

CSE 167:
Introduction to Computer Graphics
Lecture #13: Bezier Surfaces

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2013

Announcements

- ▶ Homework assignment #6 due Friday at 1:30pm
 - ▶ Last day for late submissions assignment #5
- ▶ Next Monday discussion: midterm #2
- ▶ Next Thursday: midterm #2

Overview

- ▶ Piecewise Bezier curves
- ▶ Bezier surfaces

Global Parameterization

- ▶ Given N curve segments $\mathbf{x}_0(t), \mathbf{x}_1(t), \dots, \mathbf{x}_{N-1}(t)$
- ▶ Each is parameterized for t from 0 to 1
- ▶ Define a piecewise curve
 - ▶ Global parameter u from 0 to N

$$\mathbf{x}(u) = \begin{cases} \mathbf{x}_0(u), & 0 \leq u \leq 1 \\ \mathbf{x}_1(u-1), & 1 \leq u \leq 2 \\ \vdots & \vdots \\ \mathbf{x}_{N-1}(u-(N-1)), & N-1 \leq u \leq N \end{cases}$$

$$\mathbf{x}(u) = \mathbf{x}_i(u-i), \text{ where } i = \lfloor u \rfloor \quad (\text{and } \mathbf{x}(N) = \mathbf{x}_{N-1}(1))$$

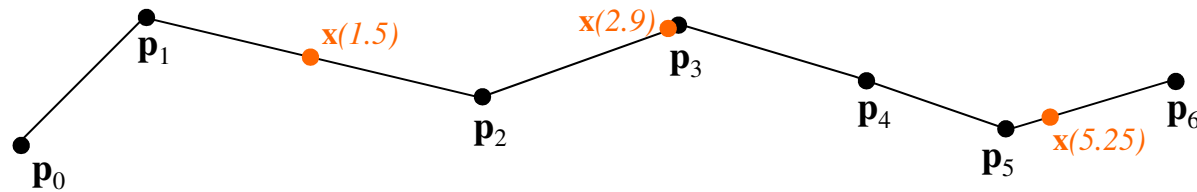
- ▶ Alternate: solution u also goes from 0 to 1

$$\mathbf{x}(u) = \mathbf{x}_i(Nu-i), \text{ where } i = \lfloor Nu \rfloor$$

Piecewise-Linear Curve

- ▶ Given $N+1$ points $\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_N$
- ▶ Define curve

$$\begin{aligned}\mathbf{x}(u) &= \text{Lerp}(u - i, \mathbf{p}_i, \mathbf{p}_{i+1}), & i \leq u \leq i+1 \\ &= (1 - u + i)\mathbf{p}_i + (u - i)\mathbf{p}_{i+1}, & i = \lfloor u \rfloor\end{aligned}$$



- ▶ $N+1$ points define N linear segments
- ▶ $\mathbf{x}(i) = \mathbf{p}_i$
- ▶ C^0 continuous by construction
- ▶ C^1 at \mathbf{p}_i when $\mathbf{p}_i - \mathbf{p}_{i-1} = \mathbf{p}_{i+1} - \mathbf{p}_i$

Piecewise Bézier curve

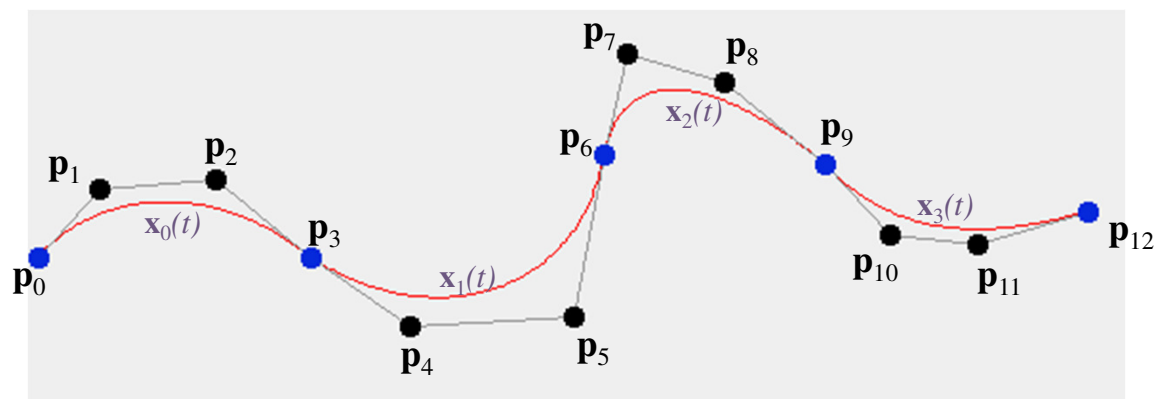
- Given $3N + 1$ points $\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_{3N}$
- Define N Bézier segments:

$$\mathbf{x}_0(t) = B_0(t)\mathbf{p}_0 + B_1(t)\mathbf{p}_1 + B_2(t)\mathbf{p}_2 + B_3(t)\mathbf{p}_3$$

$$\mathbf{x}_1(t) = B_0(t)\mathbf{p}_3 + B_1(t)\mathbf{p}_4 + B_2(t)\mathbf{p}_5 + B_3(t)\mathbf{p}_6$$

\vdots

$$\mathbf{x}_{N-1}(t) = B_0(t)\mathbf{p}_{3N-3} + B_1(t)\mathbf{p}_{3N-2} + B_2(t)\mathbf{p}_{3N-1} + B_3(t)\mathbf{p}_{3N}$$

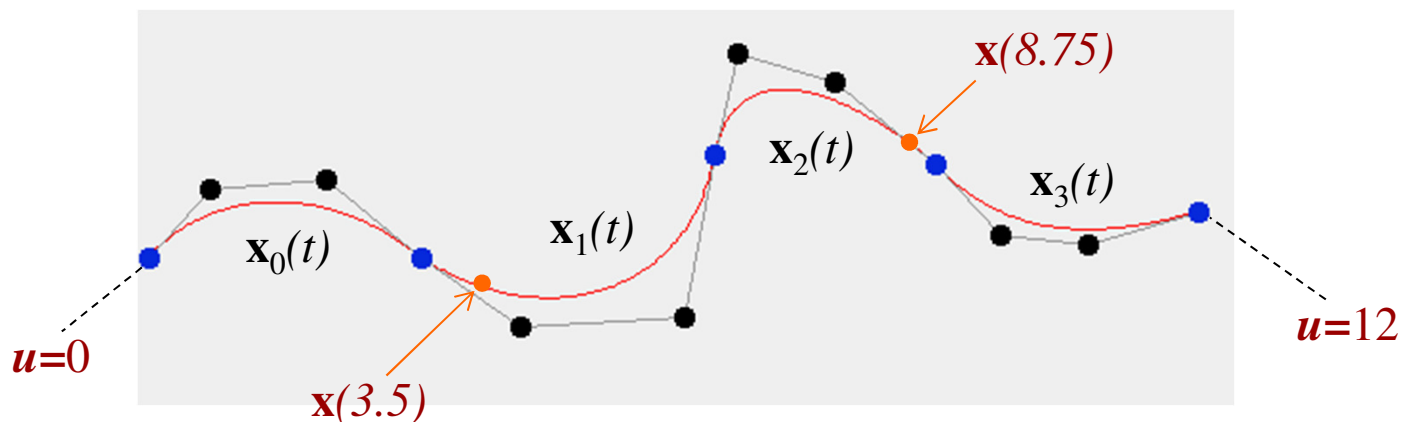


Piecewise Bézier Curve

- Parameter in $0 \leq u \leq 3N$

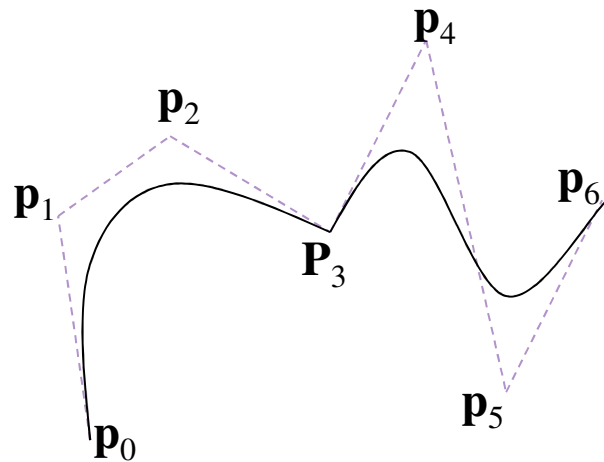
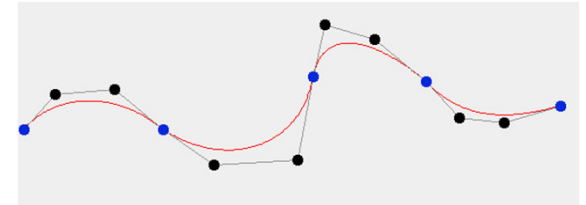
$$\mathbf{x}(u) = \begin{cases} \mathbf{x}_0(\frac{1}{3}u), & 0 \leq u \leq 3 \\ \mathbf{x}_1(\frac{1}{3}u - 1), & 3 \leq u \leq 6 \\ \vdots & \vdots \\ \mathbf{x}_{N-1}(\frac{1}{3}u - (N-1)), & 3N-3 \leq u \leq 3N \end{cases}$$

$$\mathbf{x}(u) = \mathbf{x}_i\left(\frac{1}{3}u - i\right), \text{ where } i = \left\lfloor \frac{1}{3}u \right\rfloor$$

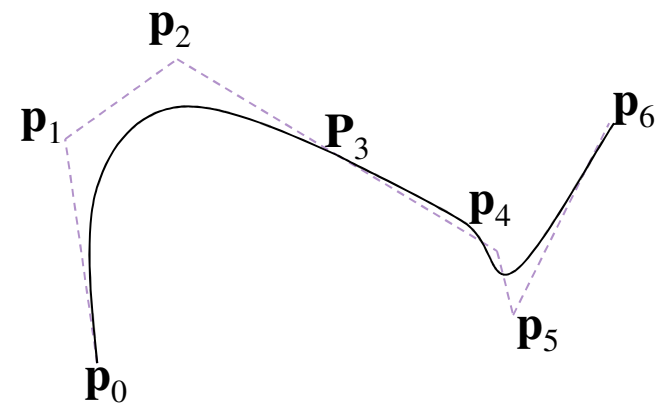


Piecewise Bézier Curve

- ▶ $3N+1$ points define N Bézier segments
- ▶ $\mathbf{x}(3i) = \mathbf{p}_{3i}$
- ▶ C_0 continuous by construction
- ▶ C_1 continuous at \mathbf{p}_{3i} when $\mathbf{p}_{3i} - \mathbf{p}_{3i-1} = \mathbf{p}_{3i+1} - \mathbf{p}_{3i}$
- ▶ C_2 is harder to achieve



