on the curve at time t

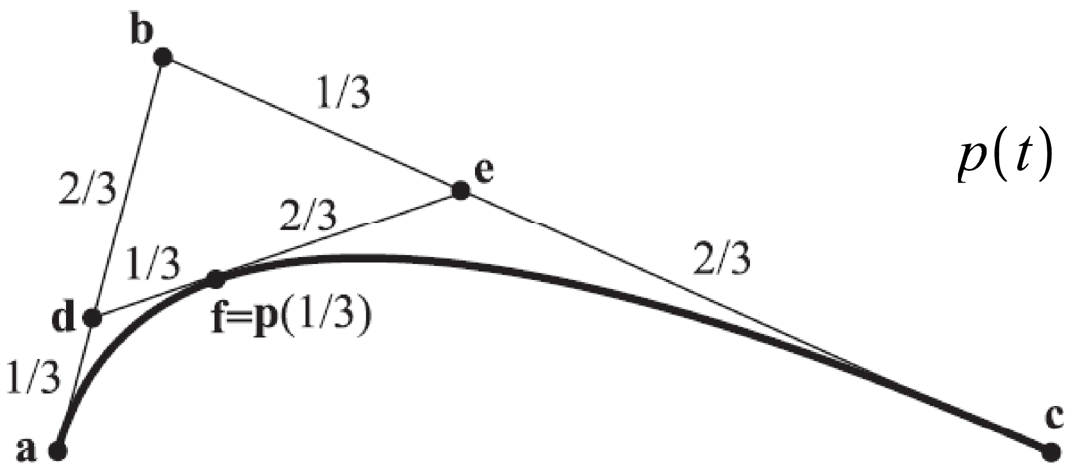
$$p_i^k(t) = (1-t)p_i^{k-1}(t) + t\,p_{i+1}^{k-1}(t) \quad \{\mathbf{k} = 1 \dots n, i = 0 \dots n-k\}$$



Bernstein Polynomials

- Algebraic Form for Bézier Curve

$$p(t) = \sum_{i=0}^{n} B_i^n(t)\, p_i \qquad B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$$



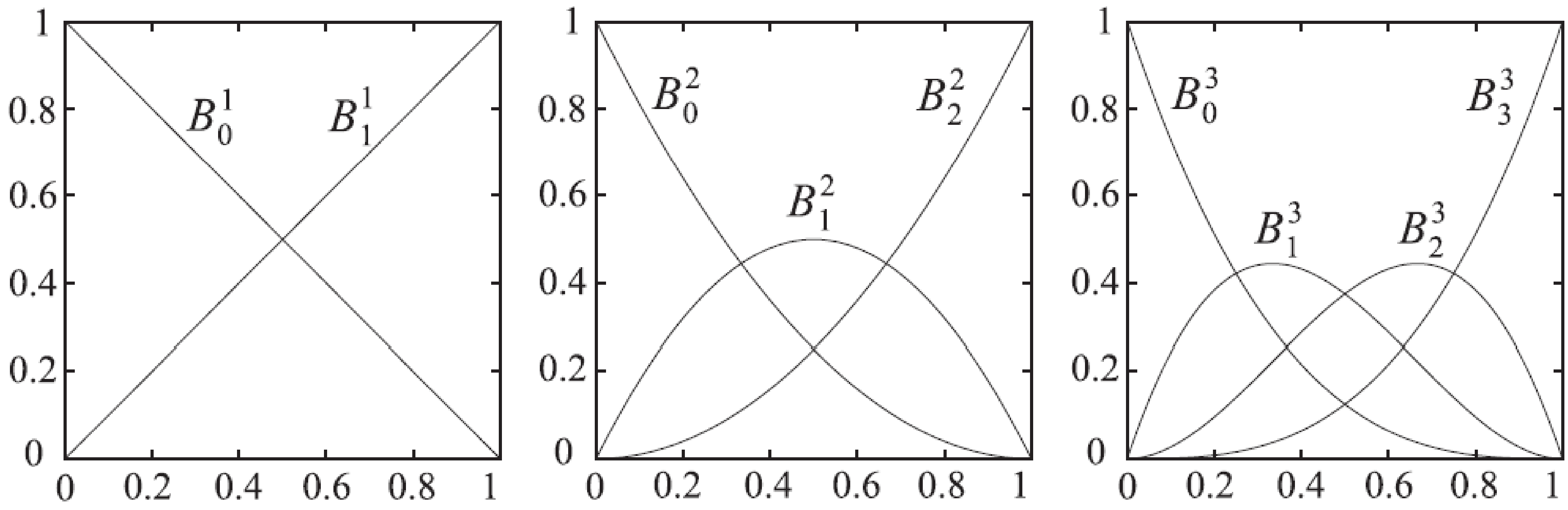
$$p(t) = B_0^2 p_0 + B_1^2 p_1 + B_2^2 p_2$$
$$= (1-\mathrm{t})^2 a + 2(1-t)tb + t^2 c$$

Same as before

Bernstein Polynomials

- Polynomials can be pre-computed and used later
- Binomial coefficient may make solution unstable for large numbers of control point

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$$



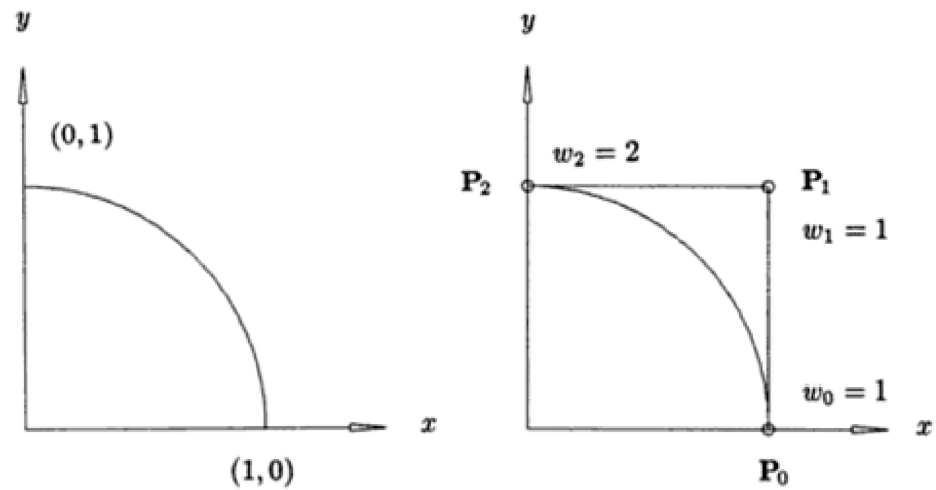
Rational Bézier Curves

- Use weighted ratio of Bernstein Polynomials
- Rational Functions Allow for representation of conic curves
  - Example: Unit Circle

$$x(u) = \frac{1 - u^2}{1 + u^2} \qquad y(u) = \frac{2u}{1 + u^2}$$

Can find weights by substitution

$$p(t) = \frac{\sum_{i=0}^{n} w_i B_i^n(t)\, p_i}{\sum_{i=0}^{n} w_i B_i^n(t)}$$

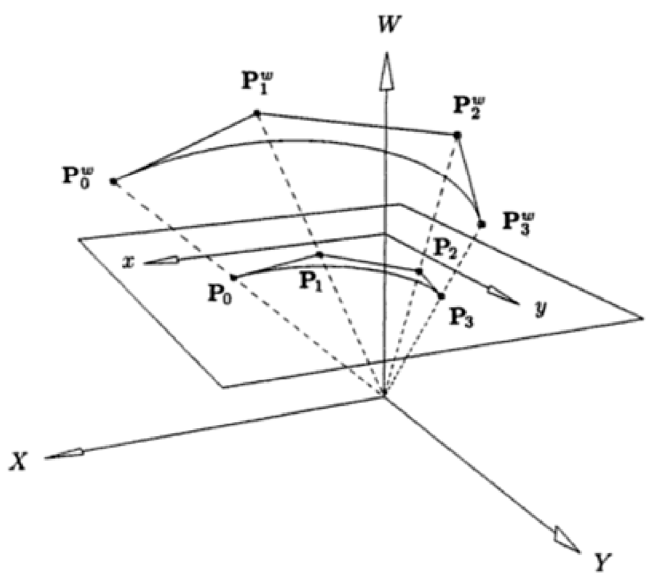


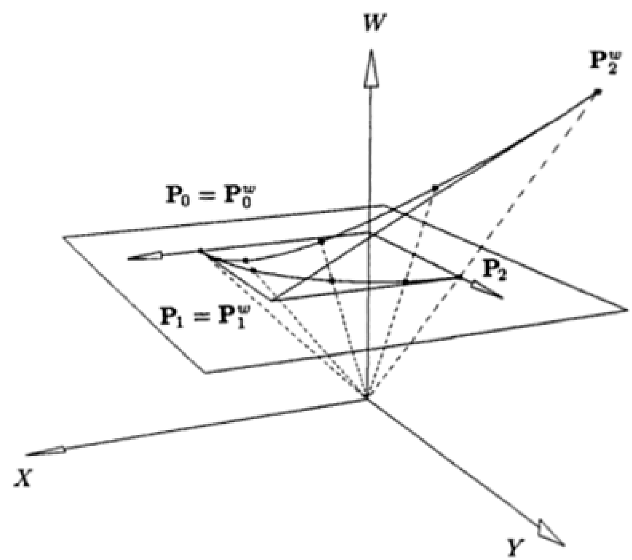
Quadrant of unit circle represented with Bézier Curve

