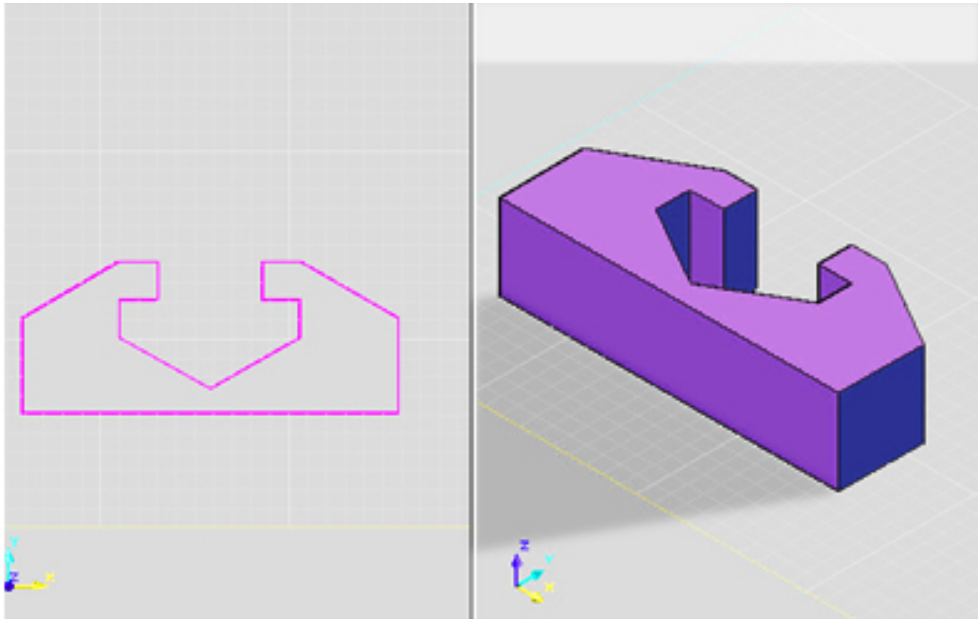
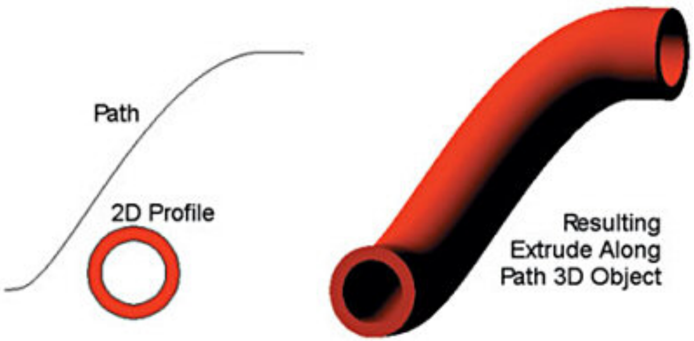
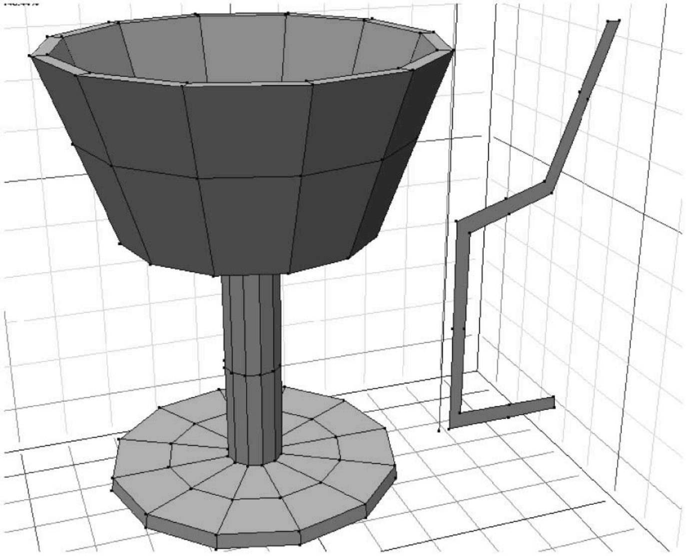
CMSC427
Parametric surfaces
(and alternatives)

Generating surfaces

- From equations
- From data
- From curves
  - Extrusion
    - Straight
    - Along path
  - Lathing (rotation)
  - Lofting

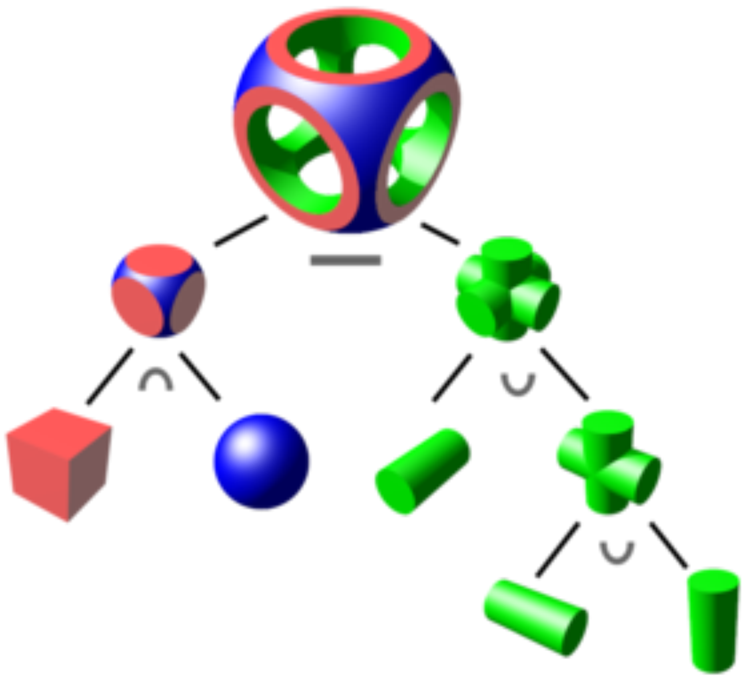






Constructive Solid Geometry (CSG)

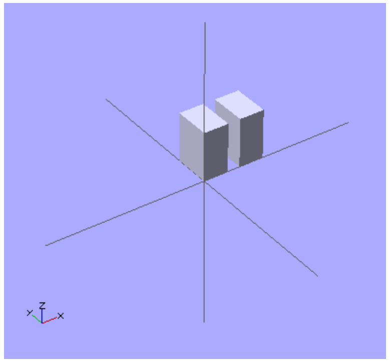
- Alternative/supplement to parametric shapes
- Vocabulary:
  - Basic set of shapes (sphere, box, cylinder, etc)
  - Set operations on shapes
  - Union
  - Intersection
  - Difference



- Demo
  - Tinkercad

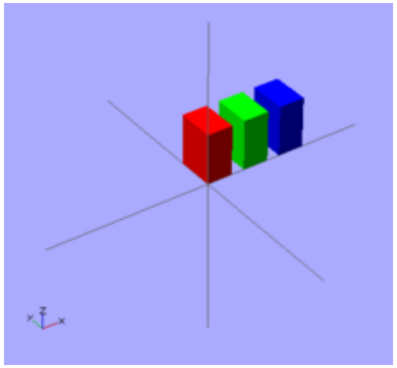
Constructive Solid Geometry (CSG)

- Computer Aided Design (CAD)
  - Precise 3D modeling for industrial design
  - Less freeform, more control and feedback on shapes
  - Often compiled (openScad.org)



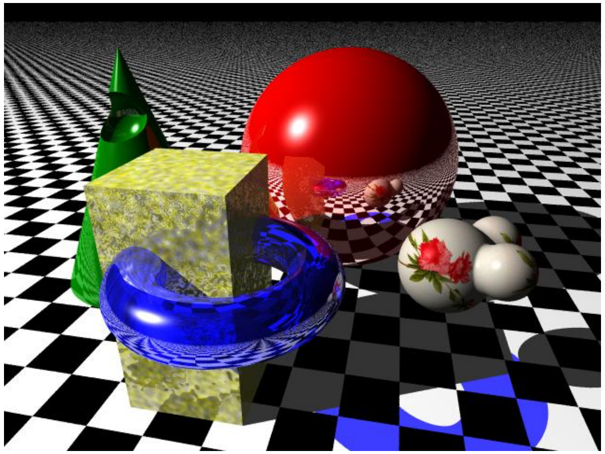
```
cube([2,3,4]);
translate([3,0,0])
  {
     cube([2,3,4]);
   }

color([1,0,0]) cube([2,3,4]);
translate([3,0,0])
    color([0,1,0]) cube([2,3,4];
translate([6,0,0])
    color([0,0,1]) cube([2,3,4]);
```



POVRAY

- Stale but interesting ray tracing software
- Scene description language (SDL)
- Pixar's Renderman