C_1 discontinuous



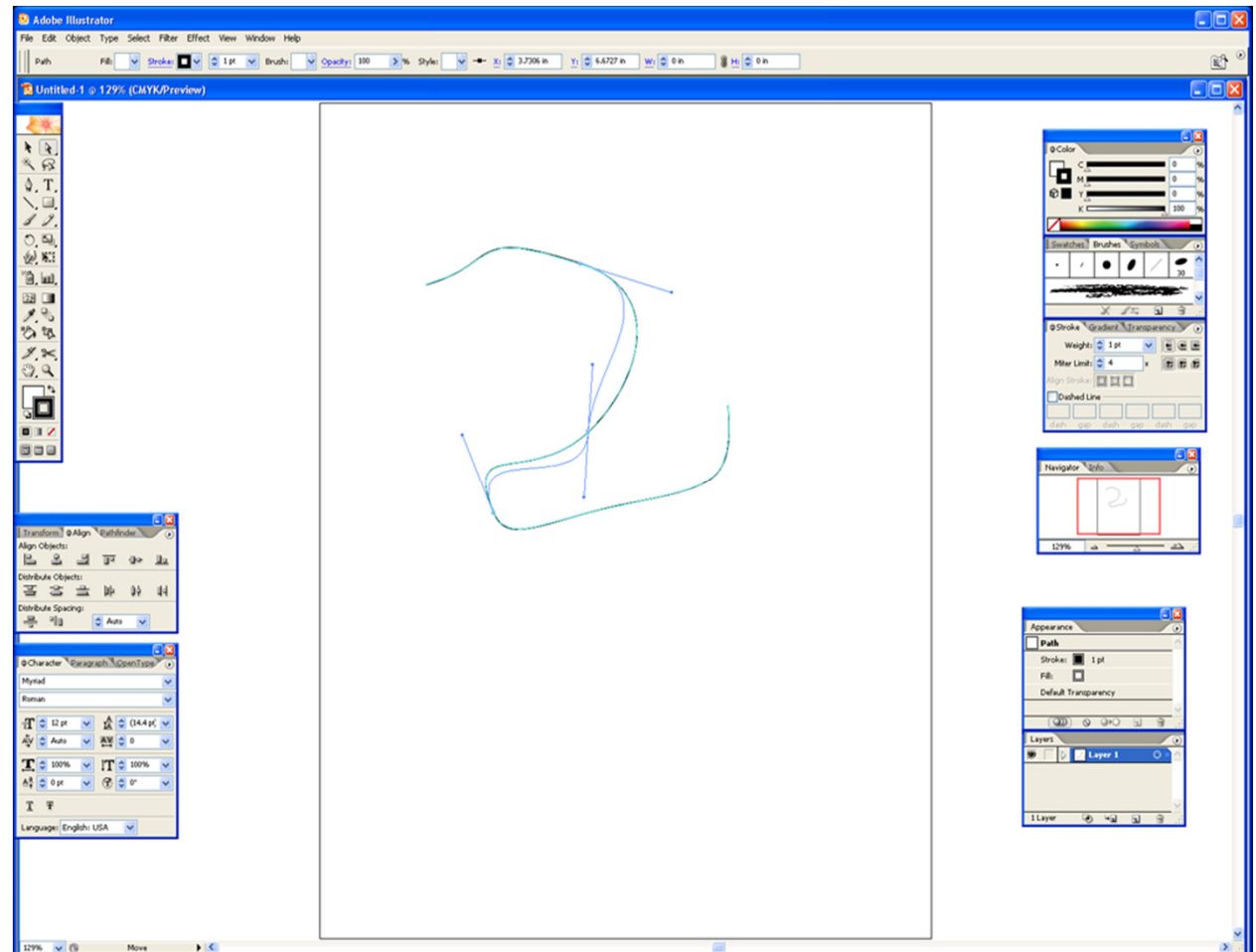
C_1 continuous

Piecewise Bézier Curves

- ▶ Used often in 2D drawing programs
- ▶ Inconveniences
 - ▶ Must have 4 or 7 or 10 or 13 or ... (1 plus a multiple of 3) control points
 - ▶ Some points interpolate, others approximate
 - ▶ Need to impose constraints on control points to obtain C^1 continuity
 - ▶ C_2 continuity more difficult
- ▶ Solutions
 - ▶ User interface using “Bézier handles”
 - ▶ Generalization to B-splines or NURBS

Bézier Handles

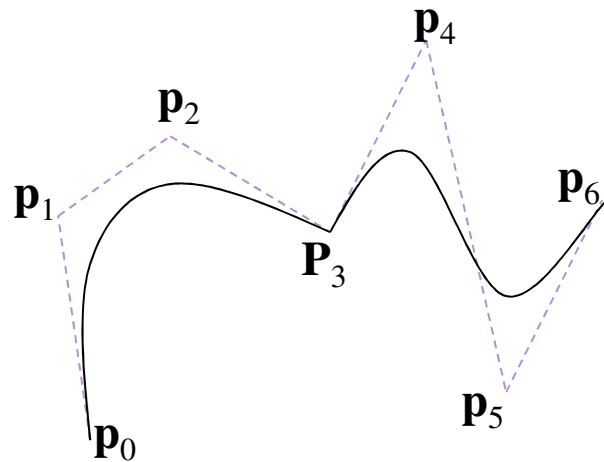
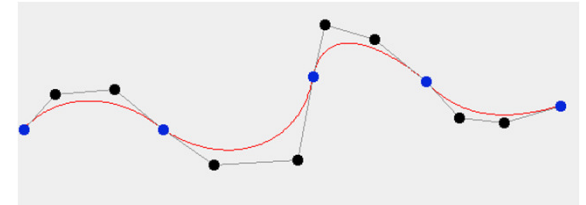
- ▶ Segment end points (interpolating) presented as curve control points
- ▶ Midpoints (approximating points) presented as “handles”
- ▶ Can have option to enforce C_1 continuity



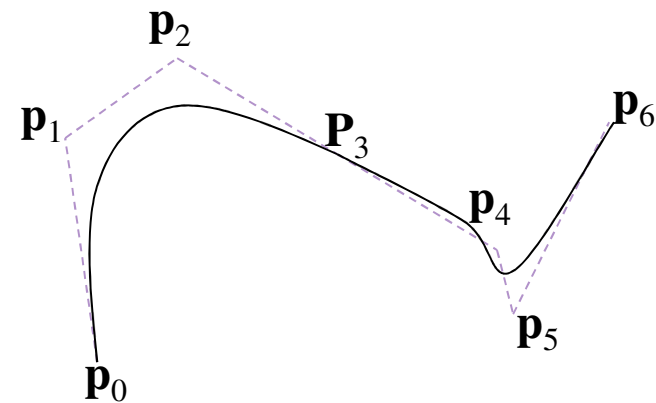
Adobe Illustrator

Piecewise Bézier Curve

- ▶ $3N+1$ points define N Bézier segments
- ▶ $\mathbf{x}(3i)=\mathbf{p}_{3i}$
- ▶ C_0 continuous by construction
- ▶ C_1 continuous at \mathbf{p}_{3i} when $\mathbf{p}_{3i} - \mathbf{p}_{3i-1} = \mathbf{p}_{3i+1} - \mathbf{p}_{3i}$
- ▶ C_2 is harder to achieve



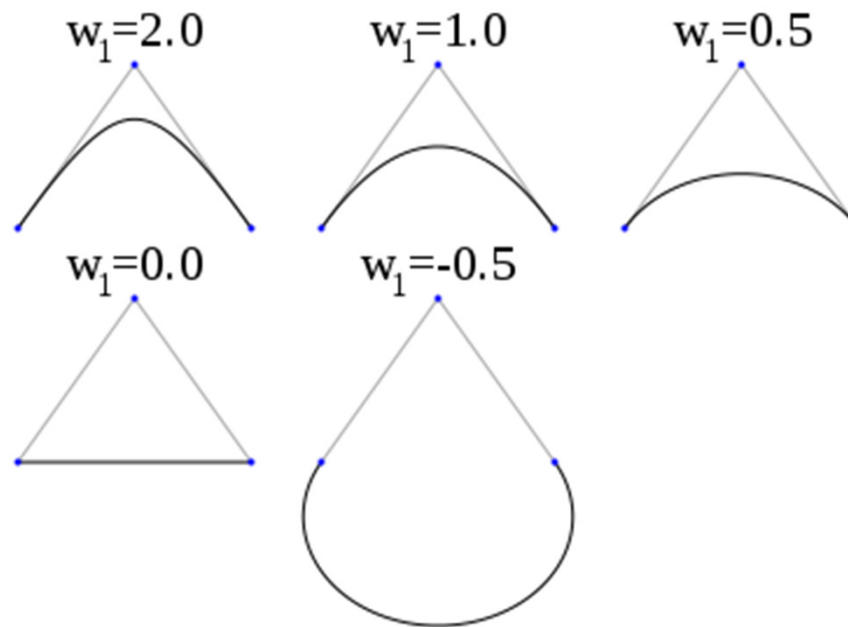
C_1 discontinuous



C_1 continuous

Rational Curves

- ▶ Weight causes point to “pull” more (or less)
- ▶ Can model circles with proper points and weights,
- ▶ Below: rational quadratic Bézier curve (three control points)



B-Splines

- ▶ B as in **B**asis-Splines
- ▶ Basis is blending function
- ▶ Difference to Bézier blending function:
 - ▶ B-spline blending function can be zero outside a particular range (limits scope over which a control point has influence)
- ▶ B-Spline is defined by control points and range in which each control point is active.