Homogenous Coordinates

- View Rational Polynomial as n-dimensional non-rational polynomial projected into n+1 dimensional space
  - Perspective projection looking down the n+1th dimensional axis
  - Project onto W = 1
- Computationally efficient representation

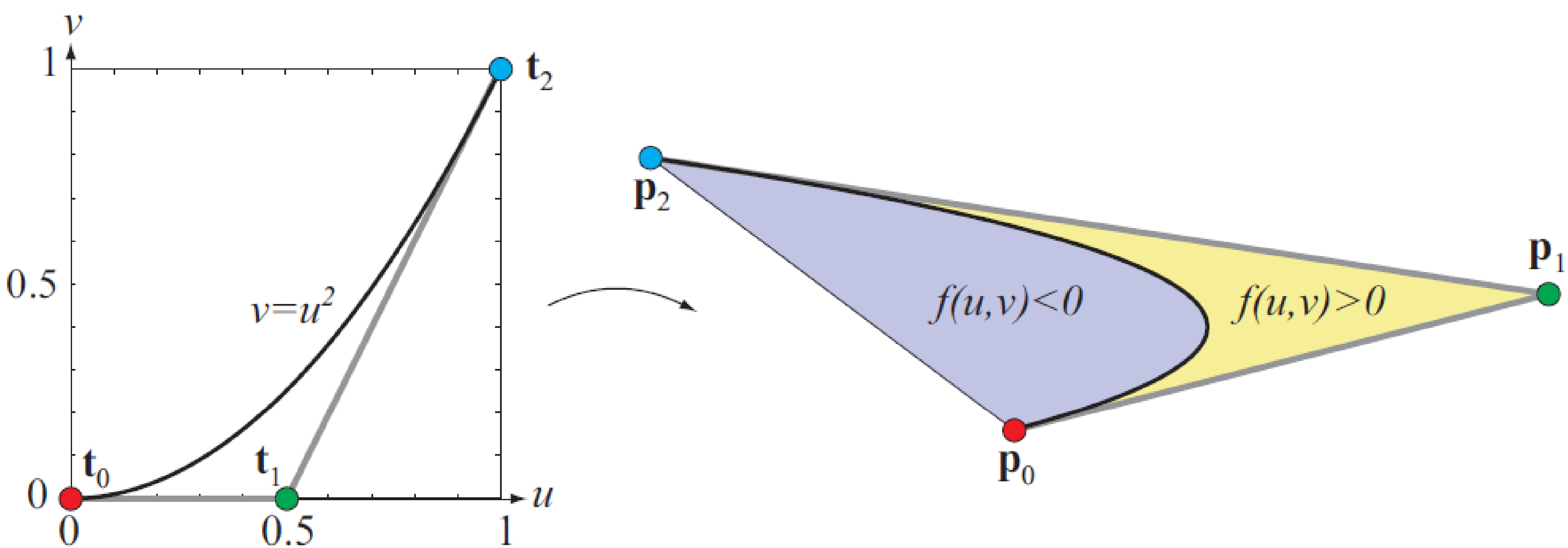
$$X(u) = \sum_{i=0}^{n} B_{i,n}(u) w_i x_i \qquad Y(u) = \sum_{i=0}^{n} B_{i,n}(u) w_i y_i \qquad W(u) = \sum_{i=0}^{n} B_{i,n}(u) w_i$$





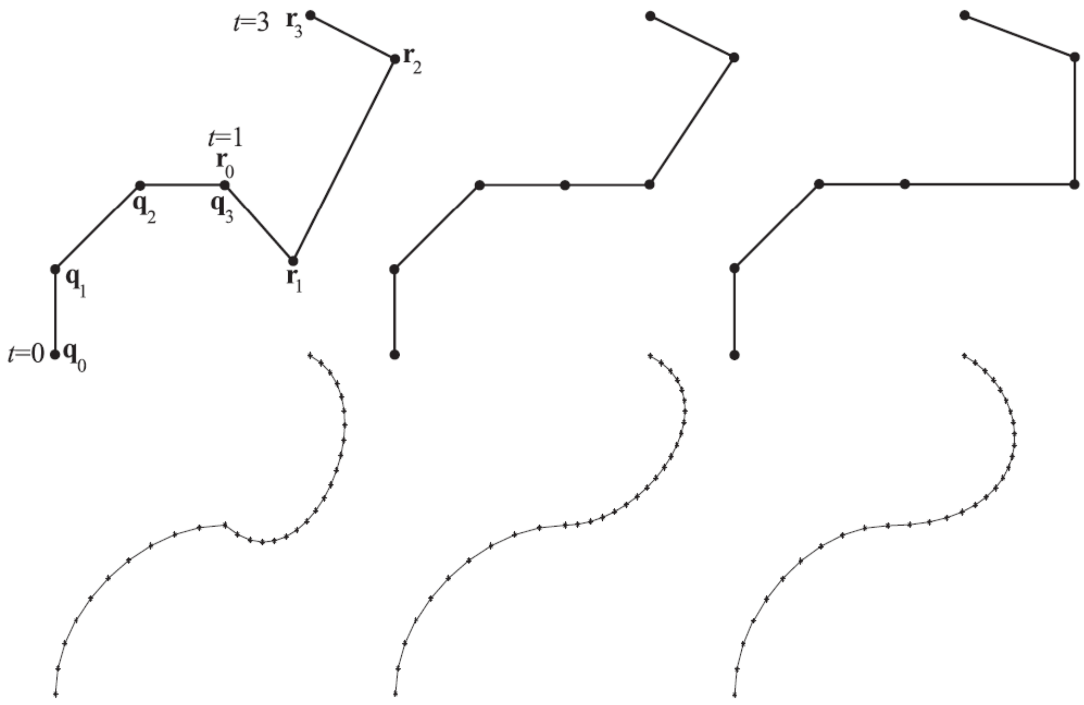
Bézier Curves on GPU (filled)

- Map control points to canonical texture space
- Texture coordinates are interpolated on hardware
- Test per-pixel texture coordinates against algebraic expression of the curve to shade



Piecewise Bézier Curves

- Curves can be joined together
  - Edge control points must match
  - Internal points must be positioned to preserve continuity

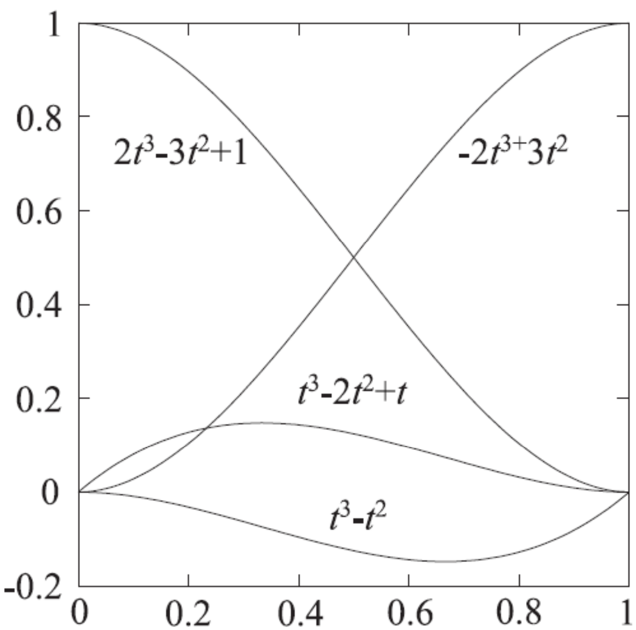


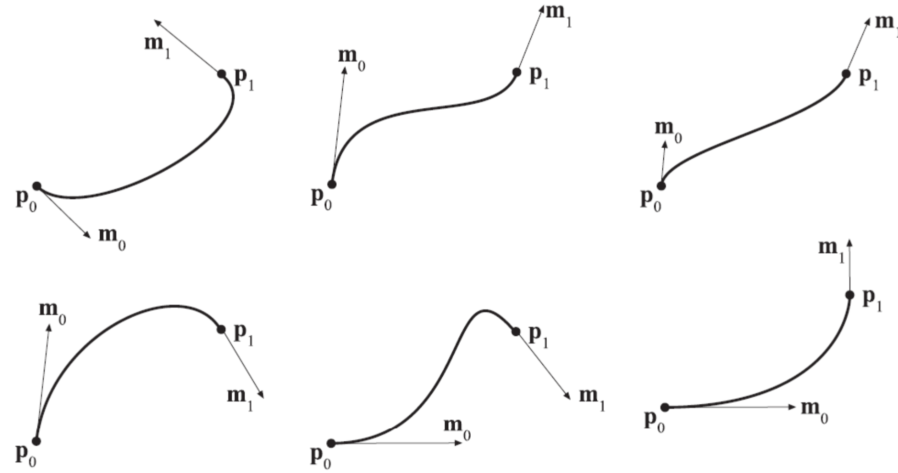
C_0 **G_1 (Tangent DirectionMatch)** **C_1**