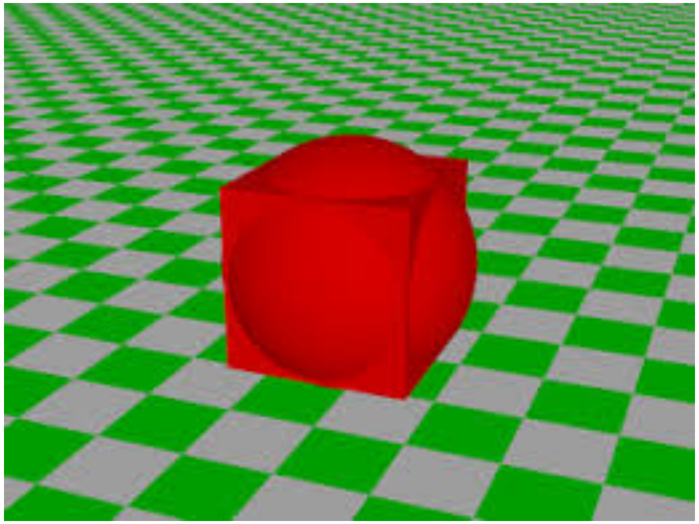


```
#include "colors.inc"
background { color Cyan }
camera {
    location <0, 2, -3>
    look_at <0, 1, 2>
    }
sphere {
   <0, 1, 2>, 2
   texture {
      pigment { color Yellow }
      }
   }
light_source {
    <2, 4, -3>
    color White
     }
```

POVRAY

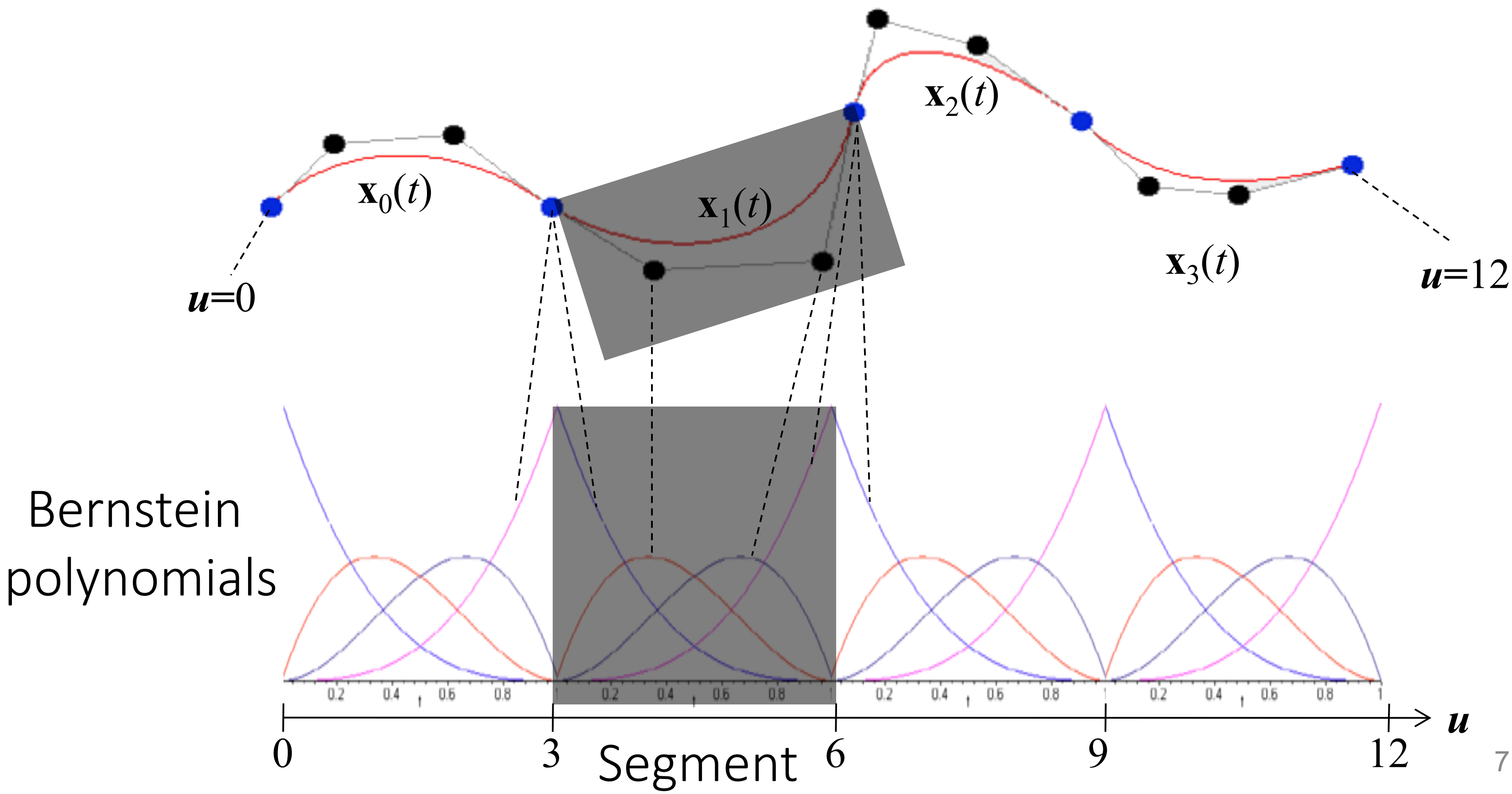
- Support CSG operations

```
union {
  box { <1, 1, 1>, <2, 2, 2> }
  sphere{ <1.5, 1.5, 1.5>, 1 }
  }
```



Piecewise Bézier curves

- Each segment spans four control points
- Each segment contains four Bernstein polynomials
- Each control point belongs to one Bernstein polynomial



Curved surfaces

Curves

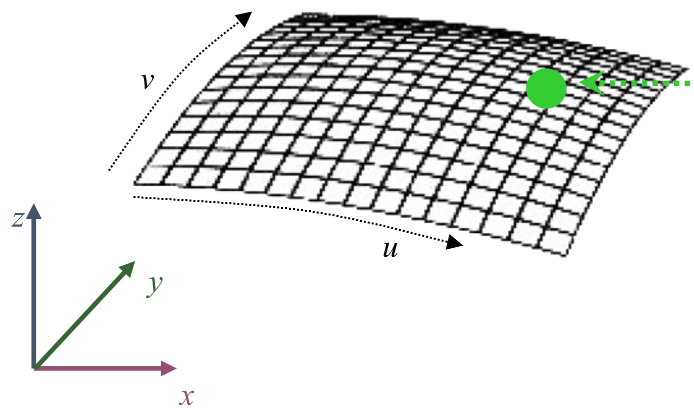
- Described by a 1D series of control points
- A function $\mathbf{x}(t)$
- Segments joined together to form a longer curve

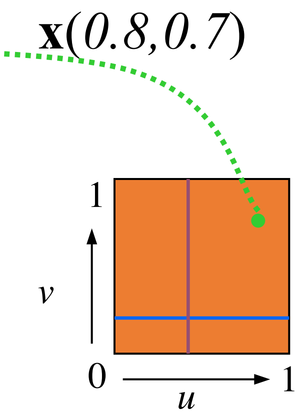
Surfaces

- 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

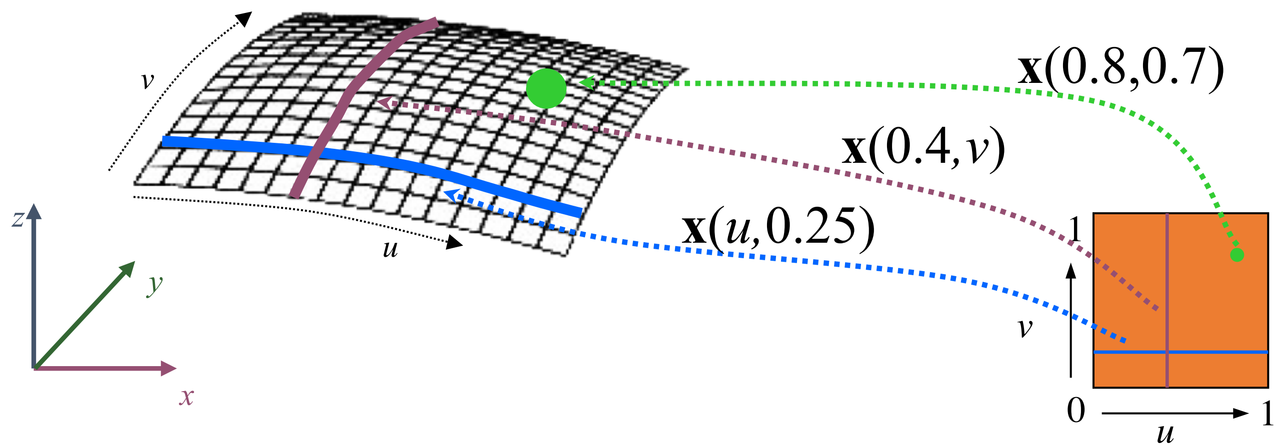




2D parameter domain

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

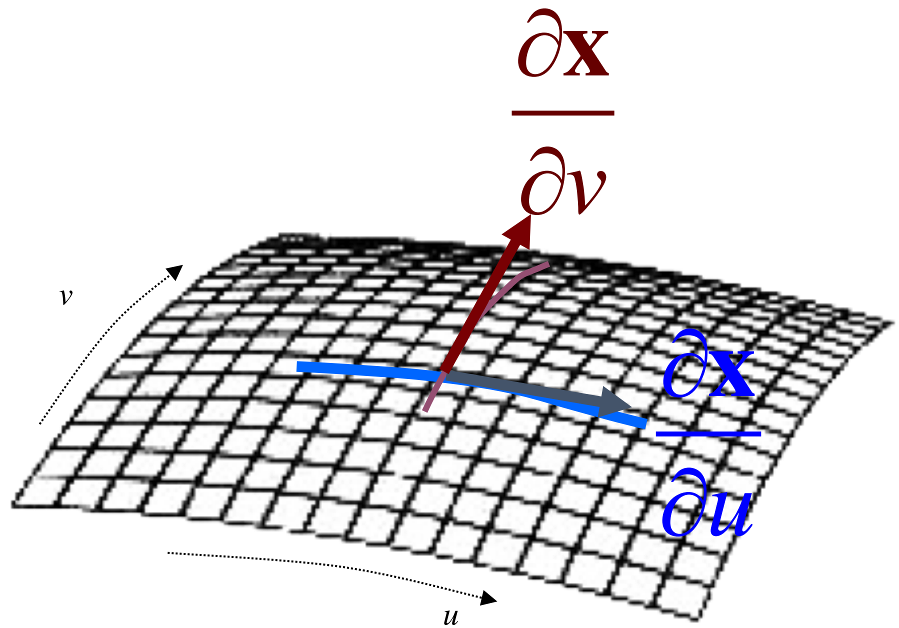


2D parameter domain

- 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 is one pair of parametric curves that go through 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: not necessarily perpendicular to each other