NURBS

- ▶ **Non Uniform Rational B-Splines**
- ▶ Generalization of Bézier curves
- ▶ Non uniform:
- ▶ Combine B-Splines (limited scope of control points) and Rational Curves (weighted control points)
- ▶ Can exactly model conic sections (circles, ellipses)
- ▶ OpenGL support: see `gluNurbsCurve`
- ▶ <http://bentonian.com/teaching/AdvGraph0809/demos/Nurbs2d/index.html>
- ▶ <http://mathworld.wolfram.com/NURBSCurve.html>

Lecture Overview

- ▶ **Bi-linear patch**
- ▶ Bi-cubic Bézier patch

Curved Surfaces

Curves

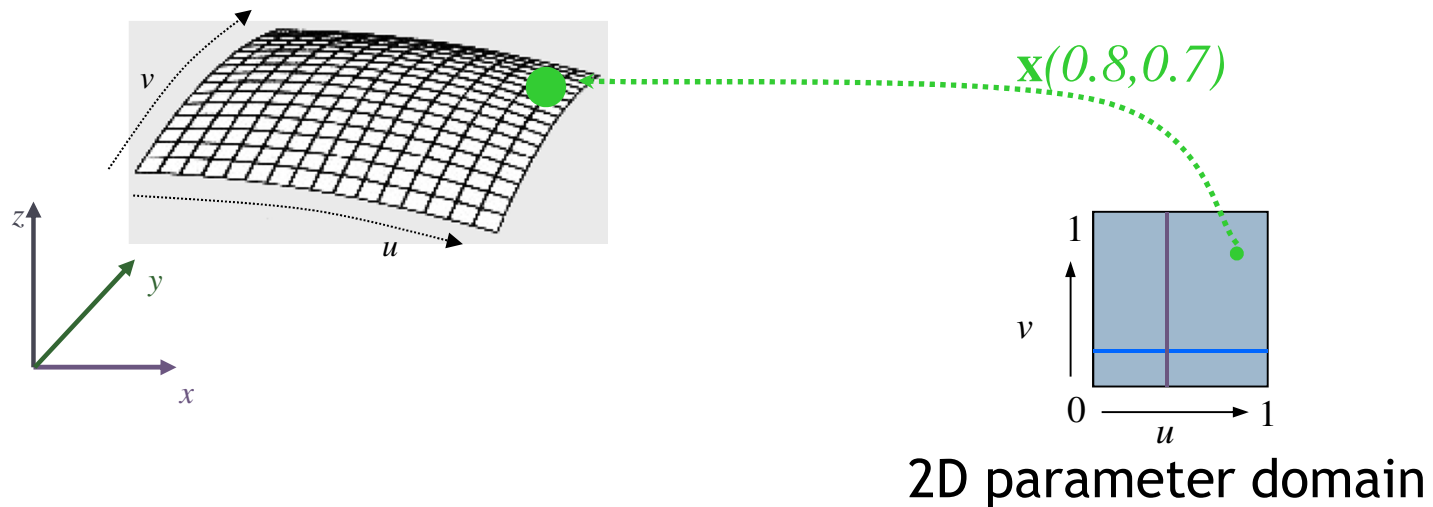
- ▶ Described by a 1D series of control points
- ▶ A function $\mathbf{x}(t)$
- ▶ Segments joined together to form a longer curve

Surfaces

- ▶ Described by a 2D mesh of control points
- ▶ Parameters have two dimensions (two dimensional parameter domain)
- ▶ A function $\mathbf{x}(u, v)$
- ▶ **Patches** joined together to form a bigger surface

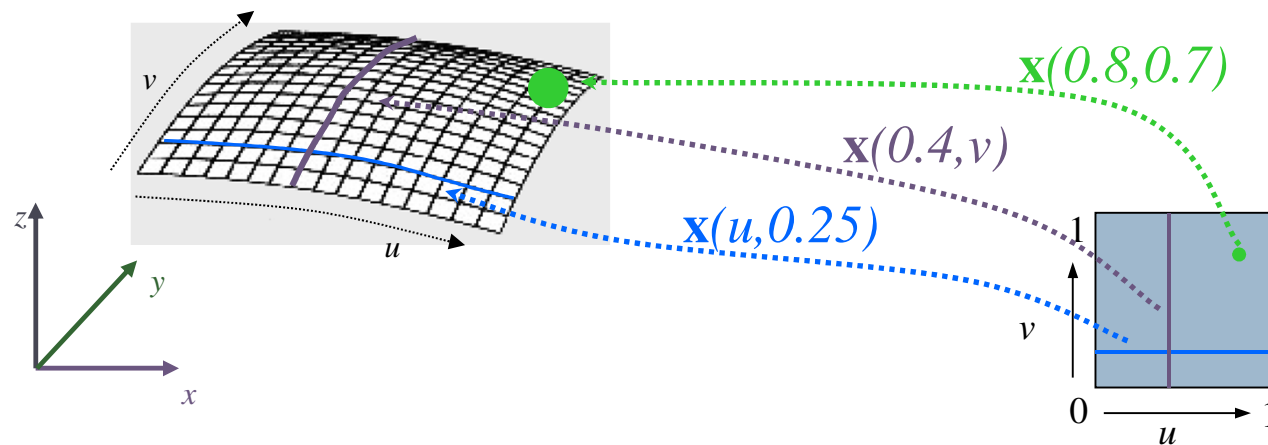
Parametric Surface Patch

- ▶ $\mathbf{x}(u,v)$ describes a point in space for any given (u,v) pair
 - ▶ u,v each range from 0 to 1



Parametric Surface Patch

- ▶ $\mathbf{x}(u,v)$ describes a point in space for any given (u,v) pair
 - ▶ u,v each range from 0 to 1

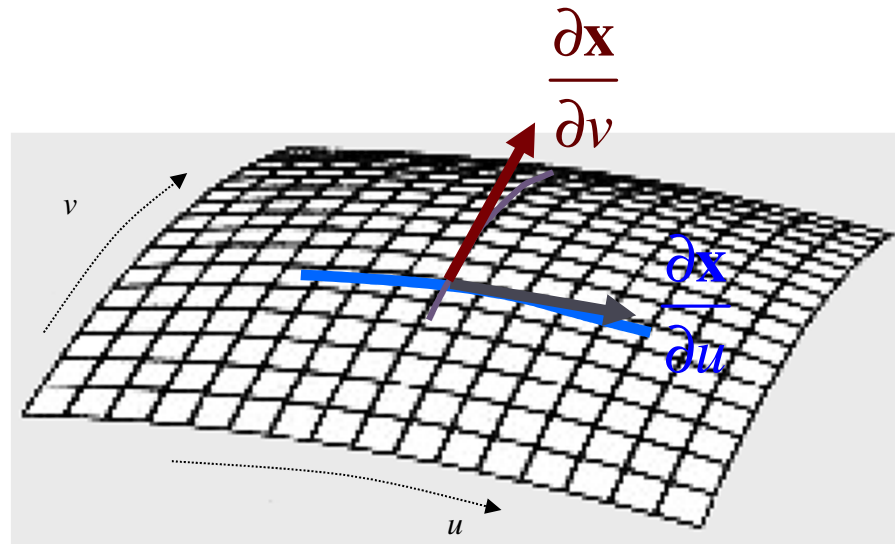


2D parameter domain

- ▶ Parametric curves
 - ▶ For fixed u_0 , have a v curve $\mathbf{x}(u_0, v)$
 - ▶ For fixed v_0 , have a u curve $\mathbf{x}(u, v_0)$
 - ▶ For any point on the surface, there are a pair of parametric curves through that point

Tangents

- ▶ The tangent to a parametric curve is also tangent to the surface
- ▶ For any point on the surface, there are a pair of (parametric) tangent vectors
- ▶ Note: these vectors are not necessarily perpendicular to each other



Tangents

- Notation:

- The tangent along a u curve, AKA the tangent in the u direction, is written as:

$$\frac{\partial \mathbf{x}}{\partial u}(u, v) \text{ or } \frac{\partial}{\partial u} \mathbf{x}(u, v) \text{ or } \mathbf{x}_u(u, v)$$

- The tangent along a v curve, AKA the tangent in the v direction, is written as:

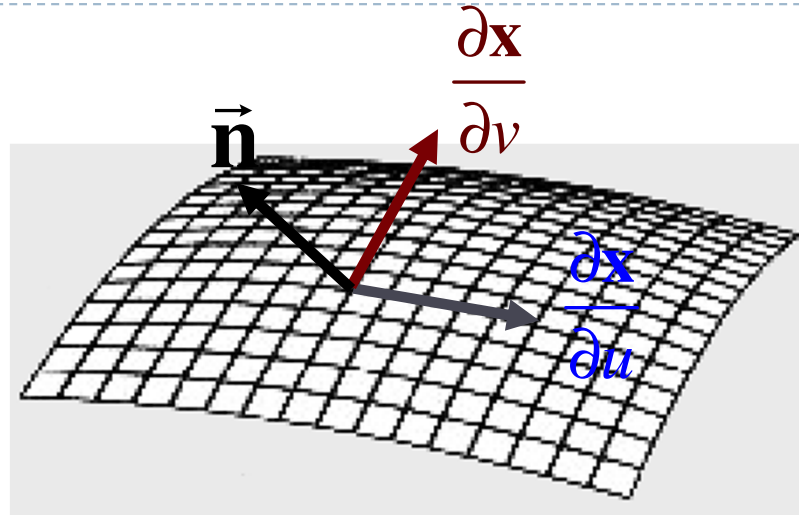
$$\frac{\partial \mathbf{x}}{\partial v}(u, v) \text{ or } \frac{\partial}{\partial v} \mathbf{x}(u, v) \text{ or } \mathbf{x}_v(u, v)$$

- Note that each of these is a vector-valued function:

- At each point $\mathbf{x}(u, v)$ on the surface, we have tangent vectors $\frac{\partial}{\partial u} \mathbf{x}(u, v)$ and $\frac{\partial}{\partial v} \mathbf{x}(u, v)$

Surface Normal

- ▶ Normal is cross product of the two tangent vectors
- ▶ Order matters!



$$\vec{n}(u, v) = \frac{\partial \mathbf{x}}{\partial u}(u, v) \times \frac{\partial \mathbf{x}}{\partial v}(u, v)$$