Cubic Hermite Interpolation

- Spline controlled by 2 control points and 2 tangents
- In general values and some derivatives at sample points must be known

$$p(t) = (2t^3 - 3t^2 + 1)p_0 + (t^3 - 2t^2 + t)m_0 + (t^3 - t^2)m_1 + (-2t^3 + 3t^2)p_1$$

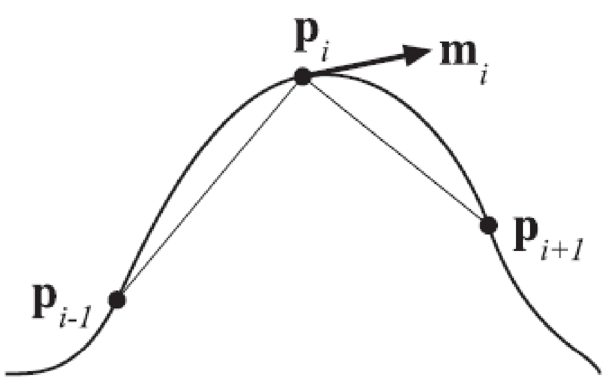


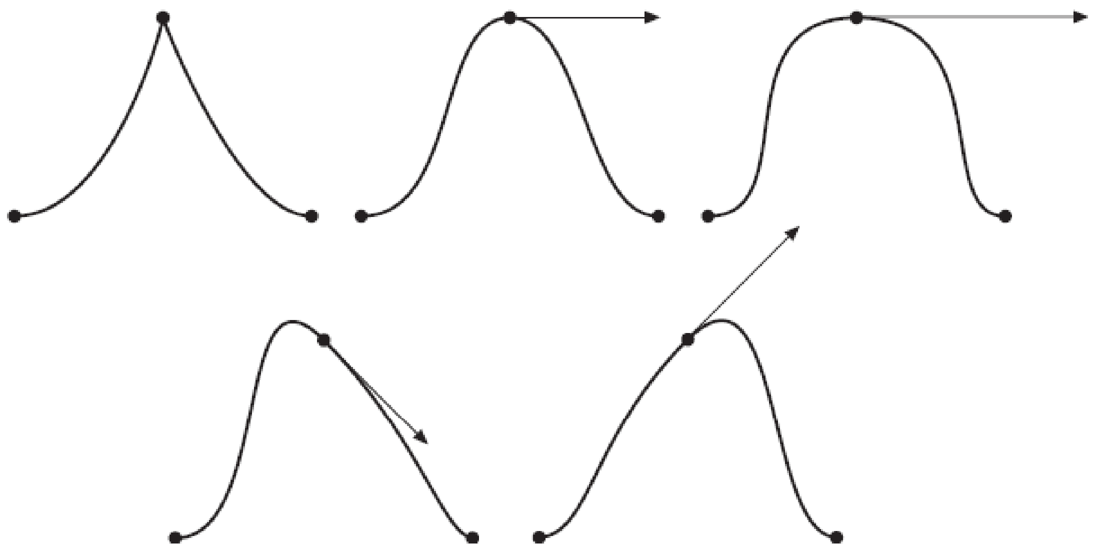


$$\frac{\partial p}{\partial t}(0) = m_0 \qquad \frac{\partial p}{\partial t}(1) = m_1$$

Kochanek-Bartels Curves

- Stitch together Cubic Hermite splines
  - Tension parameter (controls pinching at point)
  - Bias parameter (biases hump before/after i^{th} point)
  - No tension and no bias produces Catmull-Rom spline



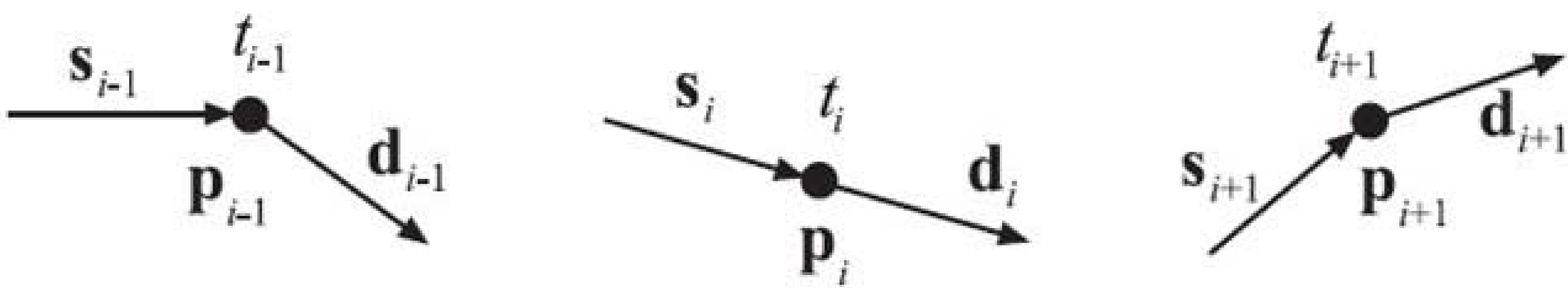


Kochanek-Bartels Curves

- Can define both input and output tangents per point
- Another parameter (continuity)
  - Determines how much in/out tangents agree
  - Can be used to make sharp corners

$$m_i = \frac{(1-a)(1+b)(1-c)}{2}\left(p_i - p_{i-1}\right) + \frac{(1-a)(1-b)(1+c)}{2}\left(p_{i+1} - p_i\right)$$

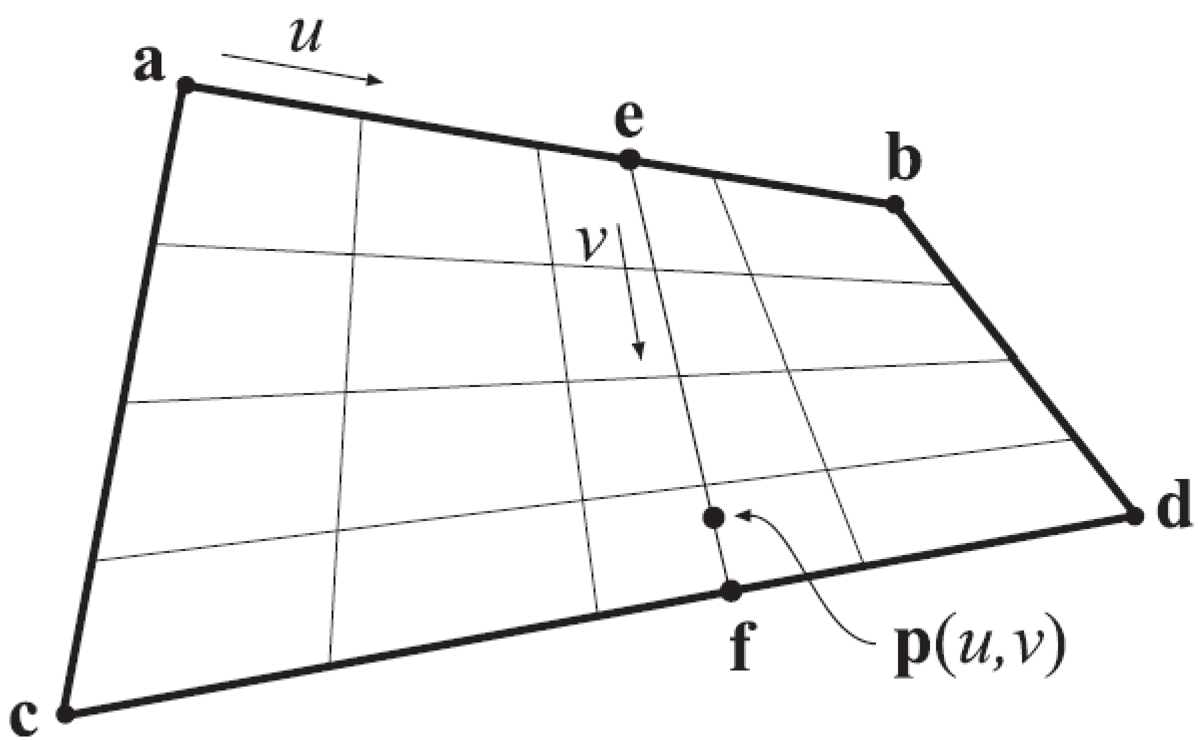
a = tension **b = bias** **c = continuity**