Notation

- Tangent along u direction

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

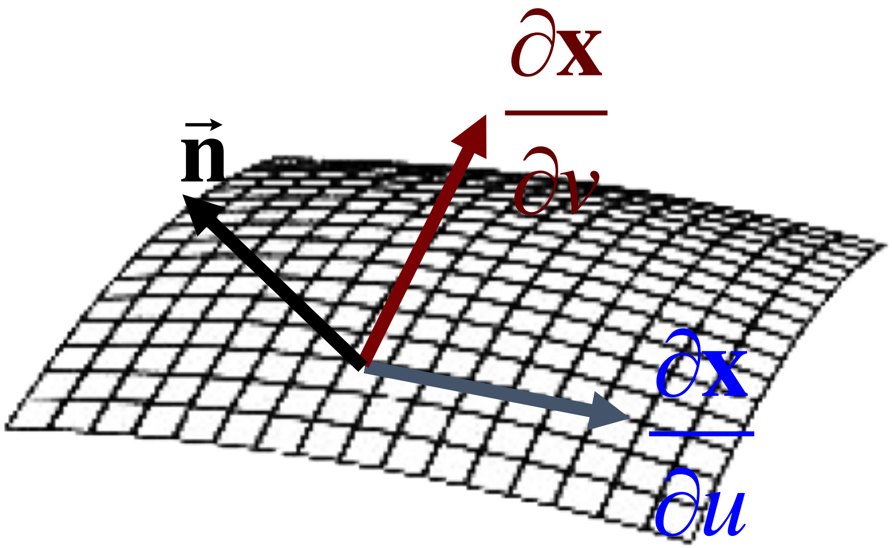
- Tangent along v direction

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

- Tangents are vector valued functions, i.e., vectors!

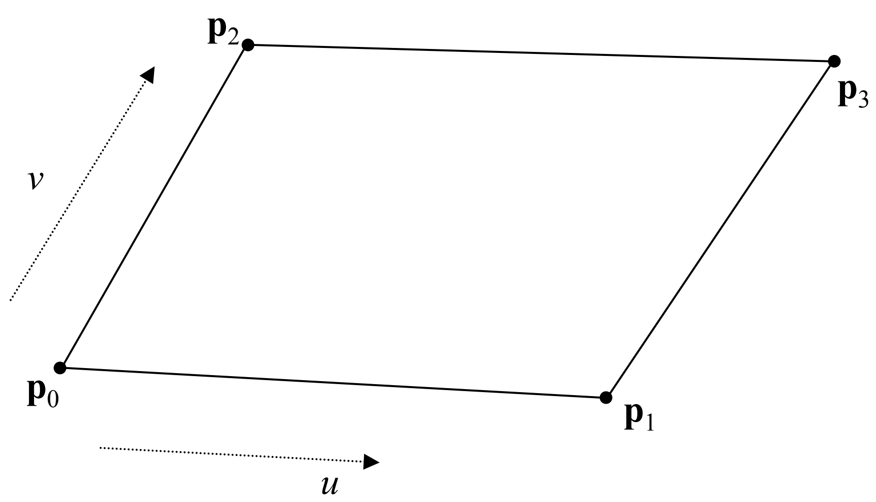
Surface normal

- Cross product of the two tangent vectors
  $\mathbf{x}_u(u,v) \times \mathbf{x}_v(u,v)$
- Order matters (determines normal orientation)
- Usually, want unit normal
  - Need to normalize by dividing through length



Bilinear patch

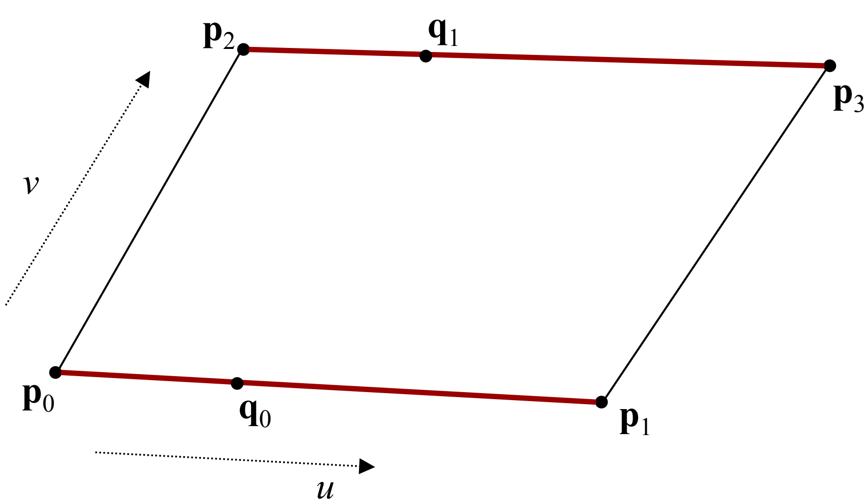
- Control mesh with four points $\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}$
- Compute $\mathbf{x}(u,v)$ using a two-step construction



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:

$$\mathbf{q}_1 = Lerp(u, \mathbf{p}_2, \mathbf{p}_3)$$

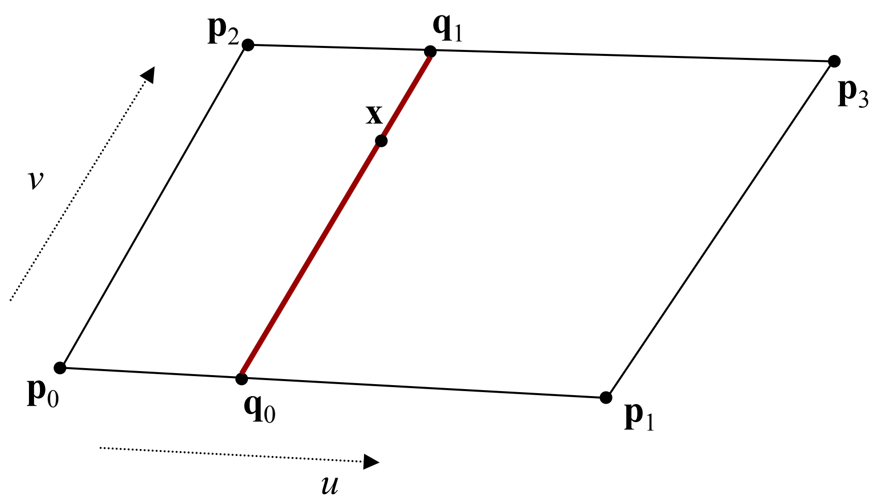


$$\mathbf{q}_0 = Lerp(u, \mathbf{p}_0, \mathbf{p}_1)$$

Bilinear patch (step 2)

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

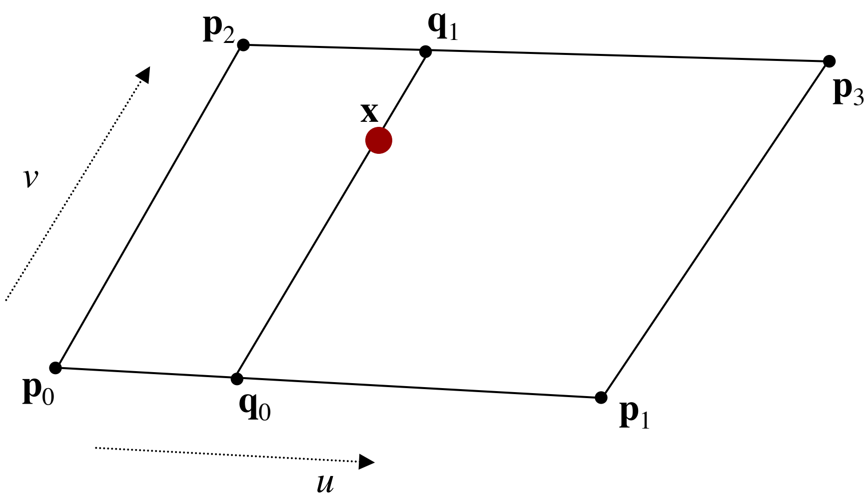
$$\mathbf{x} = Lerp(v, \mathbf{q}_0, \mathbf{q}_1)$$