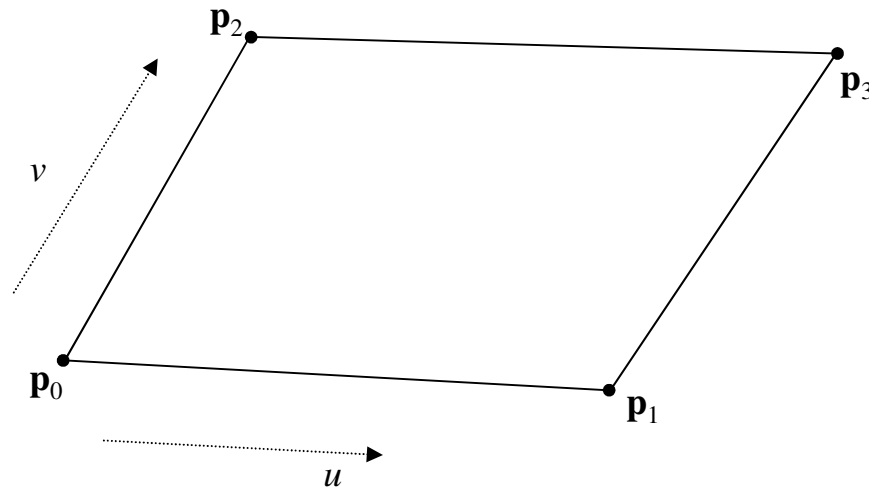
Typically we are interested in the unit normal, so we need to normalize

$$\vec{n}^*(u, v) = \frac{\partial \mathbf{x}}{\partial u}(u, v) \times \frac{\partial \mathbf{x}}{\partial v}(u, v)$$

$$\vec{n}(u, v) = \frac{\vec{n}^*(u, v)}{|\vec{n}^*(u, v)|}$$

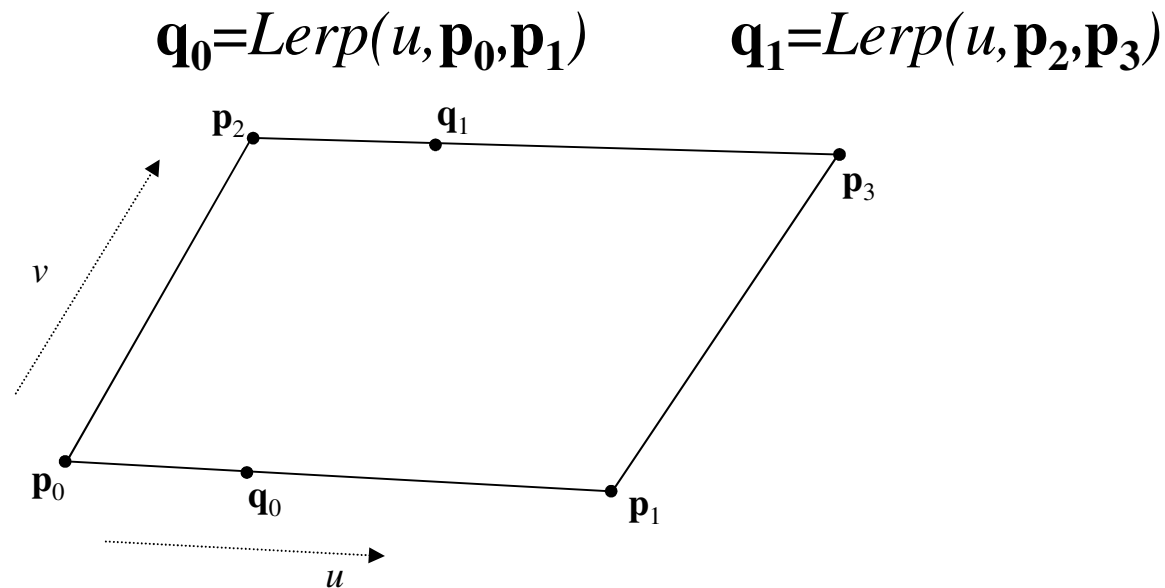
Bilinear Patch

- ▶ Control mesh with four points $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$
- ▶ Compute $\mathbf{x}(u, v)$ using a two-step construction scheme



Bilinear Patch (Step 1)

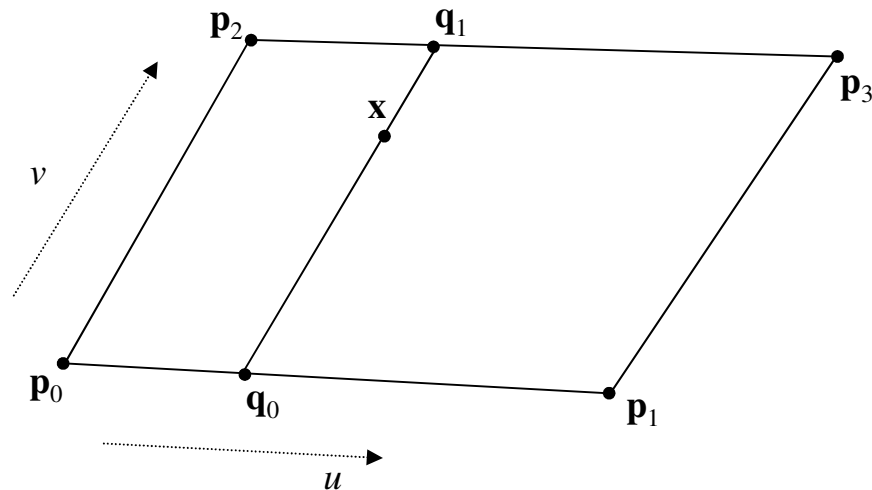
- ▶ For a given value of u , evaluate the linear curves on the two u -direction edges
- ▶ Use the same value u for both:



Bilinear Patch (Step 2)

- ▶ Consider that $\mathbf{q}_0, \mathbf{q}_1$ define a line segment
- ▶ Evaluate it using v to get \mathbf{x}

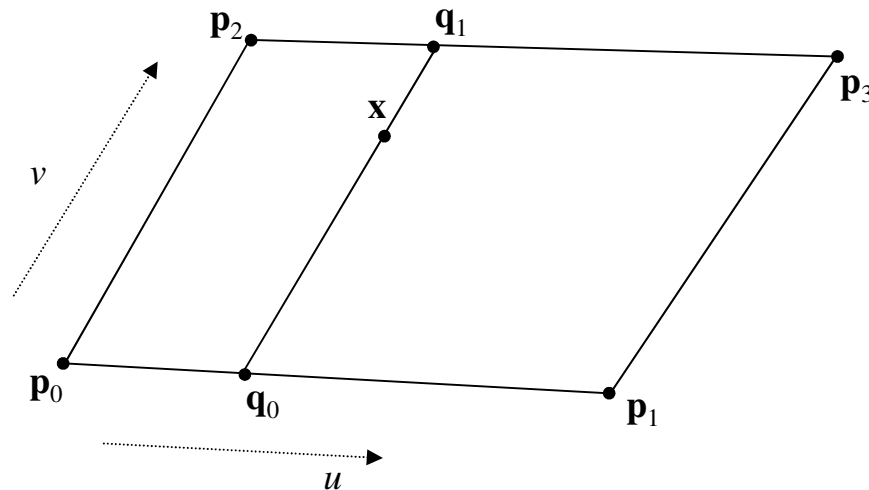
$$\mathbf{x} = \text{Lerp}(v, \mathbf{q}_0, \mathbf{q}_1)$$



Bilinear Patch

- Combining the steps, we get the full formula

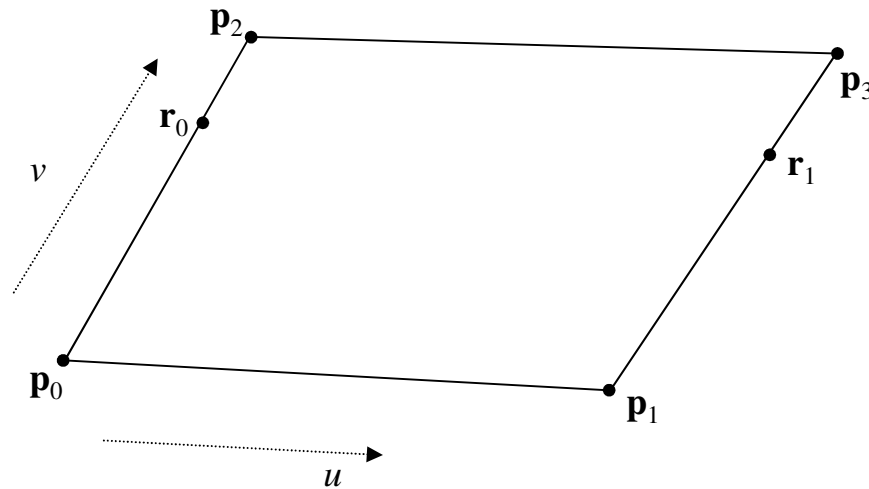
$$\mathbf{x}(u, v) = \text{Lerp}(v, \text{Lerp}(u, \mathbf{p}_0, \mathbf{p}_1), \text{Lerp}(u, \mathbf{p}_2, \mathbf{p}_3))$$



Bilinear Patch

- ▶ Try the other order
- ▶ Evaluate first in the v direction

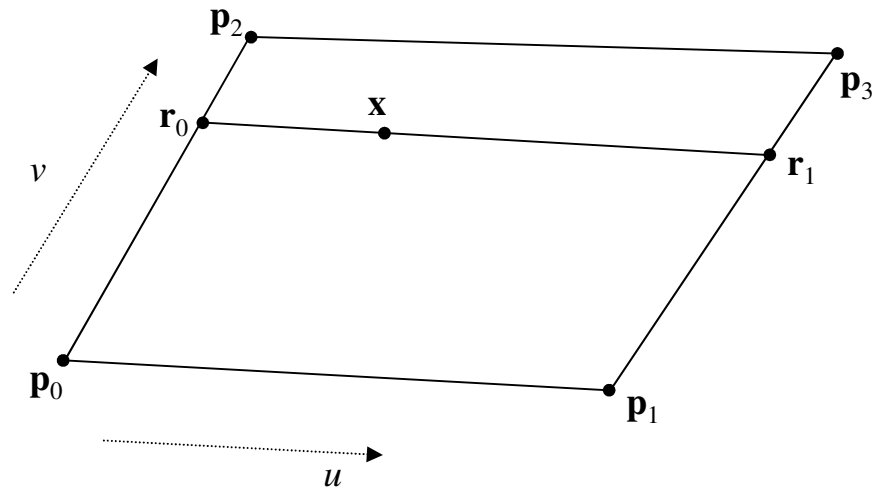
$$\mathbf{r}_0 = \text{Lerp}(v, \mathbf{p}_0, \mathbf{p}_2) \quad \mathbf{r}_1 = \text{Lerp}(v, \mathbf{p}_1, \mathbf{p}_3)$$