Bézier Patches

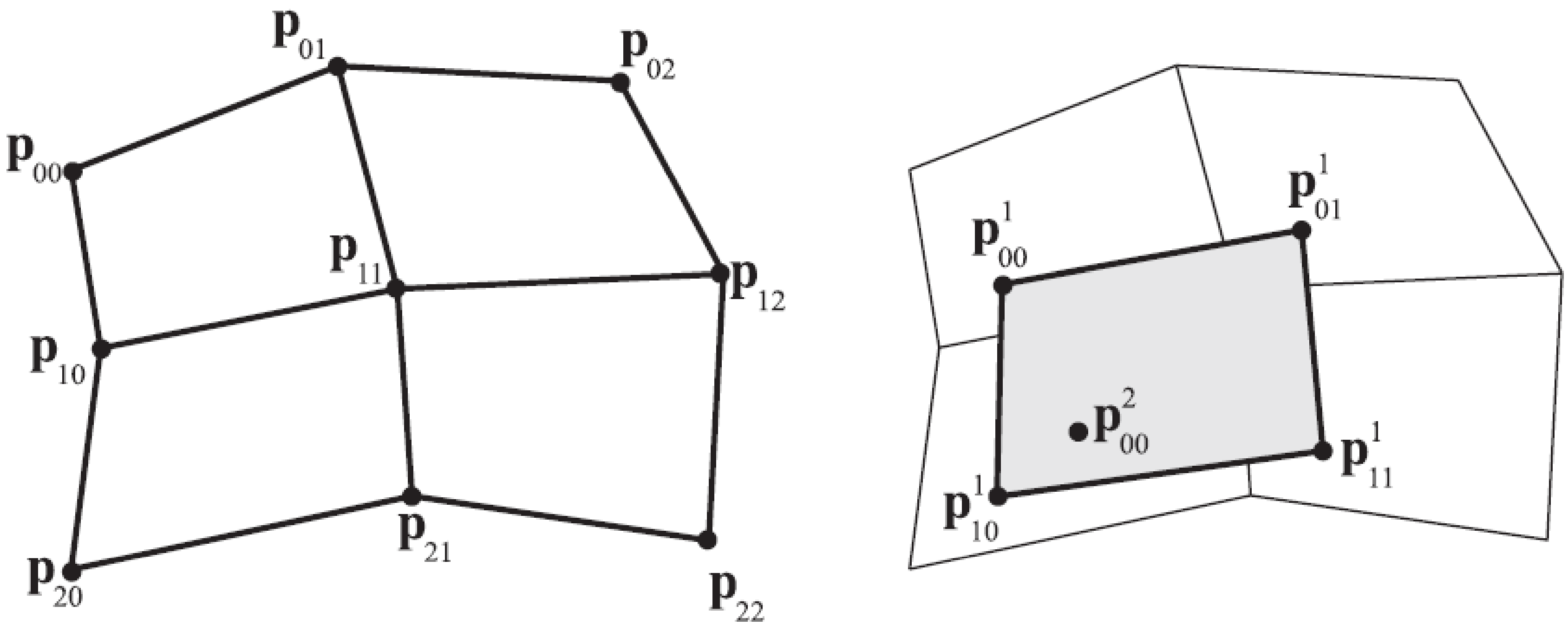
- Bi-linearly Interpolate between control points

$$p(u, v) = (1-u)(1-v)a + u(1-v)b + (1-u)v\,c + uv\,d$$



Bézier Patches

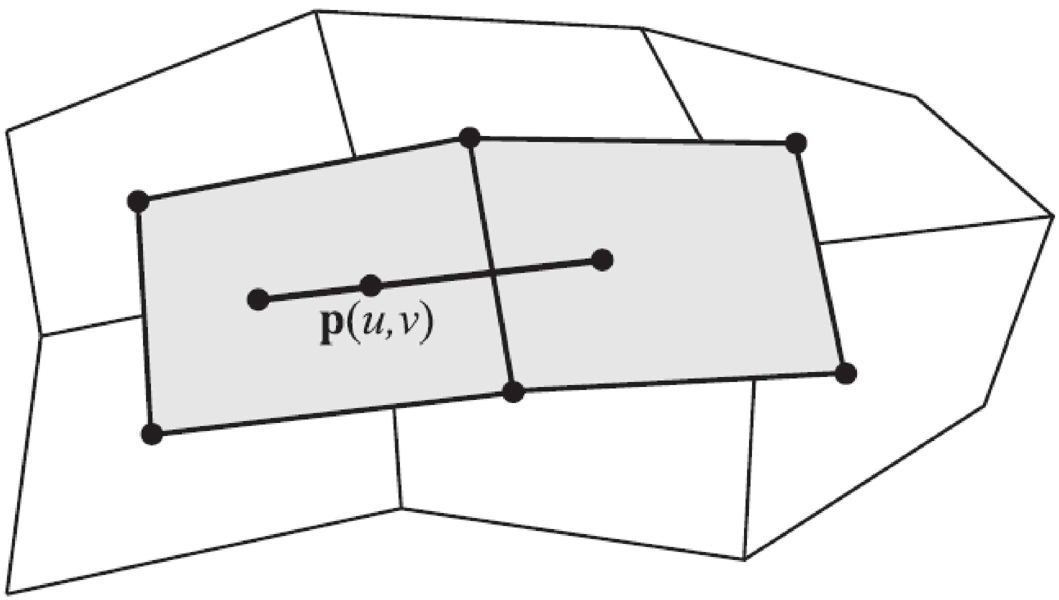
$$p^k_{i,j} = (1-u)(1-v)p^{k-1}_{i,j} + u(1-v)p^{k-1}_{i,j+1} + (1-u)v\,p^{k-1}_{i+1,j} + uv\,p^{k-1}_{i+1,j+1}$$



Generate Interpolation Points within Quads

Bernstein Patches

$$p(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} B_i^m(u) B_j^n(v)\, p_{i,j}$$



Different Degrees in each Dimension Possible

Derivatives

- Derivatives and normals well defined

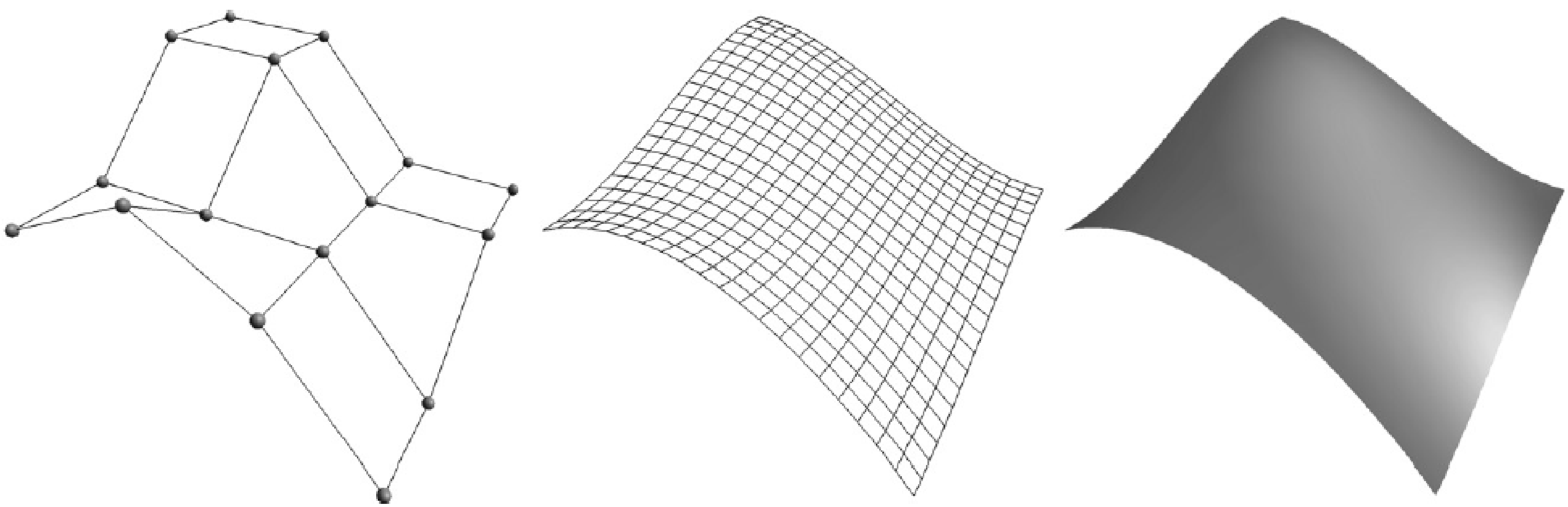
$$\frac{\partial p(u,v)}{\partial u} = m \sum_{j=0}^{n} \sum_{i=0}^{m-1} B_i^{m-1}(u) B_j^n(v) \left[p_{i+1,j} - p_{i,j} \right]$$

$$\frac{\partial p(u,v)}{\partial v} = n \sum_{i=0}^{m} \sum_{j=0}^{\mathrm{n}-1} B_i^{m}(u) B_j^{n-1}(v) \left[p_{i,j+1} - p_{i,j} \right]$$