Bilinear patch

- Combining the steps, we get the full formula

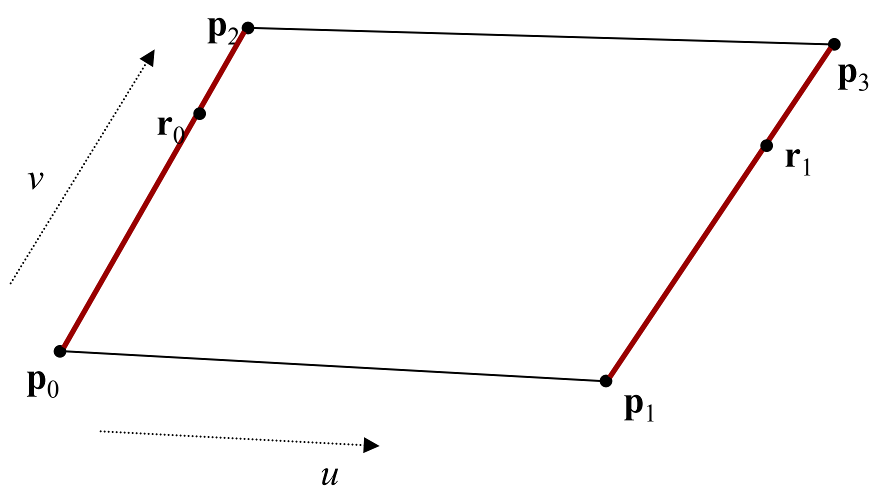
$$\mathbf{x}(u,v) = Lerp(v, Lerp(u,\mathbf{p}_0,\mathbf{p}_1), Lerp(u,\mathbf{p}_2,\mathbf{p}_3))$$



Bilinear patch

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

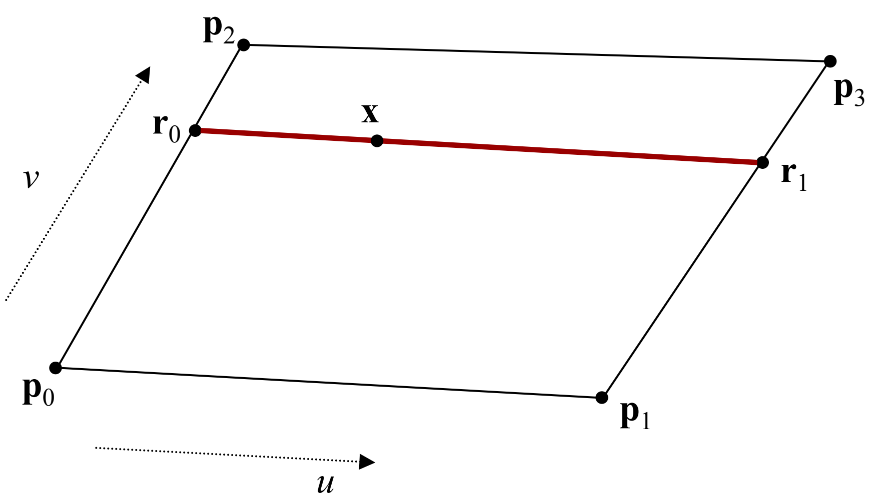
$$\mathbf{r}_0 = Lerp(v, \mathbf{p}_0, \mathbf{p}_2) \qquad \mathbf{r}_1 = Lerp(v, \mathbf{p}_1, \mathbf{p}_3)$$



Bilinear patch

- Consider that $\mathbf{r_0}$, $\mathbf{r_1}$ define a line segment
- Evaluate it using u to get $\mathbf{x}$

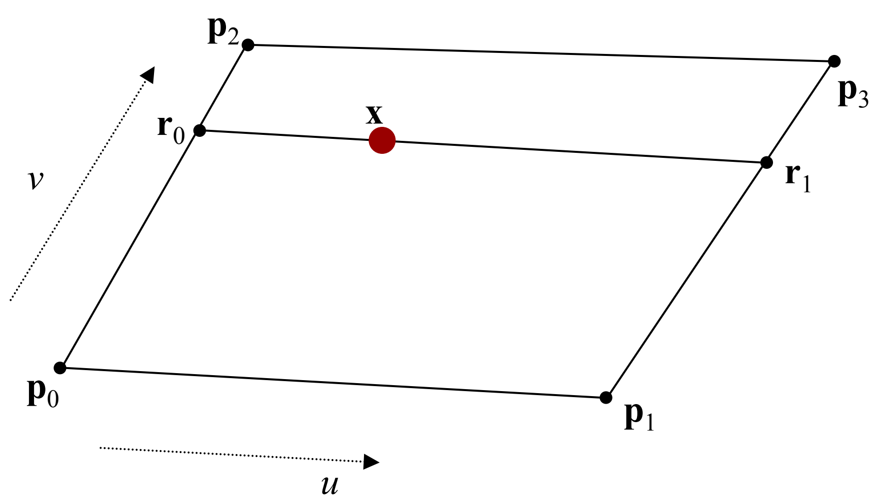
$$\mathbf{x} = Lerp(u, \mathbf{r}_0, \mathbf{r}_1)$$



Bilinear patch

- The full formula for the *v* direction first:

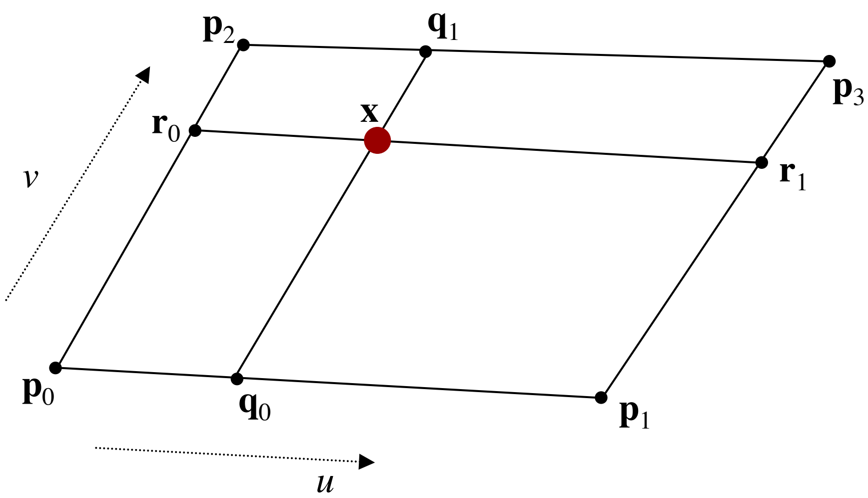
$$\mathbf{x}(u,v) = Lerp(u, Lerp(v, \mathbf{p}_0, \mathbf{p}_2), Lerp(v, \mathbf{p}_1, \mathbf{p}_3))$$



Bilinear patch

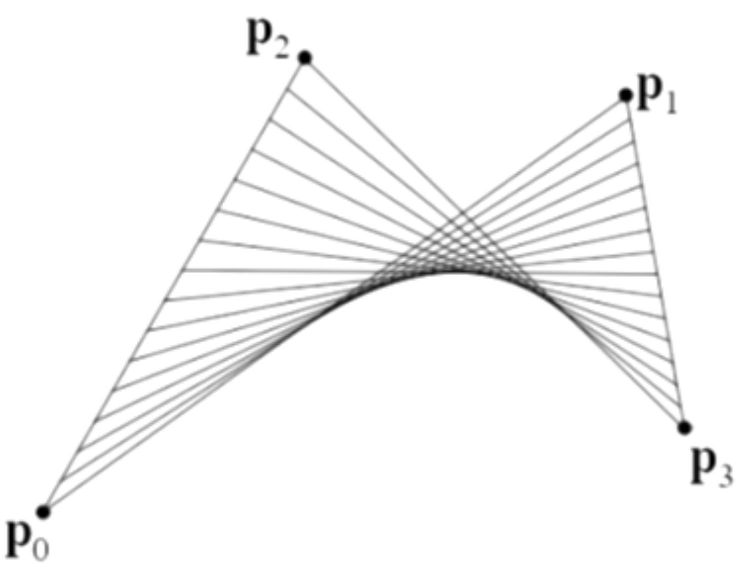
- It works out the same either way!