Bilinear Patch

- ▶ Consider that $\mathbf{r}_0, \mathbf{r}_1$ define a line segment
- ▶ Evaluate it using u to get \mathbf{x}

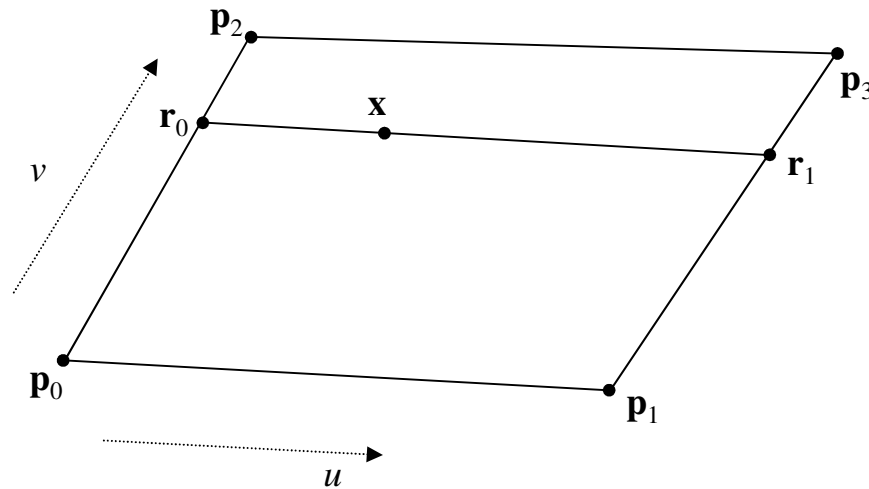
$$\mathbf{x} = \text{Lerp}(u, \mathbf{r}_0, \mathbf{r}_1)$$



Bilinear Patch

- ▶ The full formula for the v direction first:

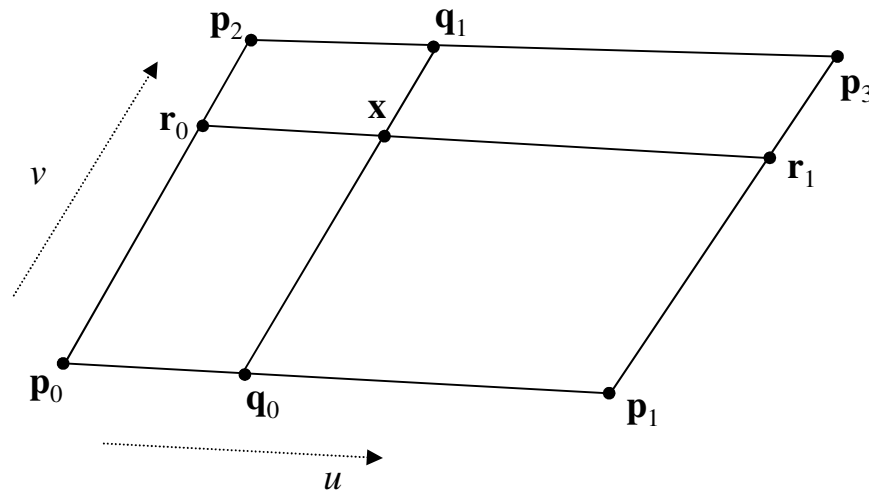
$$\mathbf{x}(u, v) = \text{Lerp}(u, \text{Lerp}(v, \mathbf{p}_0, \mathbf{p}_2), \text{Lerp}(v, \mathbf{p}_1, \mathbf{p}_3))$$



Bilinear Patch

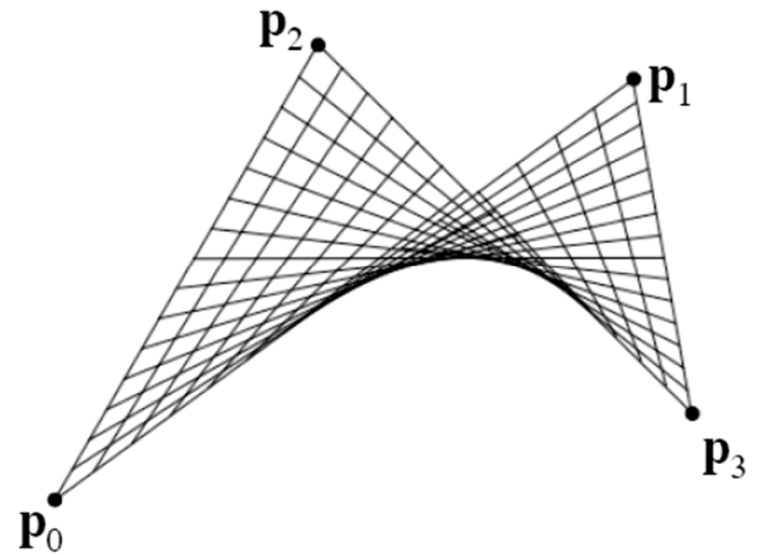
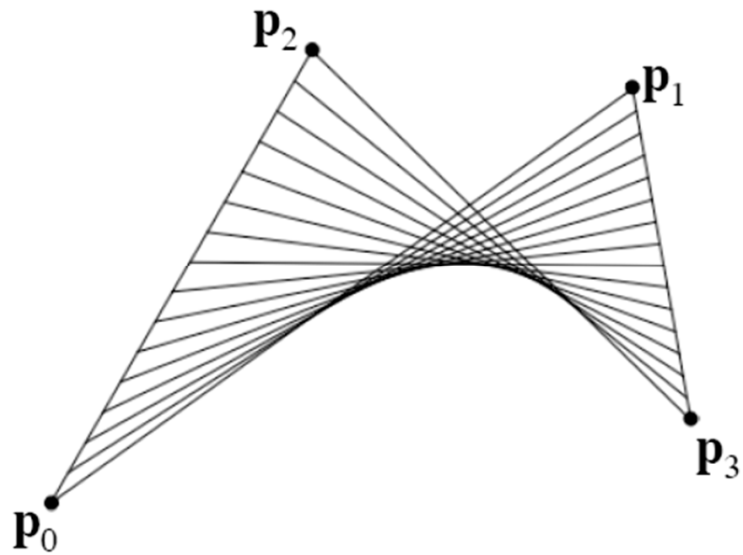
- Patch geometry is independent of the order of u and v

$$\begin{aligned}\mathbf{x}(u,v) &= \text{Lerp}(v, \text{Lerp}(u, \mathbf{p}_0, \mathbf{p}_1), \text{Lerp}(u, \mathbf{p}_2, \mathbf{p}_3)) \\ \mathbf{x}(u,v) &= \text{Lerp}(u, \text{Lerp}(v, \mathbf{p}_0, \mathbf{p}_2), \text{Lerp}(v, \mathbf{p}_1, \mathbf{p}_3))\end{aligned}$$



Bilinear Patch

► Visualization



Bilinear Patches

- ▶ **Weighted sum of control points**

$$\mathbf{x}(u, v) = (1-u)(1-v)\mathbf{p}_0 + u(1-v)\mathbf{p}_1 + (1-u)v\mathbf{p}_2 + uv\mathbf{p}_3$$

- ▶ **Bilinear polynomial**

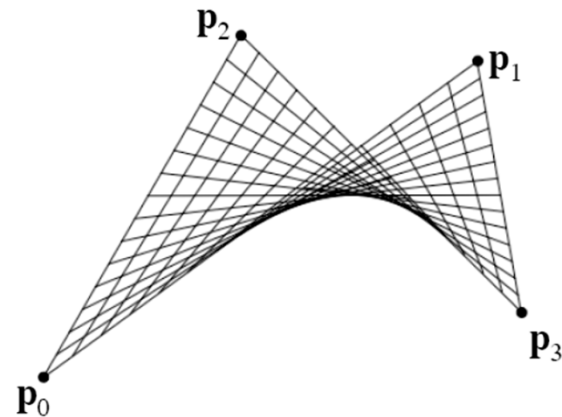
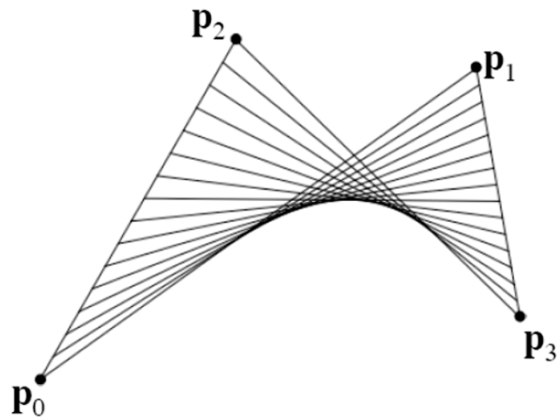
$$\mathbf{x}(u, v) = (\mathbf{p}_0 - \mathbf{p}_1 - \mathbf{p}_2 + \mathbf{p}_3)uv + (\mathbf{p}_1 - \mathbf{p}_0)u + (\mathbf{p}_2 - \mathbf{p}_0)v + \mathbf{p}_0$$

- ▶ **Matrix form**

$$x(u, v) = \begin{bmatrix} 1-u & u \end{bmatrix} \begin{bmatrix} p_0 & p_2 \\ p_1 & p_3 \end{bmatrix} \begin{bmatrix} 1-v \\ v \end{bmatrix}$$