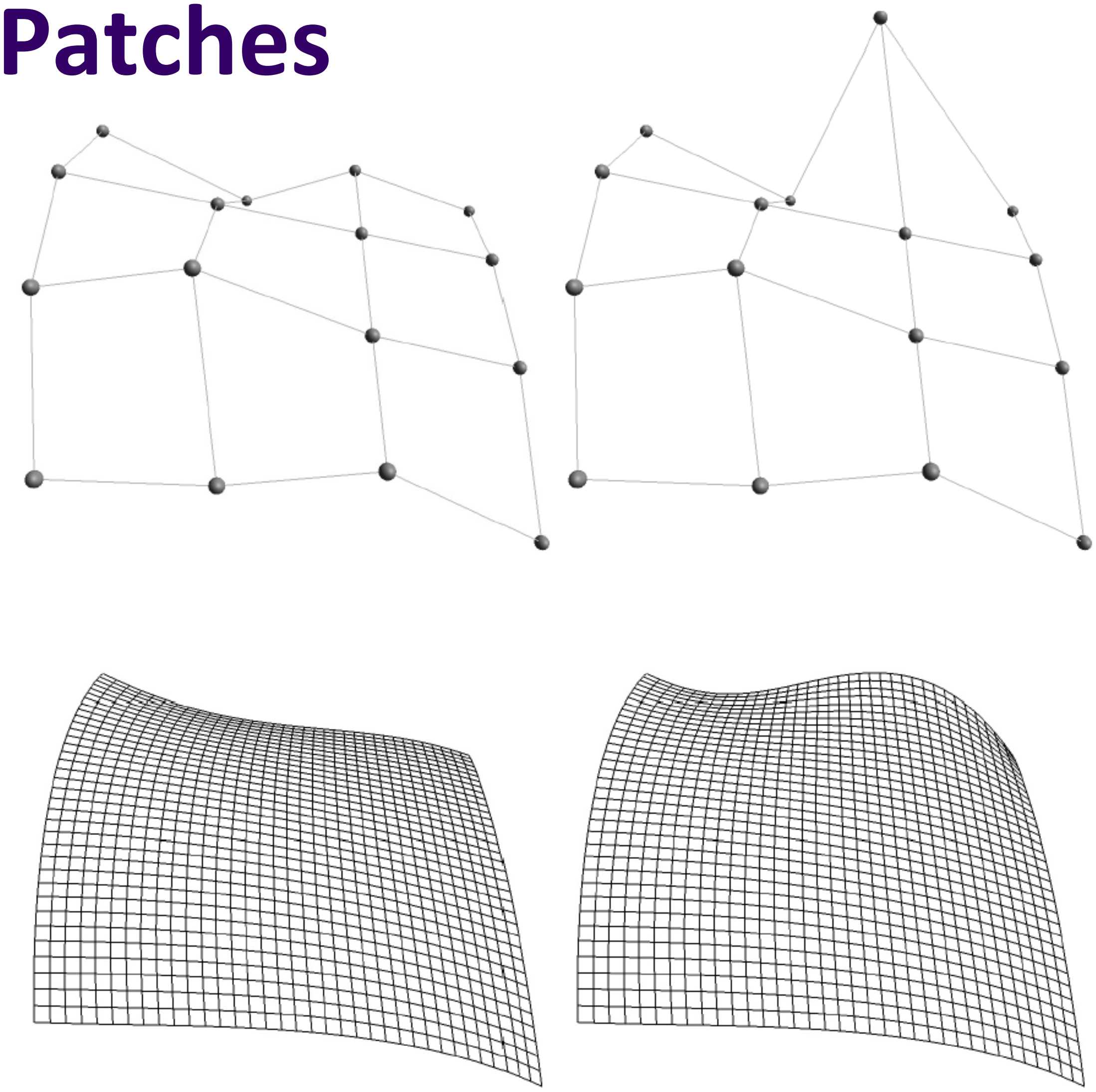
$$n(u,v) = \frac{\partial p(u,v)}{\partial u} \times \frac{\partial p(u,v)}{\partial v}$$

Bézier Patches

- Below: Control points, connected points and normals sampled from the patches, and a render of the computed quads



Bézier Patches

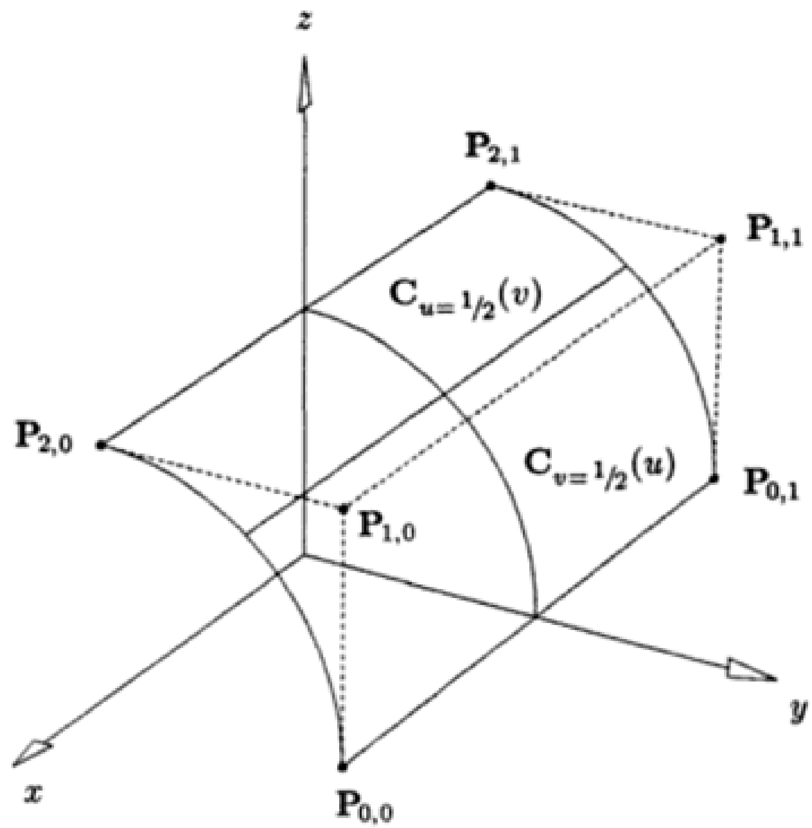


Manipulation of Control Points

Rational Bézier Patches

- Surface still contained within convex hull

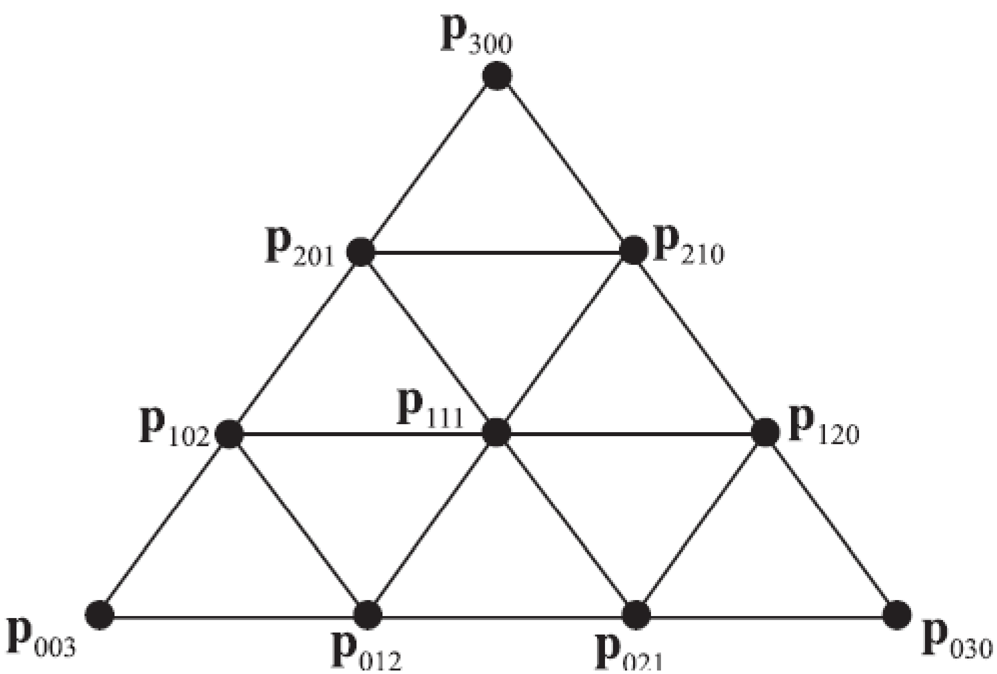
$$p(u,v) = \frac{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{i,j} B_i^m(u) B_j^n(v)\, p_{i,j}}{\sum_{i=0}^{m} \sum_{j=0}^{n} w_{i,j} B_i^m(u) B_j^n(v)}$$