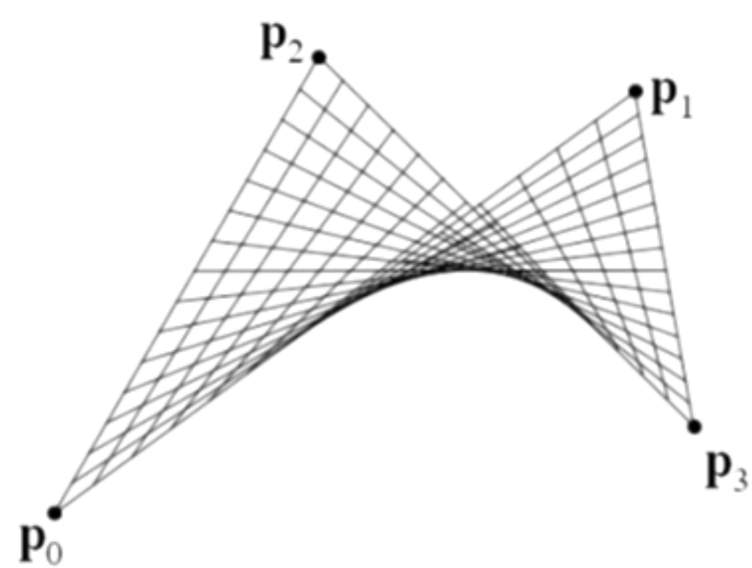
$$\mathbf{x}(u,v) = Lerp(v, Lerp(u,\mathbf{p}_0,\mathbf{p}_1), Lerp(u,\mathbf{p}_2,\mathbf{p}_3))$$
$$\mathbf{x}(u,v) = Lerp(u, Lerp(v,\mathbf{p}_0,\mathbf{p}_2), Lerp(v,\mathbf{p}_1,\mathbf{p}_3))$$



Bilinear patch

- Visualization





Bilinear patches

- Weighted sum of control points

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

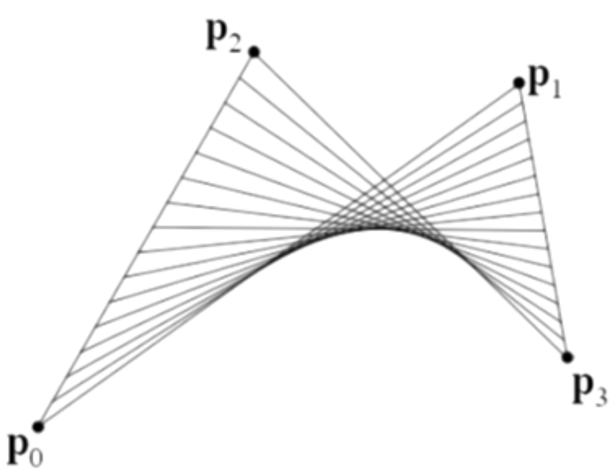
- Bilinear polynomial

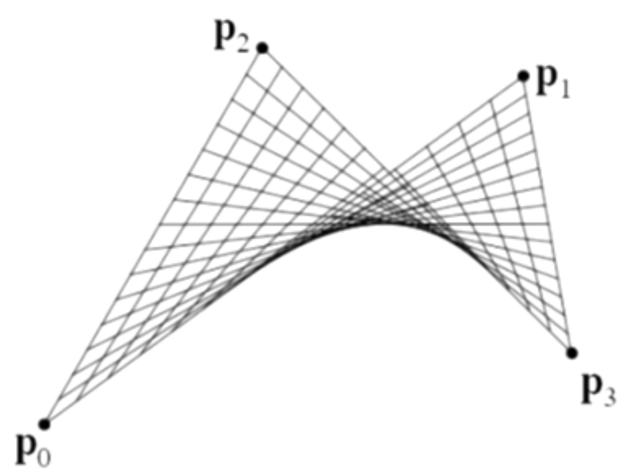
$$\mathbf{x}(u,v) = (\mathbf{p}_0 - \mathbf{p}_1 - \mathbf{p}_2 + \mathbf{p}_3)uv + (\mathbf{p}_1 - \mathbf{p}_0)u + (\mathbf{p}_2 - \mathbf{p}_0)v + \mathbf{p}_0$$

- Matrix form exists, too

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 coplanar, get a curved surface
  - saddle shape, AKA hyperbolic paraboloid
- The parametric curves are all straight line segments!
  - a (doubly) *ruled surface:* has (two) straight lines through every point

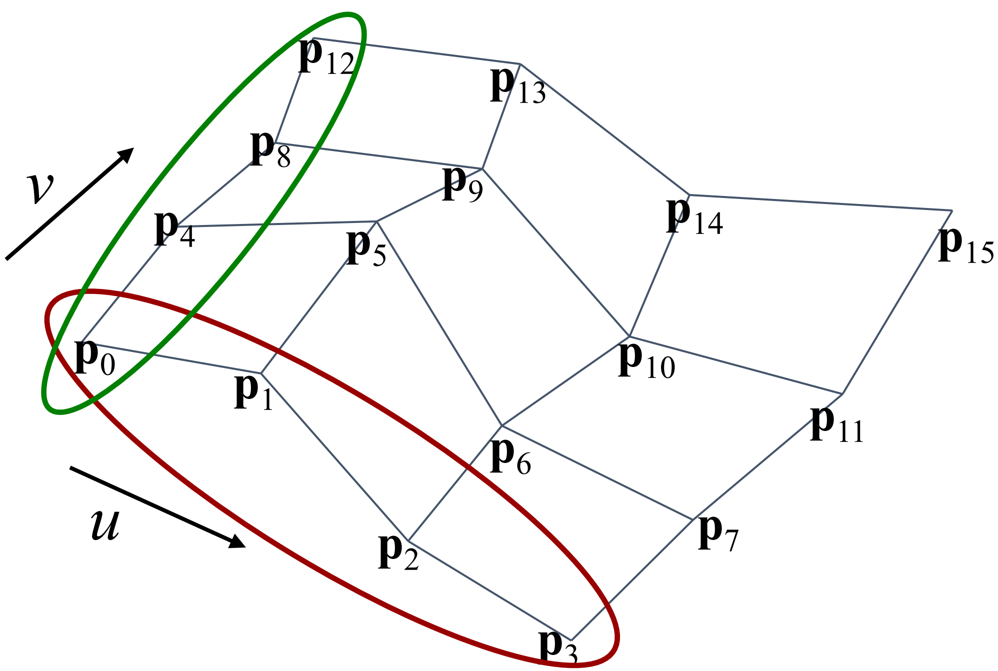




- N

Bicubic Bézier patch

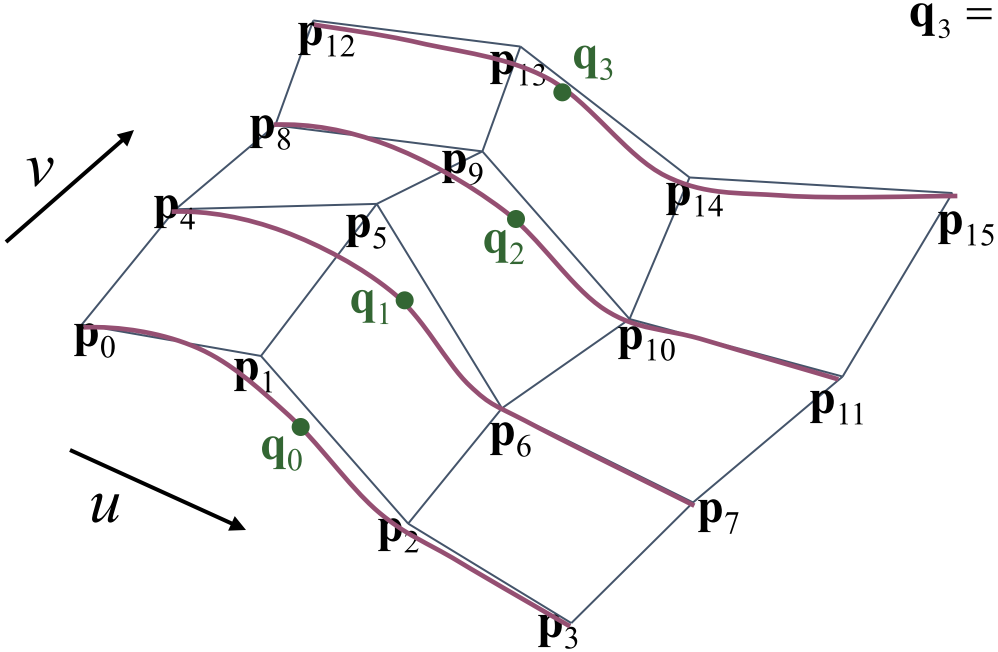
- Grid of 4x4 control points, $\mathbf{p}_0$ through $\mathbf{p}_{15}$
- Four rows of control points define Bézier curves along u
  $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$; $\mathbf{p}_4, \mathbf{p}_5, \mathbf{p}_6, \mathbf{p}_7$; $\mathbf{p}_8, \mathbf{p}_9, \mathbf{p}_{10}, \mathbf{p}_{11}$;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



Bicubic Bézier patch (step 1)

- Evaluate four u-direction Bézier curves at u
- Get intermediate points $\mathbf{q}_0 \ldots \mathbf{q}_3$