Properties

- ▶ Interpolates the control points
- ▶ The boundaries are straight line segments
- ▶ If all 4 points of the control mesh are co-planar, the patch is flat
- ▶ If the points are not co-planar, we get a curved surface
 - ▶ saddle shape (hyperbolic paraboloid)
- ▶ *The parametric curves are all straight line segments!*
 - ▶ a (doubly) *ruled surface*: has (two) straight lines through every point



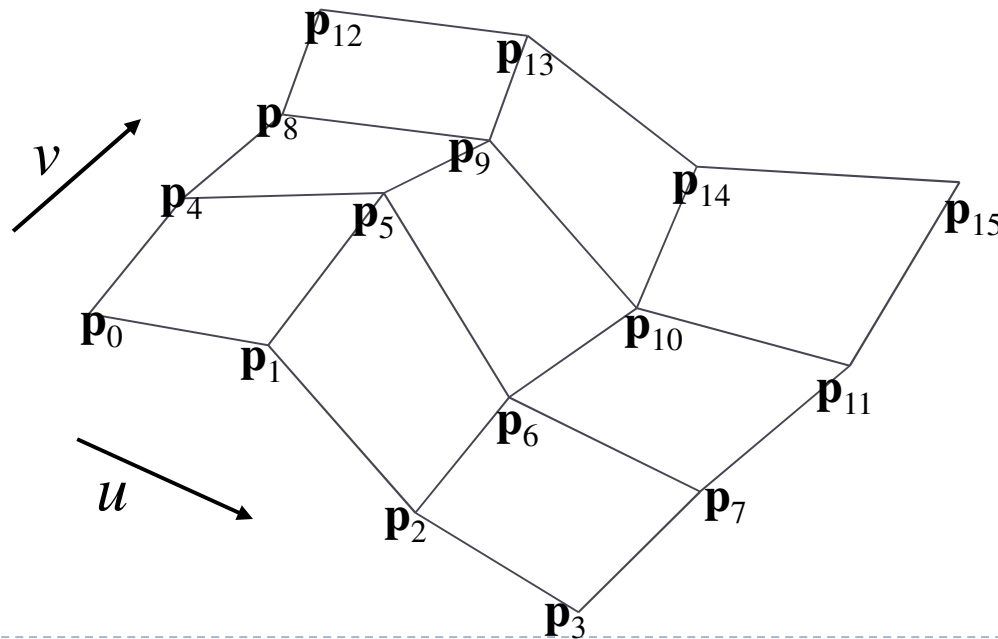
- ▶ Not terribly useful as a modeling primitive

Lecture Overview

- ▶ Bi-linear patch
- ▶ Bi-cubic Bézier patch

Bicubic Bézier patch

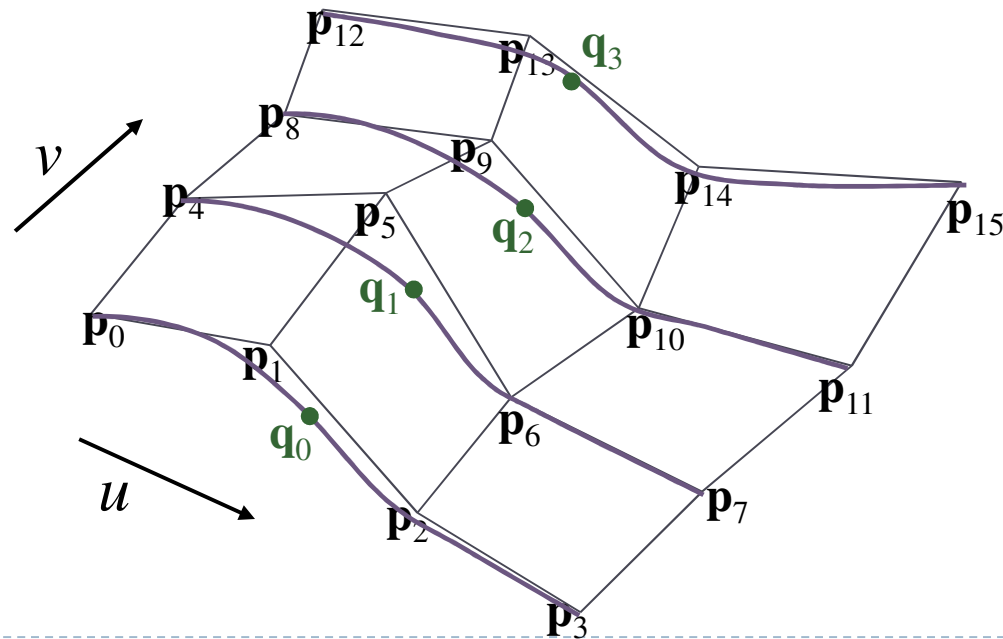
- ▶ Grid of 4x4 control points, \mathbf{p}_0 through \mathbf{p}_{15}
- ▶ Four rows of control points define Bézier curves along u
 $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$; $\mathbf{p}_4, \mathbf{p}_5, \mathbf{p}_6, \mathbf{p}_7$; $\mathbf{p}_8, \mathbf{p}_9, \mathbf{p}_{10}, \mathbf{p}_{11}$; $\mathbf{p}_{12}, \mathbf{p}_{13}, \mathbf{p}_{14}, \mathbf{p}_{15}$
- ▶ Four columns define Bézier curves along v
 $\mathbf{p}_0, \mathbf{p}_4, \mathbf{p}_8, \mathbf{p}_{12}$; $\mathbf{p}_1, \mathbf{p}_5, \mathbf{p}_9, \mathbf{p}_{13}$; $\mathbf{p}_2, \mathbf{p}_6, \mathbf{p}_{10}, \mathbf{p}_{14}$; $\mathbf{p}_3, \mathbf{p}_7, \mathbf{p}_{11}, \mathbf{p}_{15}$



Bézier Patch (Step 1)

- ▶ Evaluate four u -direction Bézier curves at scalar value u $[0..1]$

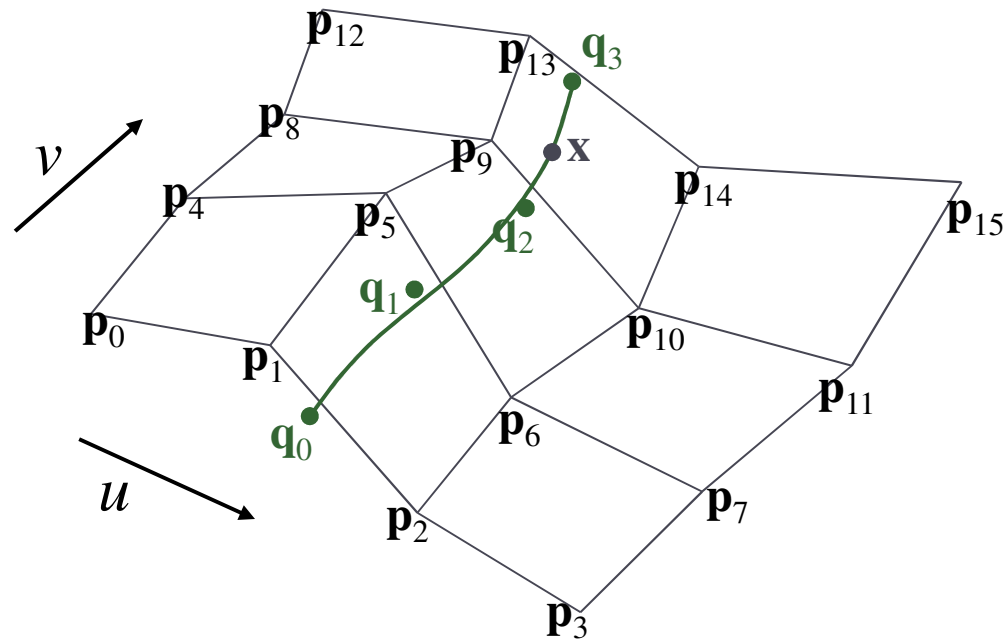
- ▶ Get points $\mathbf{q}_0 \dots \mathbf{q}_3$
 $\mathbf{q}_0 = \text{Bez}(u, \mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$
 $\mathbf{q}_1 = \text{Bez}(u, \mathbf{p}_4, \mathbf{p}_5, \mathbf{p}_6, \mathbf{p}_7)$
 $\mathbf{q}_2 = \text{Bez}(u, \mathbf{p}_8, \mathbf{p}_9, \mathbf{p}_{10}, \mathbf{p}_{11})$
 $\mathbf{q}_3 = \text{Bez}(u, \mathbf{p}_{12}, \mathbf{p}_{13}, \mathbf{p}_{14}, \mathbf{p}_{15})$



Bézier Patch (Step 2)

- ▶ Points $\mathbf{q}_0 \dots \mathbf{q}_3$ define a Bézier curve
- ▶ Evaluate it at $v \in [0..1]$

$$\mathbf{x}(u, v) = \text{Bez}(v, \mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3)$$



Bézier Patch

- Same result in either order (evaluate u before v or vice versa)

$$\mathbf{q}_0 = \text{Bez}(u, \mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$$

$$\mathbf{q}_1 = \text{Bez}(u, \mathbf{p}_4, \mathbf{p}_5, \mathbf{p}_6, \mathbf{p}_7)$$

$$\mathbf{q}_2 = \text{Bez}(u, \mathbf{p}_8, \mathbf{p}_9, \mathbf{p}_{10}, \mathbf{p}_{11}) \Leftrightarrow$$

$$\mathbf{q}_3 = \text{Bez}(u, \mathbf{p}_{12}, \mathbf{p}_{13}, \mathbf{p}_{14}, \mathbf{p}_{15})$$

$$\mathbf{r}_0 = \text{Bez}(v, \mathbf{p}_0, \mathbf{p}_4, \mathbf{p}_8, \mathbf{p}_{12})$$