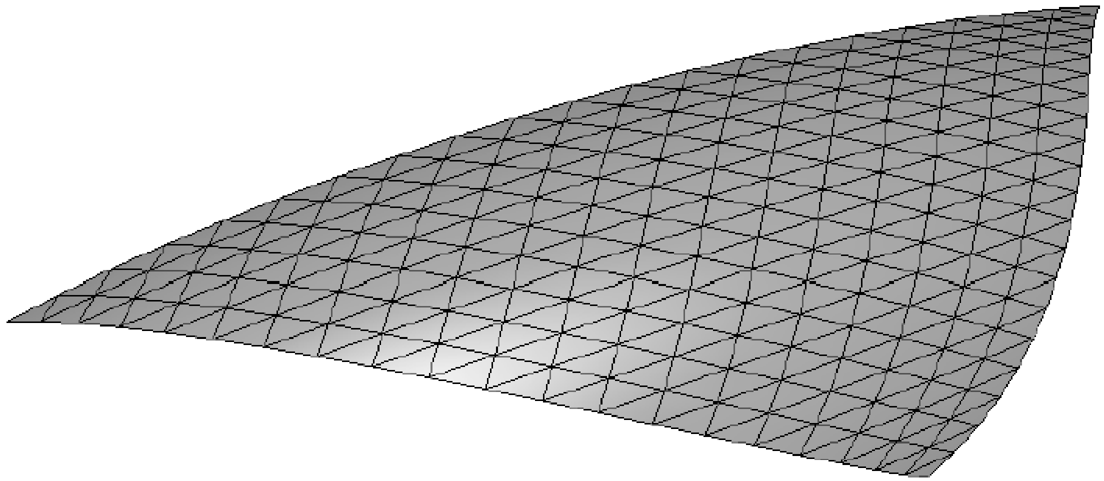
Bézier Triangles

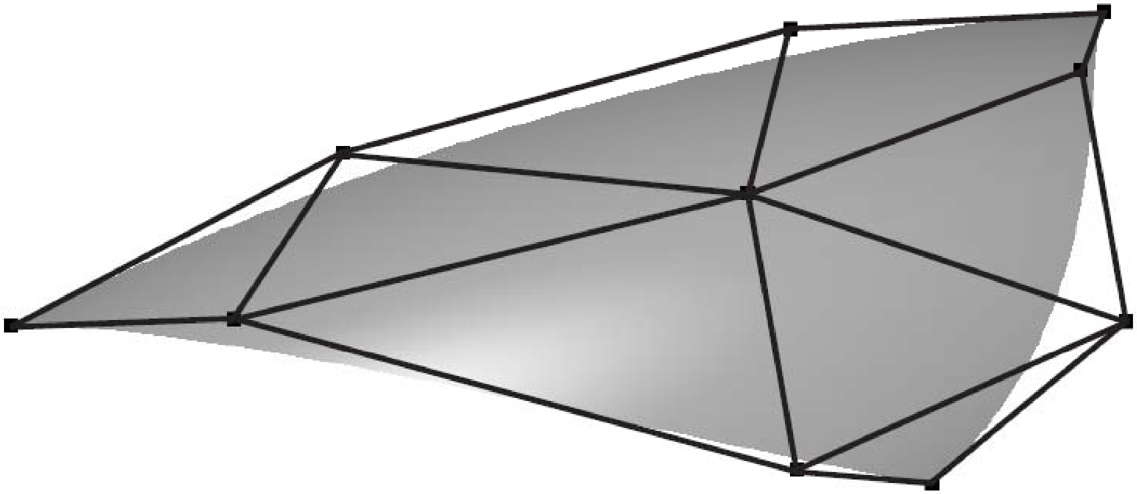
- Interpolate with barycentric coordinates

$$p(u, v) = (1 - u - v)p_0 + up_1 + vp_2$$



Bernstein Triangles





Bézier Triangle: Representations

- de Casteljau

$$p^{l}_{i,j,k}(u,v) = u\,p^{l-1}_{i+1,j,k} + v\,p^{l-1}_{i,j+1,k} + (1-u-v)p^{l-1}_{i,j,k+1}$$

- Bernstein

$$p(u,v) = \sum_{i+j+k=n} B^{n}_{ijk}(u,v)\,p_{ijk}$$

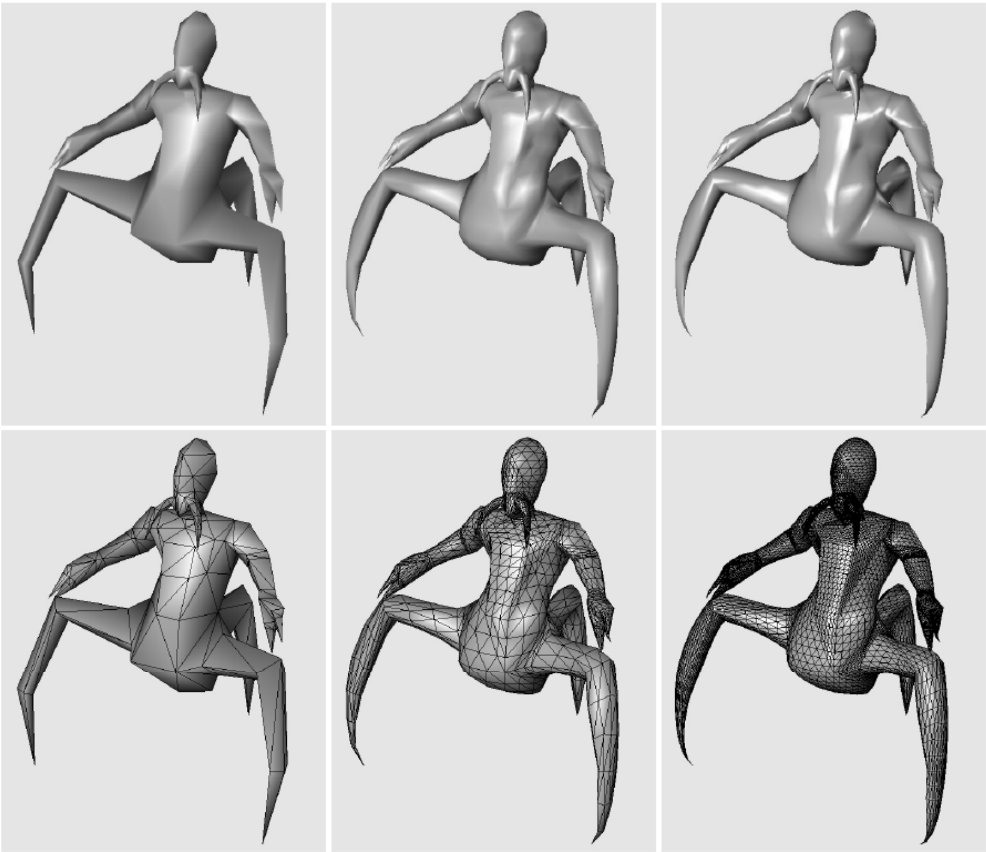
$$B^{n}_{i,j,k}(u,v) = \frac{n!}{i!j!k!}u^{i}v^{j}(1-u-v)^{k} \quad where \quad n = i+j+k$$

- Derivatives

$$\frac{\partial p(u,v)}{\partial u} = \sum_{i+j+k=n-1} B^{n-1}_{ijk}(u,v)\,p_{i+1,j,k} \qquad \frac{\partial p(u,v)}{\partial v} = \sum_{i+j+k=n-1} B^{n-1}_{ijk}(u,v)\,p_{i,j+1,k}$$

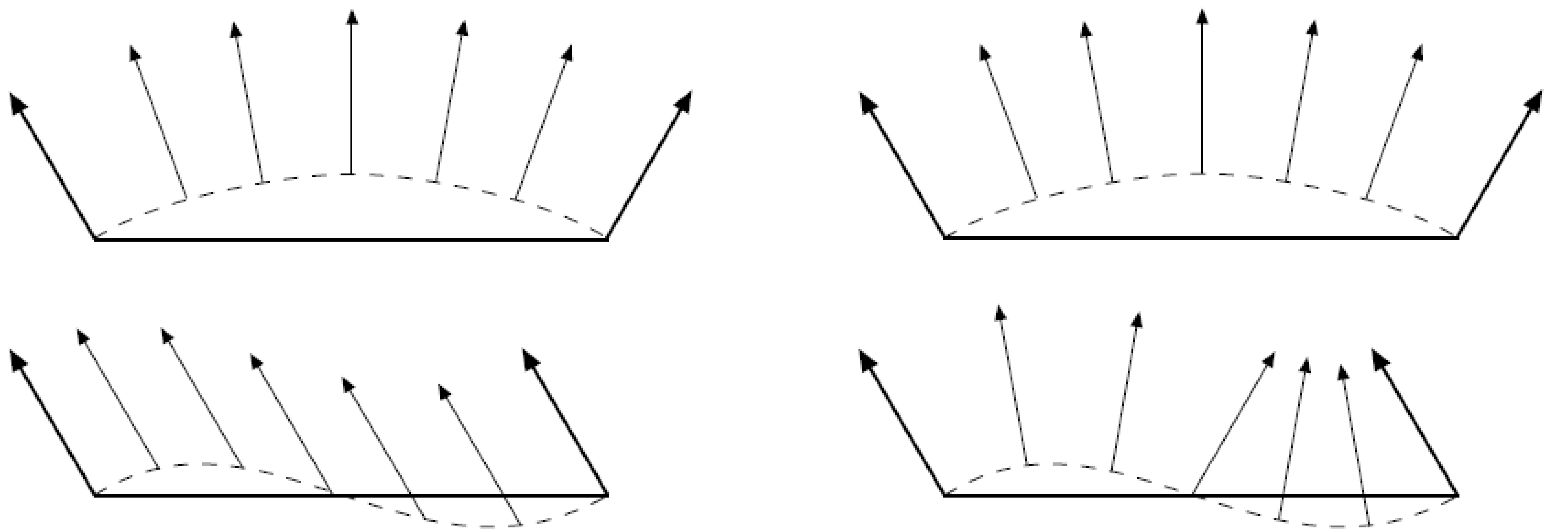
N-Patches

- Generate smooth LOD mesh with N-Patches
  - Each triangle generate 4 internal triangles