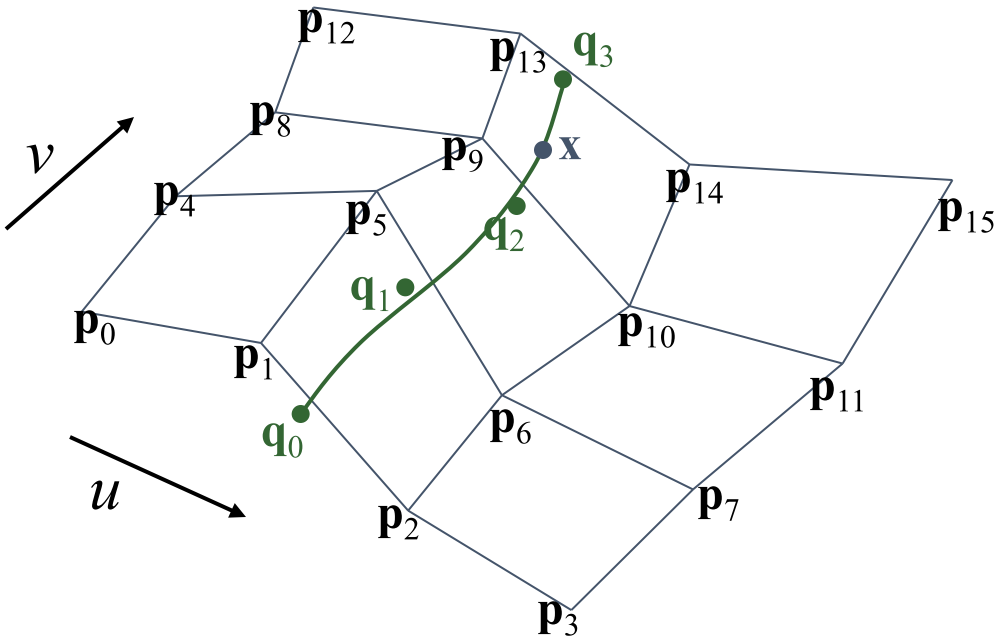
$$\mathbf{q}_0 = Bez(u, \mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)$$
$$\mathbf{q}_1 = Bez(u, \mathbf{p}_4, \mathbf{p}_5, \mathbf{p}_6, \mathbf{p}_7)$$
$$\mathbf{q}_2 = Bez(u, \mathbf{p}_8, \mathbf{p}_9, \mathbf{p}_{10}, \mathbf{p}_{11})$$
$$\mathbf{q}_3 = Bez(u, \mathbf{p}_{12}, \mathbf{p}_{13},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



Bicubic Bézier patch (step 2)

- Points $\mathbf{q}_0 \ldots \mathbf{q}_3$ define a Bézier curve
- Evaluate it at v

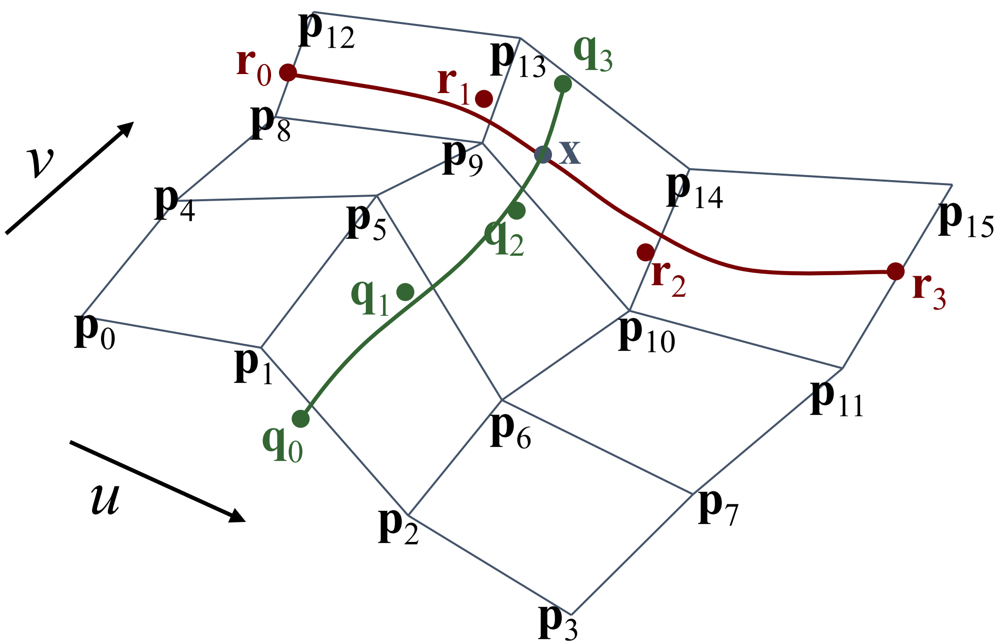
$$\mathbf{x}(u,v) = Bez(v, \mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3)$$



Bicubic Bézier patch

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

$$
\begin{aligned}
\mathbf{q}_0 &= Bez(u,\mathbf{p}_0,\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3) & \qquad \mathbf{r}_0 &= Bez(v,\mathbf{p}_0,\mathbf{p}_4,\mathbf{p}_8,\mathbf{p}_{12}) \\
\mathbf{q}_1 &= Bez(u,\mathbf{p}_4,\mathbf{p}_5,\mathbf{p}_6,\mathbf{p}_7) & \mathbf{r}_1 &= Bez(v,\mathbf{p}_1,\mathbf{p}_5,\mathbf{p}_9,\mathbf{p}_{13}) \\
\mathbf{q}_2 &= Bez(u,\mathbf{p}_8,\mathbf{p}_9,\mathbf{p}_{10},\mathbf{p}_{11}) \quad \Leftrightarrow & \mathbf{r}_2 &= Bez(v,\mathbf{p}_2,\mathbf{p}_6,\mathbf{p}_{10},\mathbf{p}_{14}) \\
\mathbf{q}_3 &= Bez(u,\mathbf{p}_{12},\mathbf{p}_{13},\mathbf{p}_{14},\mathbf{p}_{15}) & \mathbf{r}_3 &= Bez(v,\mathbf{p}_3,\mathbf{p}_7,\mathbf{p}_{11},\mathbf{p}_{15}) \\
\mathbf{x}(u,v) &= Bez(v,\mathbf{q}_0,\mathbf{q}_1,\mathbf{q}_2,\mathbf{q}_3) & \mathbf{x}(u,v) &= Bez(u,\mathbf{r}_0,\mathbf{r}_1,\mathbf{r}_2,\mathbf{r}_3)
\end{aligned}
$$



Tensor product formulation

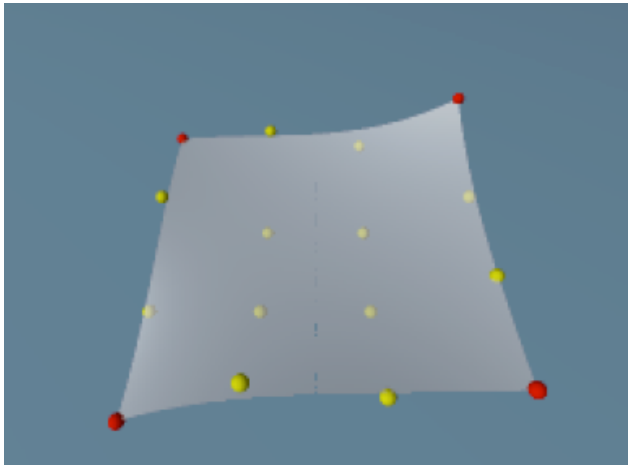
- Corresponds to weighted average formulation
- Construct two-dimensional weighting function as product of two one-dimensional functions
  - Bernstein polynomials B_i, B_j as for curves

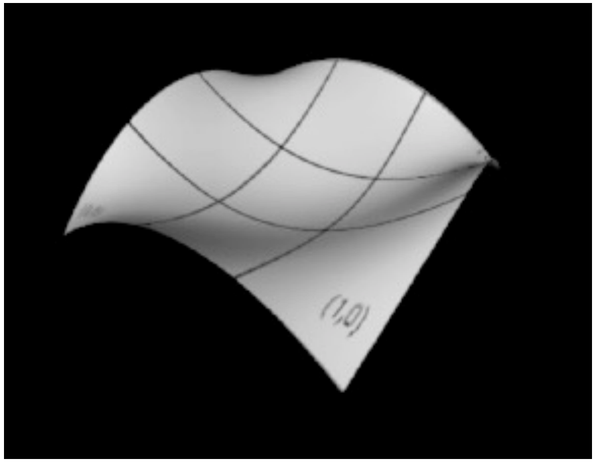
- Same tensor product construction applies to higher order Bézier and NURBS surfaces

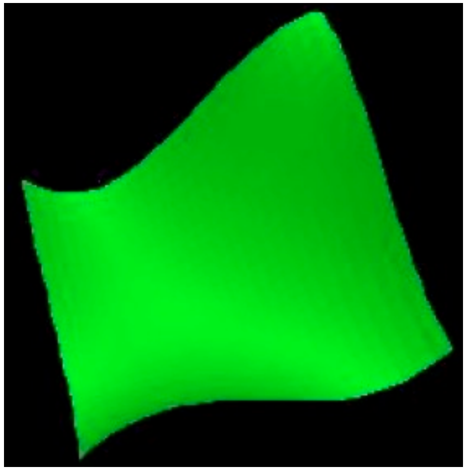
$$\mathbf{x}(u,v) = \sum_i \sum_j \mathbf{p}_{i,j} B_i(u) B_j(v)$$