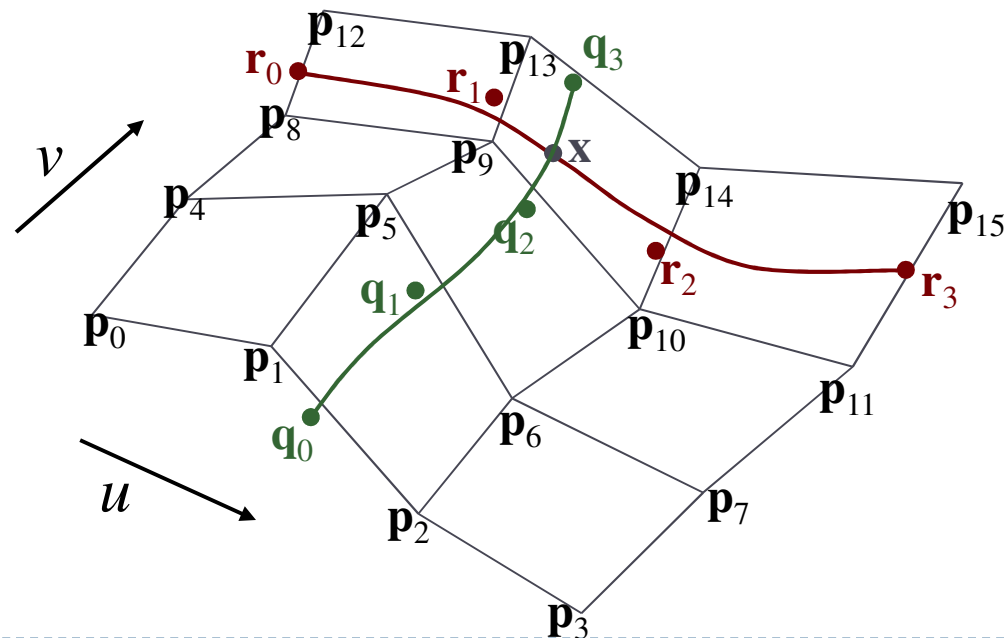
$$\mathbf{r}_1 = \text{Bez}(v, \mathbf{p}_1, \mathbf{p}_5, \mathbf{p}_9, \mathbf{p}_{13})$$

$$\mathbf{r}_2 = \text{Bez}(v, \mathbf{p}_2, \mathbf{p}_6, \mathbf{p}_{10}, \mathbf{p}_{14})$$

$$\mathbf{r}_3 = \text{Bez}(v, \mathbf{p}_3, \mathbf{p}_7, \mathbf{p}_{11}, \mathbf{p}_{15})$$

$$\mathbf{x}(u, v) = \text{Bez}(v, \mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3)$$

$$\mathbf{x}(u, v) = \text{Bez}(u, \mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3)$$



Bézier Patch: Matrix Form

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

$$\mathbf{B}_{Bez} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} = \mathbf{B}_{Bez}^T$$

$$\mathbf{C}_x = \mathbf{B}_{Bez}^T \mathbf{G}_x \mathbf{B}_{Bez}$$

$$\mathbf{C}_y = \mathbf{B}_{Bez}^T \mathbf{G}_y \mathbf{B}_{Bez}$$

$$\mathbf{C}_z = \mathbf{B}_{Bez}^T \mathbf{G}_z \mathbf{B}_{Bez}$$

$$\mathbf{G}_x = \begin{bmatrix} p_{0x} & p_{1x} & p_{2x} & p_{3x} \\ p_{4x} & p_{5x} & p_{6x} & p_{7x} \\ p_{8x} & p_{9x} & p_{10x} & p_{11x} \\ p_{12x} & p_{13x} & p_{14x} & p_{15x} \end{bmatrix}, \quad \mathbf{G}_y = \dots, \quad \mathbf{G}_z = \dots$$

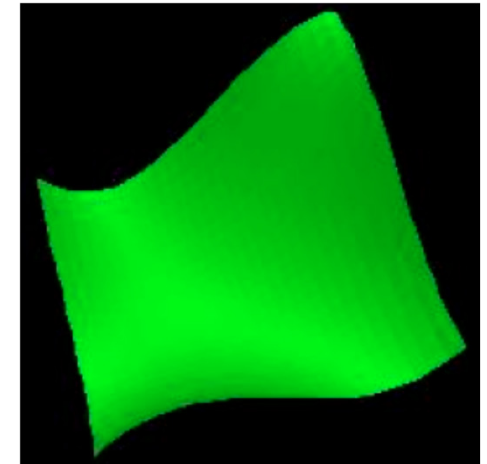
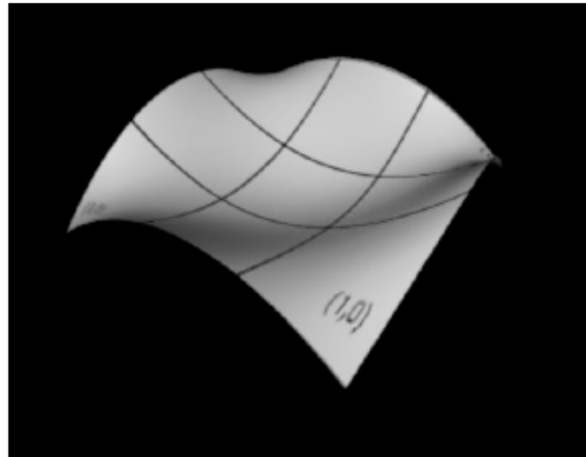
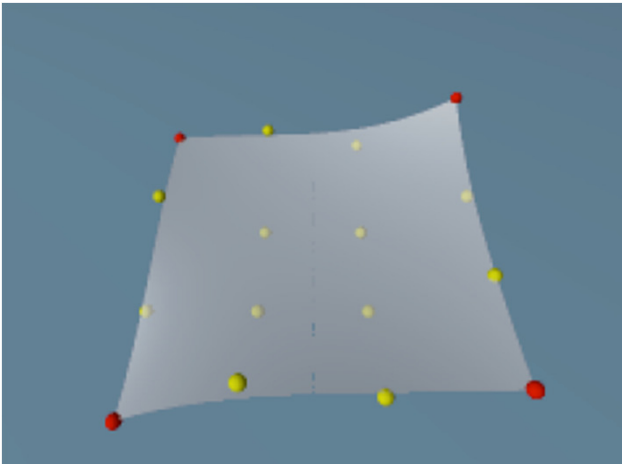
$$\mathbf{x}(u, v) = \begin{bmatrix} \mathbf{V}^T \mathbf{C}_x \mathbf{U} \\ \mathbf{V}^T \mathbf{C}_y \mathbf{U} \\ \mathbf{V}^T \mathbf{C}_z \mathbf{U} \end{bmatrix}$$

Bézier Patch: Matrix Form

- ▶ \mathbf{C}_x stores the coefficients of the bicubic equation for x
 - ▶ \mathbf{C}_y stores the coefficients of the bicubic equation for y
 - ▶ \mathbf{C}_z stores the coefficients of the bicubic equation for z
 - ▶ \mathbf{G}_x stores the geometry (x components of the control points)
 - ▶ \mathbf{G}_y stores the geometry (y components of the control points)
 - ▶ \mathbf{G}_z stores the geometry (z components of the control points)
 - ▶ \mathbf{B}_{Bez} is the basis matrix (Bézier basis)
 - ▶ \mathbf{U} and \mathbf{V} are the vectors formed from the powers of u and v
-
- ▶ Compact notation
 - ▶ Leads to efficient method of computation
 - ▶ Can take advantage of hardware support for 4x4 matrix arithmetic

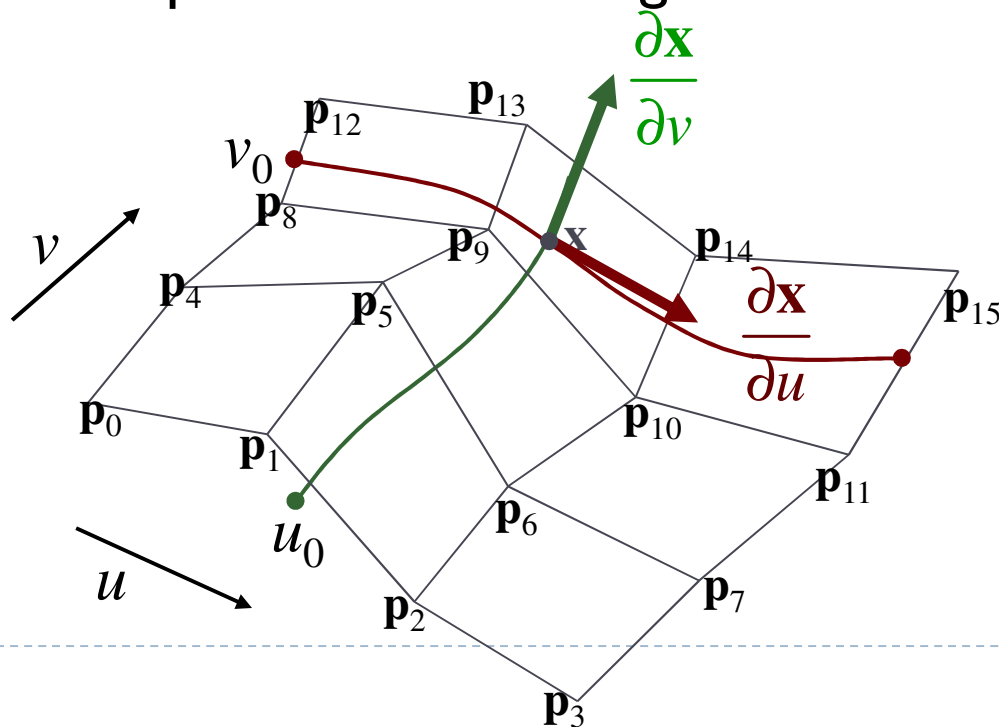
Properties

- ▶ Convex hull: any point on the surface will fall within the convex hull of the control points
- ▶ Interpolates 4 corner points
- ▶ Approximates other 12 points, which act as “handles”
- ▶ The boundaries of the patch are the Bézier curves defined by the points on the mesh edges
- ▶ The parametric curves are all Bézier curves



Tangents of a Bézier patch

- ▶ Remember parametric curves $\mathbf{x}(u, v_0)$, $\mathbf{x}(u_0, v)$ where v_0, u_0 is fixed
- ▶ Tangents to surface = tangents to parametric curves
- ▶ Tangents are partial derivatives of $\mathbf{x}(u, v)$
- ▶ Normal is cross product of the tangents