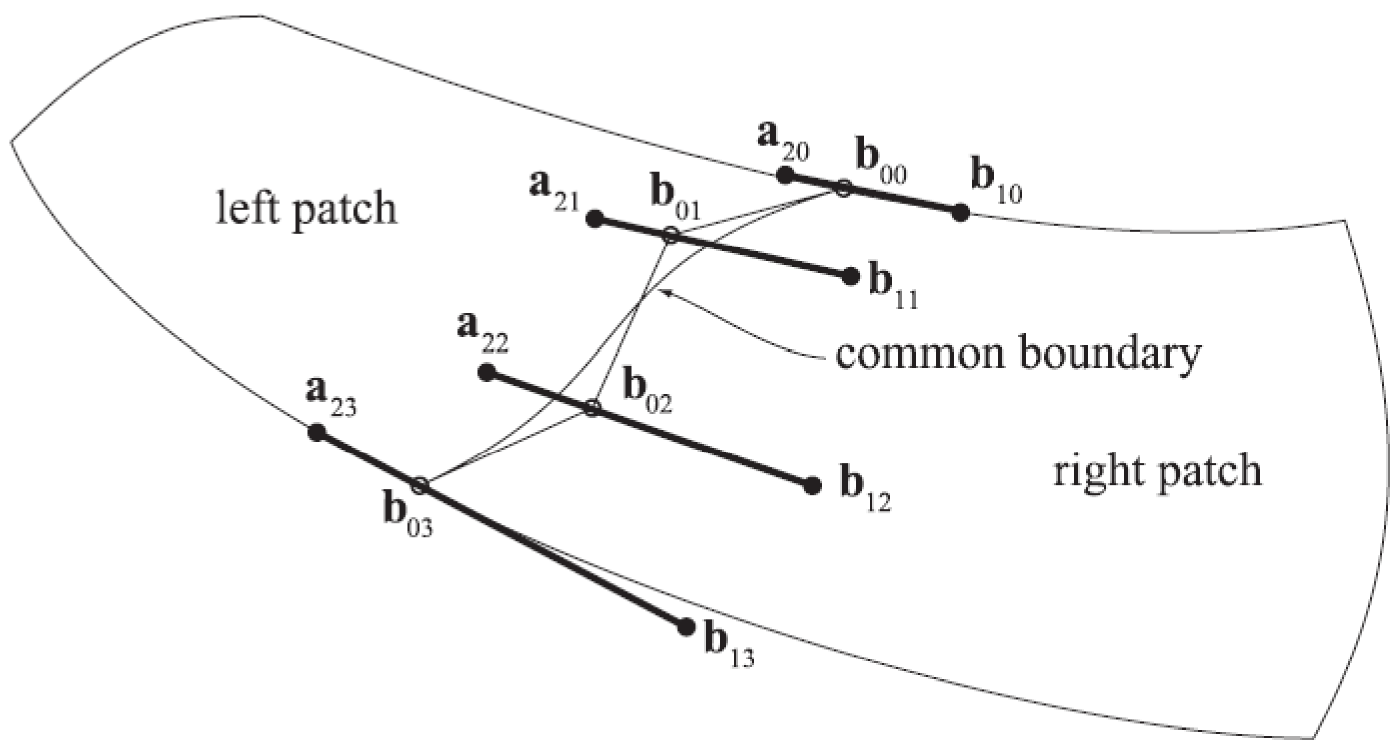
N-Patches

- Quadratic interpolation of normals used to handle inflections

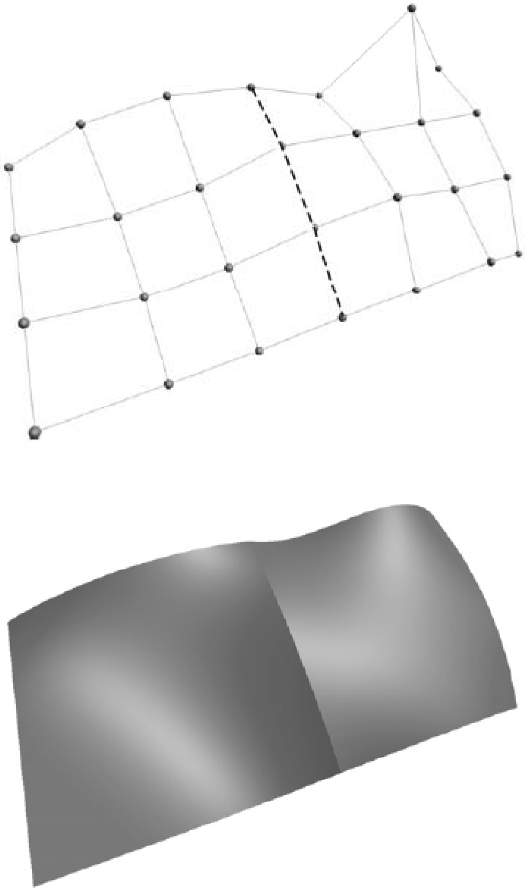


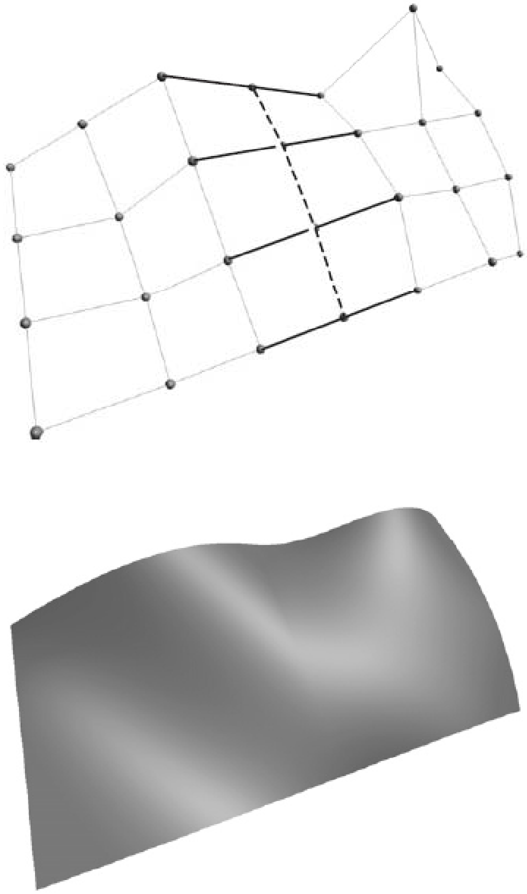
Continuity

- When connecting Bézier patches, points next to the boundary must be collinear to preserve C_1



Continuity

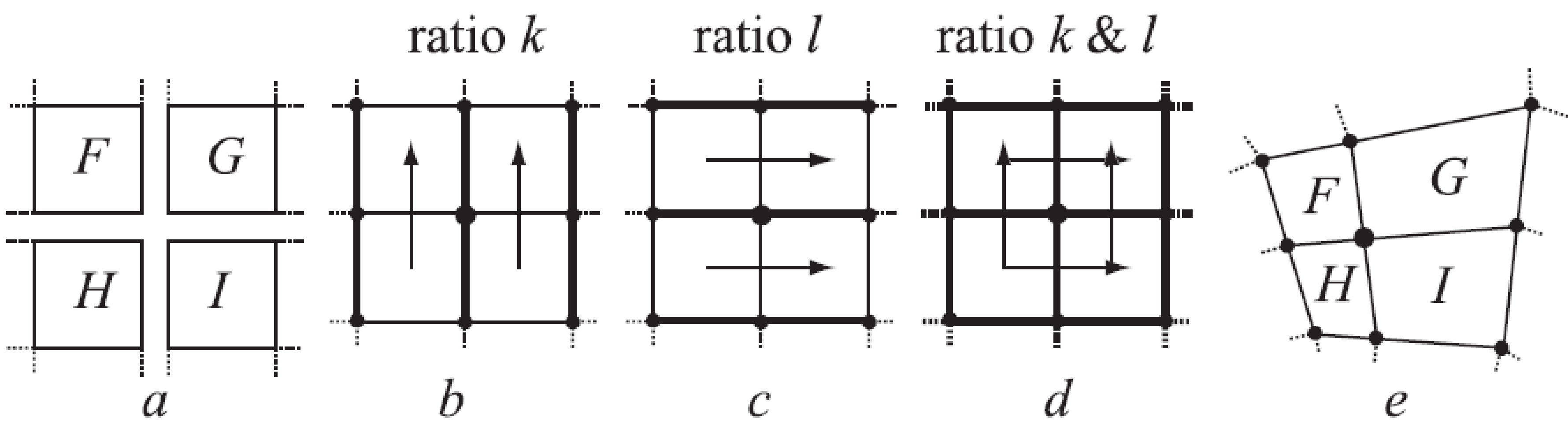




Discontinuous **Continuous**

Continuity

- G_1 continuity if points adjacent to shared corner lie in a plane
- At patch corners vertical and horizontal control points must be spaced at equal rations for C_1