Bicubic Bézier patch: 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

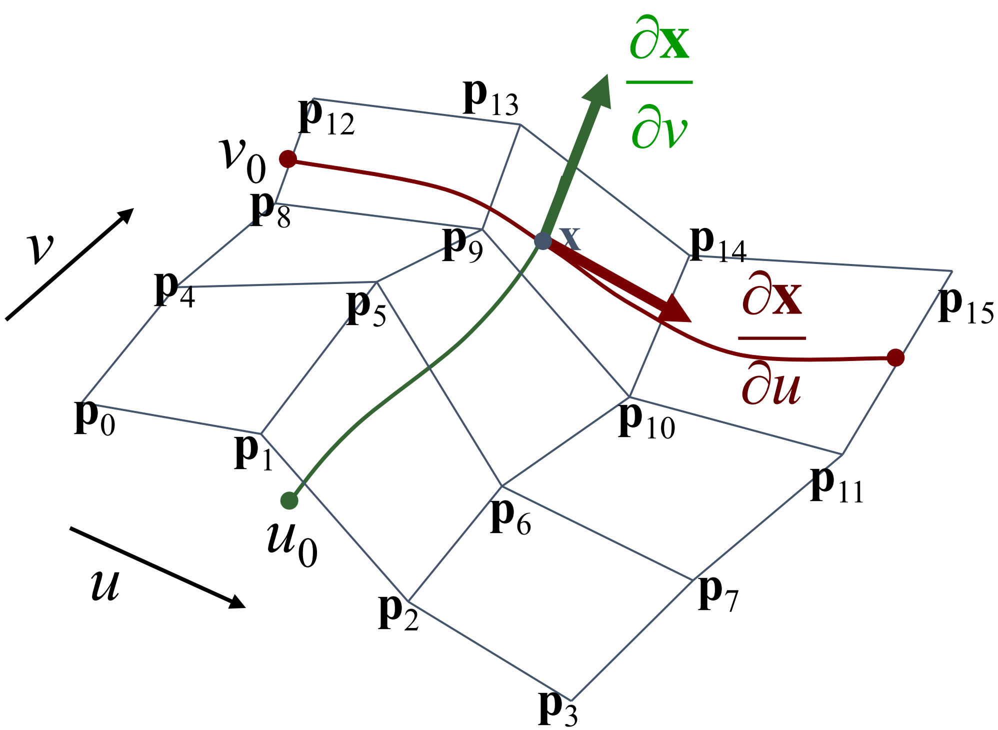






Tangents of Bézier patch

- 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 Bézier patch

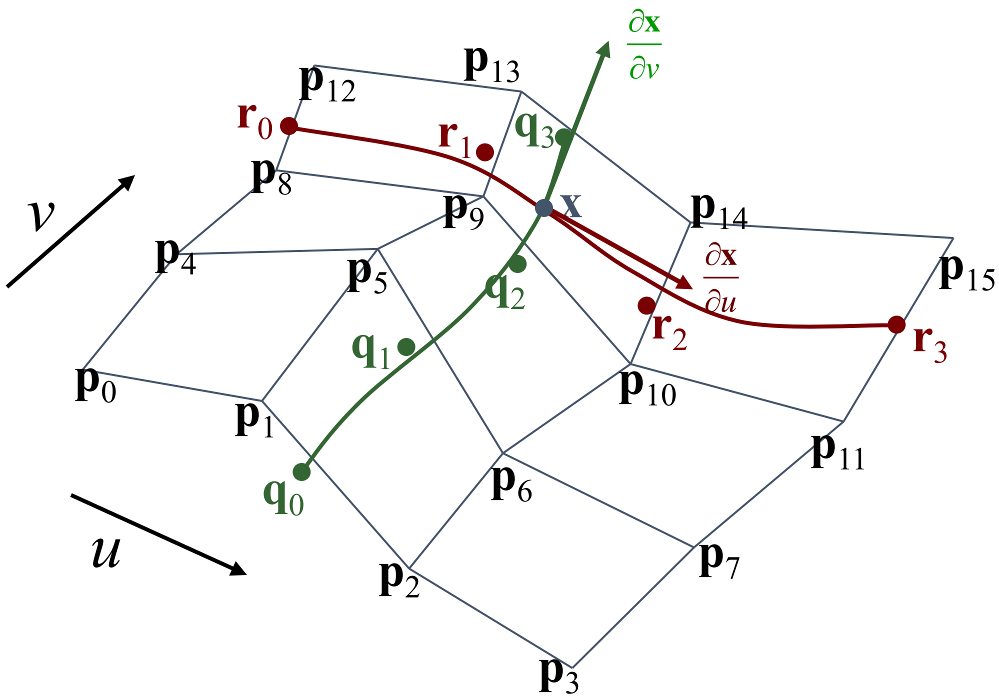
$$\mathbf{q}_0 = Bez(u, \mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3) \qquad \mathbf{r}_0 = Bez(v, \mathbf{p}_0, \mathbf{p}_4, \mathbf{p}_8, \mathbf{p}_{12})$$

$$\mathbf{q}_1 = Bez(u, \mathbf{p}_4, \mathbf{p}_5, \mathbf{p}_6, \mathbf{p}_7) \qquad \mathbf{r}_1 = Bez(v, \mathbf{p}_1, \mathbf{p}_5, \mathbf{p}_9, \mathbf{p}_{13})$$

$$\mathbf{q}_2 = Bez(u, \mathbf{p}_8, \mathbf{p}_9, \mathbf{p}_{10}, \mathbf{p}_{11}) \qquad \mathbf{r}_2 = Bez(v, \mathbf{p}_2, \mathbf{p}_6, \mathbf{p}_{10}, \mathbf{p}_{14})$$

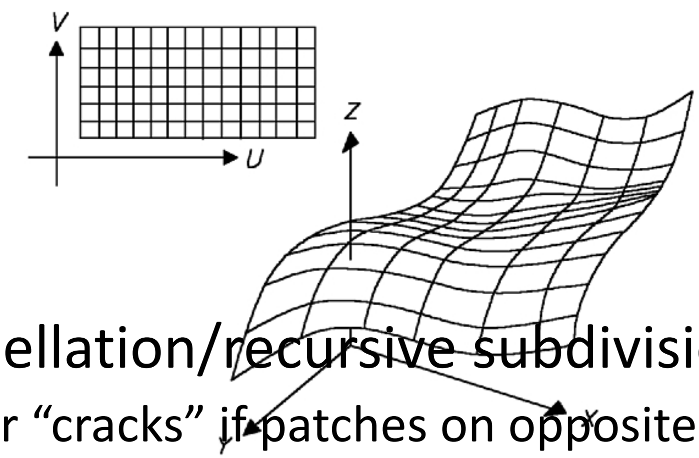
$$\mathbf{q}_3 = Bez(u, \mathbf{p}_{12}, \mathbf{p}_{13}, \mathbf{p}_{14}, \mathbf{p}_{15}) \qquad \mathbf{r}_3 = Bez(v, \mathbf{p}_3, \mathbf{p}_7, \mathbf{p}_{11}, \mathbf{p}_{15})$$

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



Tessellating a Bézier patch

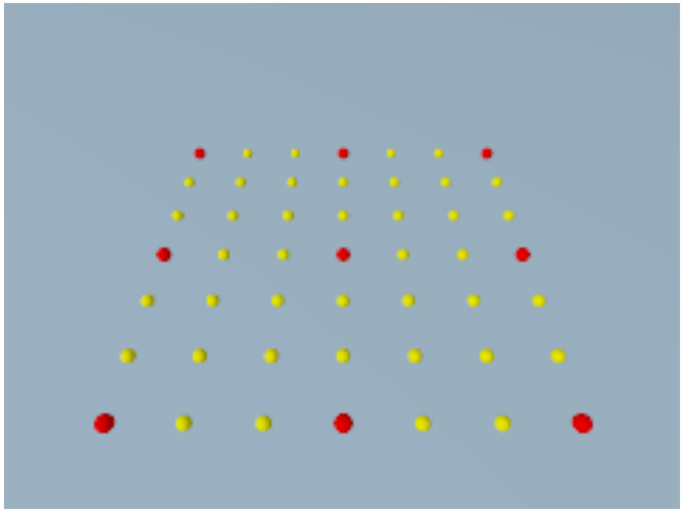
- Uniform tessellation is most straightforward
  - Evaluate points on uniform grid of u, v coordinates
  - Compute tangents at each point, take cross product to get per-vertex normal
  - Draw triangle strips (several choices of direction)



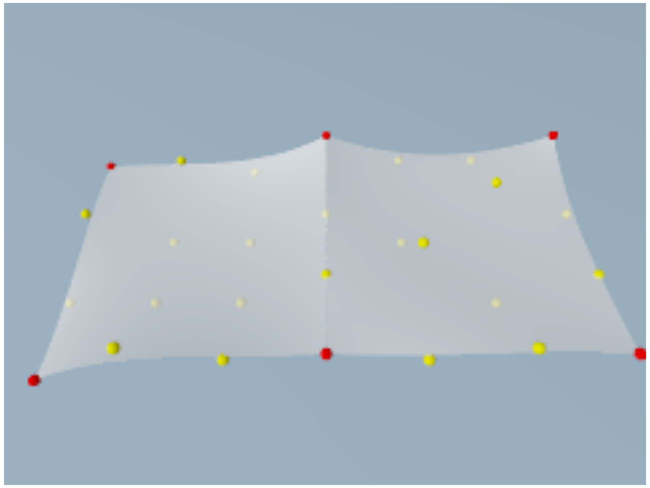
- Adaptive tessellation/recursive subdivision
  - Potential for “cracks” if patches on opposite sides of an edge divide differently
  - Tricky to get right, but can be done