Tangents of a Bézier patch

$$\mathbf{q}_0 = \text{Bez}(u, \mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$$

$$\mathbf{q}_1 = \text{Bez}(u, \mathbf{p}_4, \mathbf{p}_5, \mathbf{p}_6, \mathbf{p}_7)$$

$$\mathbf{q}_2 = \text{Bez}(u, \mathbf{p}_8, \mathbf{p}_9, \mathbf{p}_{10}, \mathbf{p}_{11})$$

$$\mathbf{q}_3 = \text{Bez}(u, \mathbf{p}_{12}, \mathbf{p}_{13}, \mathbf{p}_{14}, \mathbf{p}_{15})$$

$$\mathbf{r}_0 = \text{Bez}(v, \mathbf{p}_0, \mathbf{p}_4, \mathbf{p}_8, \mathbf{p}_{12})$$

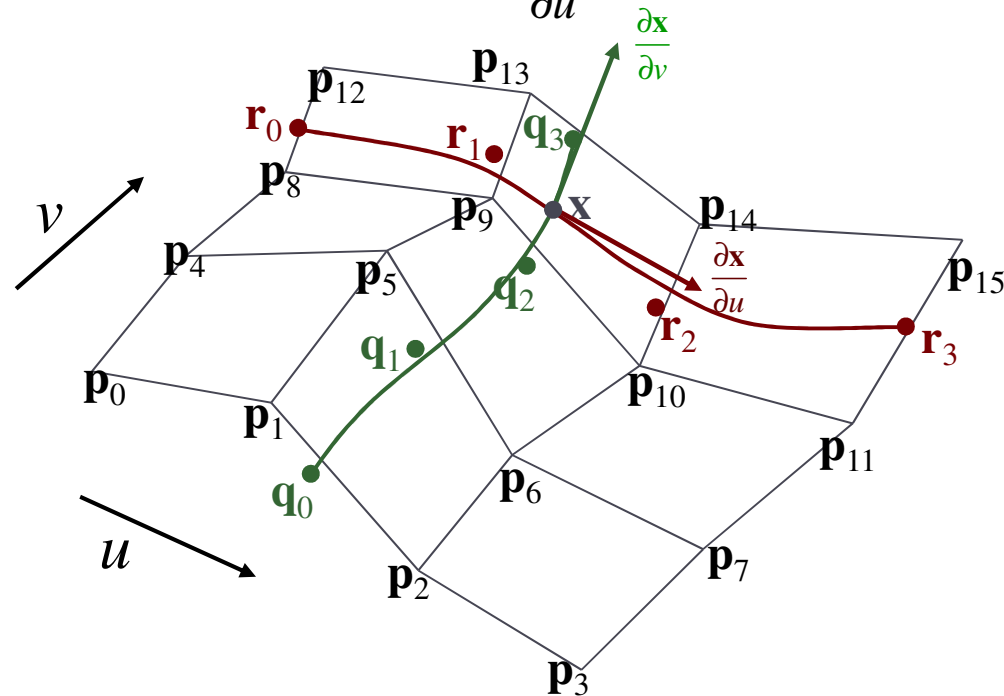
$$\mathbf{r}_1 = \text{Bez}(v, \mathbf{p}_1, \mathbf{p}_5, \mathbf{p}_9, \mathbf{p}_{13})$$

$$\mathbf{r}_2 = \text{Bez}(v, \mathbf{p}_2, \mathbf{p}_6, \mathbf{p}_{10}, \mathbf{p}_{14})$$

$$\mathbf{r}_3 = \text{Bez}(v, \mathbf{p}_3, \mathbf{p}_7, \mathbf{p}_{11}, \mathbf{p}_{15})$$

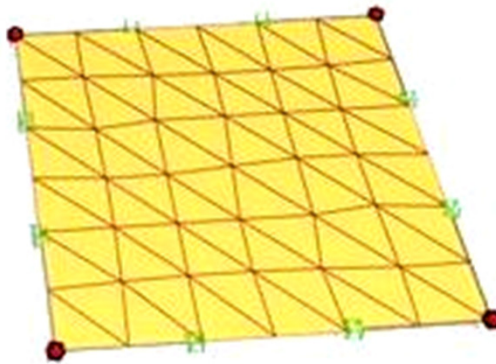
$$\frac{\partial \mathbf{x}}{\partial v}(u, v) = \text{Bez}'(v, \mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3)$$

$$\frac{\partial \mathbf{x}}{\partial u}(u, v) = \text{Bez}'(u, \mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3)$$



Tessellating a Bézier patch

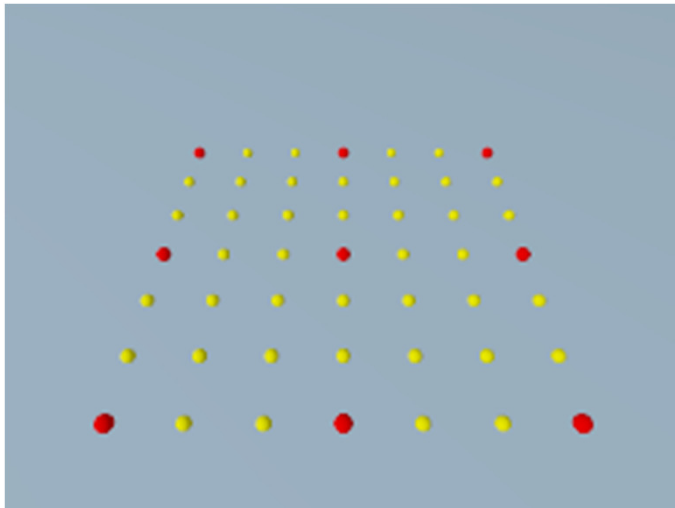
- ▶ Uniform tessellation is most straightforward
 - ▶ Evaluate points on a grid of u, v coordinates
 - ▶ Compute tangents at each point, take cross product to get per-vertex normal
 - ▶ Draw triangle strips with `glBegin(GL_TRIANGLE_STRIP)`



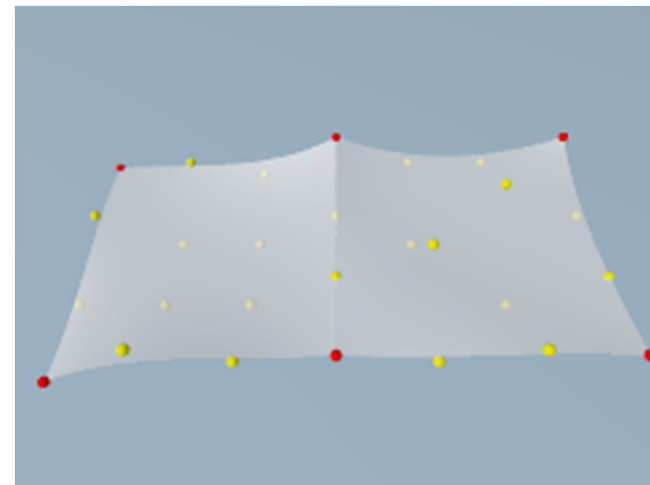
- ▶ Adaptive tessellation/recursive subdivision
 - ▶ Potential for “cracks” if patches on opposite sides of an edge divide differently
 - ▶ Tricky to get right, but can be done

Piecewise Bézier Surface

- ▶ Lay out grid of adjacent meshes of control points
- ▶ For C^0 continuity, must share points on the edge
 - ▶ Each edge of a Bézier patch is a Bézier curve based only on the edge mesh points
 - ▶ So if adjacent meshes share edge points, the patches will line up exactly
- ▶ But we have a crease...



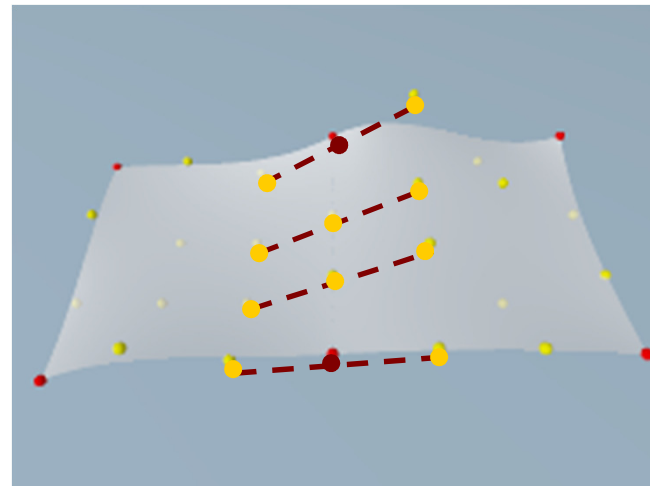
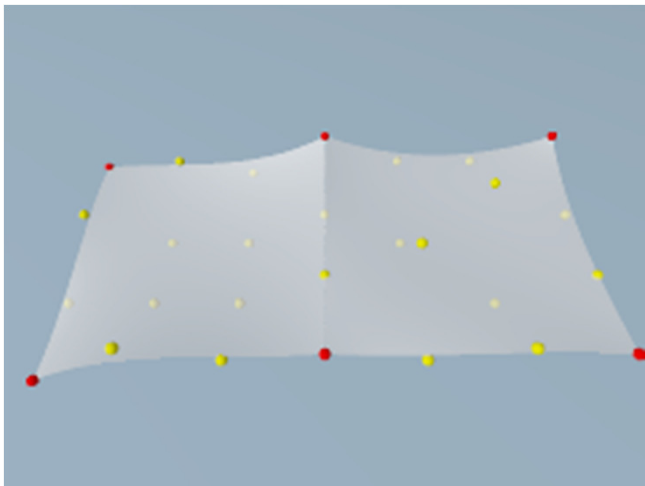
Grid of control points



Piecewise Bézier surface

C^1 Continuity

- ▶ We want the parametric curves that cross each edge to have C^1 continuity
 - ▶ So the handles must be equal-and-opposite across the edge:



<http://www.spiritone.com/~english/cyclopedia/patches.html>

Modeling With Bézier Patches

- ▶ Original Utah teapot, from Martin Newell's PhD thesis, consisted of 28 Bézier patches.
- ▶ The original had no rim for the lid and no bottom
- ▶ Later, four more patches were added to create a bottom, bringing the total to 32
- ▶ The data set was used by a number of people, including graphics guru Jim Blinn. In a demonstration of a system of his he scaled the teapot by .75, creating a stubbier teapot. He found it more pleasing to the eye, and it was this scaled version that became the highly popular dataset used today.