Basis-Splines

- Offer Local Control
- Can be expressed as Bézier curves
- Suppress Error for many control points
- Continuity controlled for any number of points*

$$f(p) = \sum_{i=0}^{m-n-2} p_i N_{i,n}(t)$$

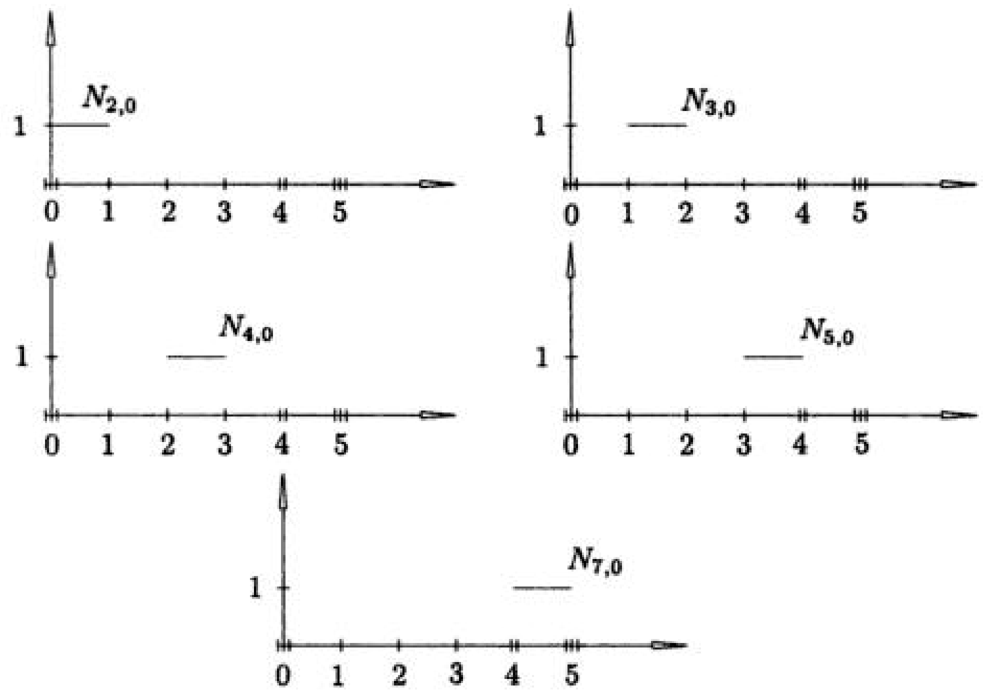
$$n = degree$$
$$m - n - 1 = number\ of\ control\ points$$

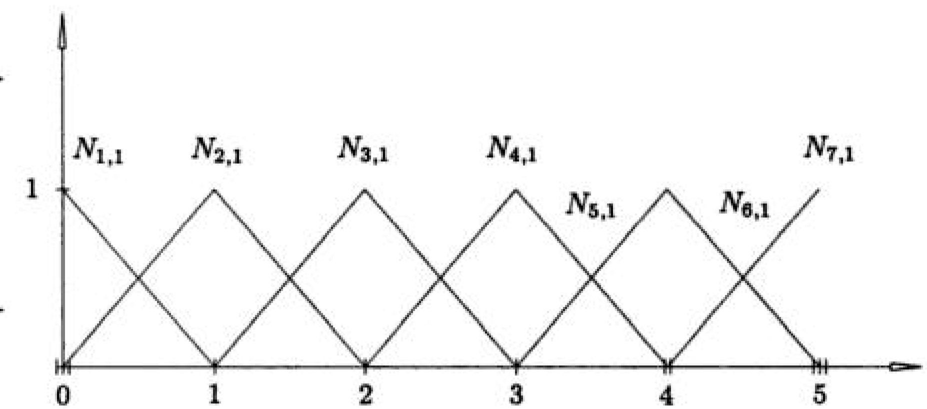
$$N_{j,0}(t) = \begin{cases} 1, & t_j \le t < t_{j+1} \\ \textbf{else}\ 0 \end{cases}$$

$$N_{j,n}(t) = \frac{t - t_j}{t_{j+n} - t_j} N_{j,n-1}(t) + \frac{t_{j+n+1} - t}{t_{j+n+1} - t_{j+1}} N_{j+1,n-1}(t)$$

Knot Vectors

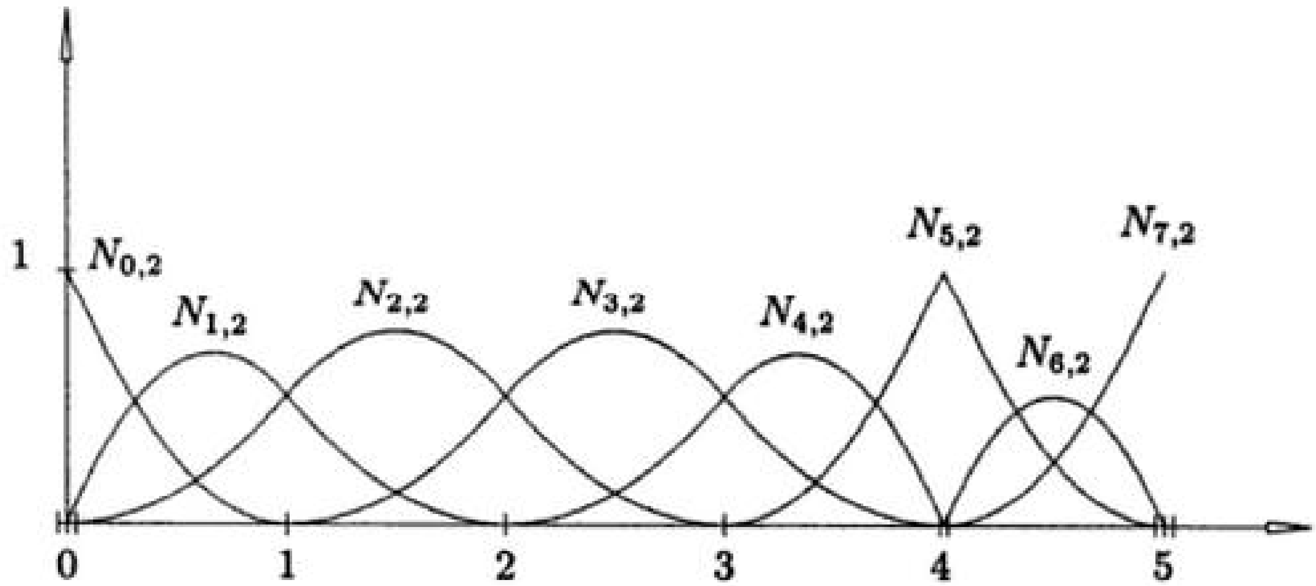
- Describe set of basis functions (from Cox-de Boor)
- Knot values can be repeated to reduce the span of a basis function
- Need not be integer valued





$$U = \{0, 0, 0, 1, 2, 3, 4, 4, 5, 5, 5\}$$

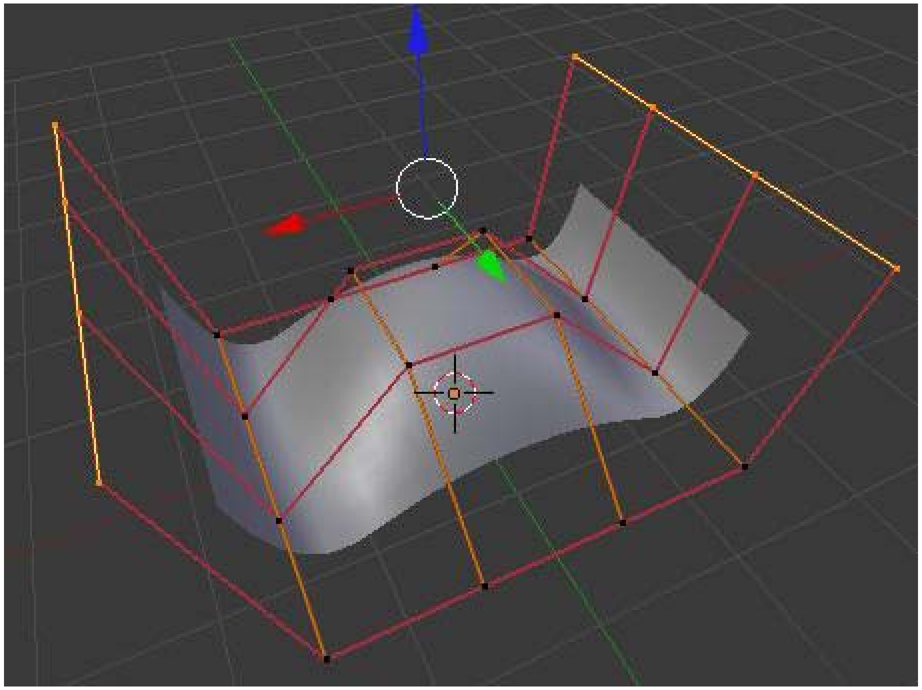
Knot Vectors: Non-Uniform Example



$$U = \{0, 0, 0, 1, 2, 3, 4, 4, 5, 5, 5\}$$

NURBS

- Knot vectors can take on values that are not uniformly spaced
- Use Rational B-Splines



NURBS Texturing

- Map parameters to texture [0,1]
- texcpts = [0, 0, 0, 1, 1, 0, 1, 1]

```
gluBeginSurface(globj);
gluNurbsSurface(globj, 4, U, 4, V, 4, 2, texcpts, 2, 2,
        GL_MAP2_TEXTURE_COORD_2);
gluNurbsSurface(globj, n + p + 1, U, m + q + 1, V, 3
        m, 3, cpts, p + 1, q + 1, GL_MAP2_VERTEX_3);
gluEndSurface(globj);
```

Some Demos

Run Demos

References

- *Real-Time Rendering* by Tomas Akenine-Möller, Eric Haines, and Naty Hoffman, from A.K. Peters Ltd., 3rd edition.
- *L. Piegl and W. Tiller, The NURBS Book (Monographs in Visual Communication), 2nd ed.*
- *NURBS Textures, Peter Salvi, June 30, 2008*