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

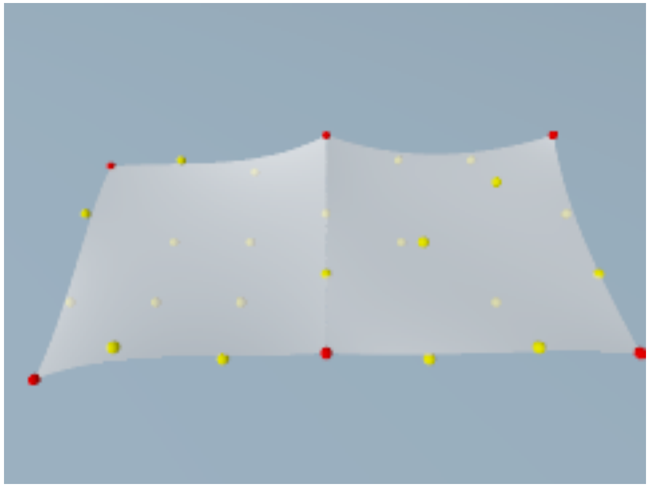


Piecewise Bézier surface

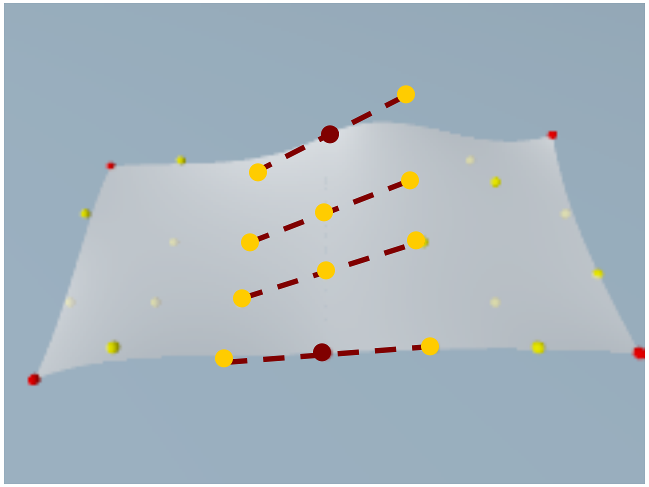
C^1 continuity

- Want parametric curves that cross each edge to have C^1 continuity
  - Handles must be equal-and-opposite across edge

C^0 continuous



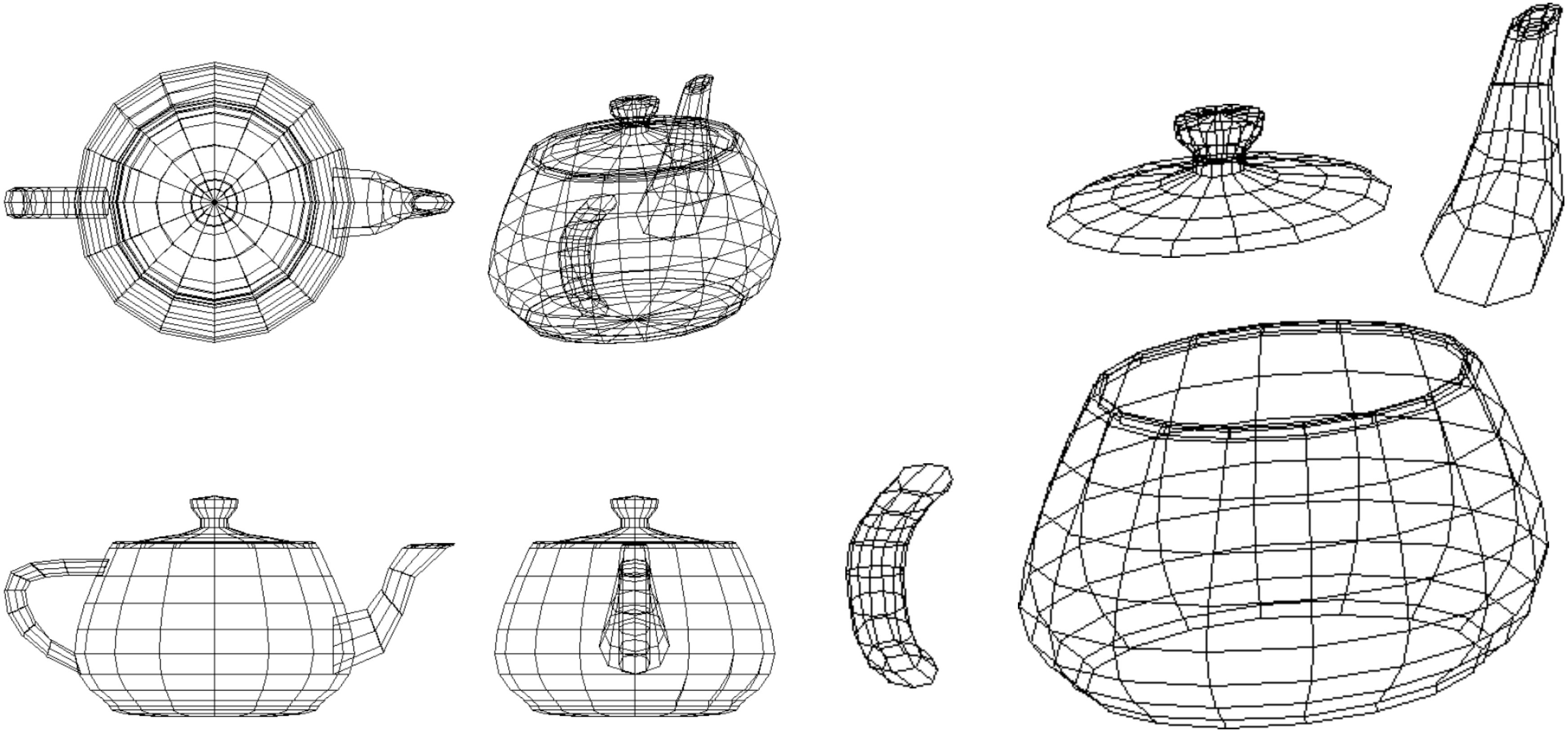
C^1 continuous



[http://www.spiritone.com/~english/cyclopedia/patches.html]

Modeling with Bézier patches

- Original Utah teapot specified as Bézier Patches
  http://en.wikipedia.org/wiki/Utah_teapot

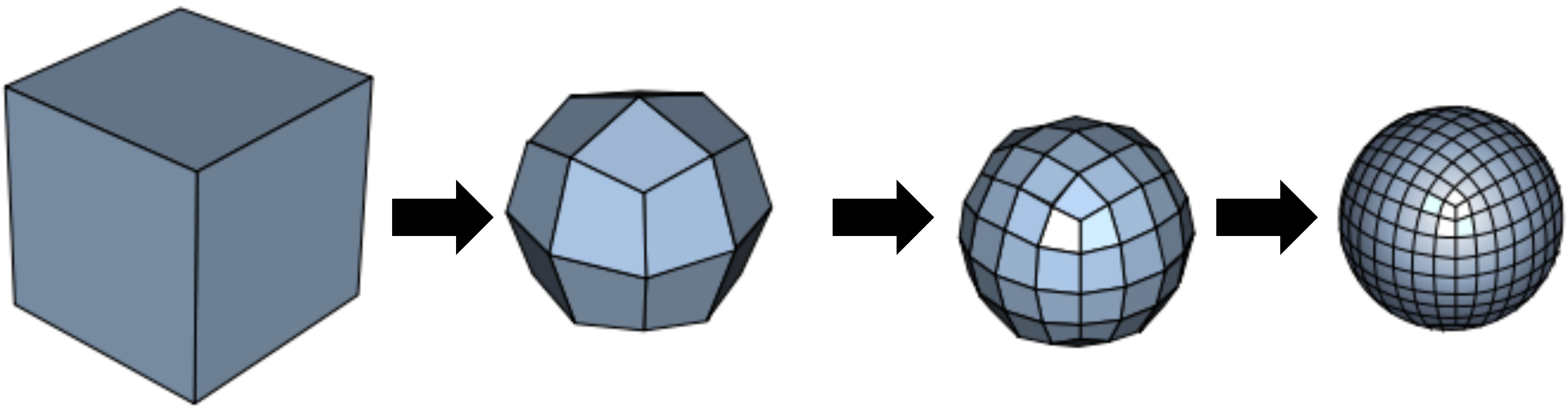


Subdivision surfaces

- Goal
  - Create smooth surfaces from small number of control points, like splines
  - More flexibility for the topology of the control points (not restricted to quadrilateral grid)
- Idea
  - Start with initial coarse polygon mesh
  - Create smooth surface recursively by
    1. Splitting (subdividing) mesh into finer polygons
    2. Smoothing the vertices of the polygons
    3. Repeat from 1.

Subdivision surfaces

http://en.wikipedia.org/wiki/Catmull%E2%80%93Clark_subdivision_surface

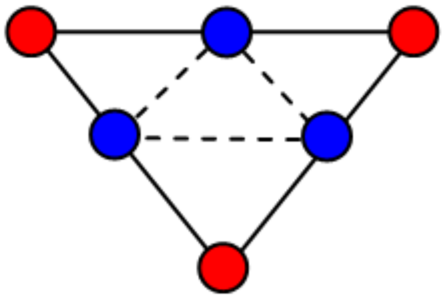


Input mesh | Subdivision & smoothing | Subdivision & smoothing | Subdivision & smoothing

Limit surface

Loop subdivision

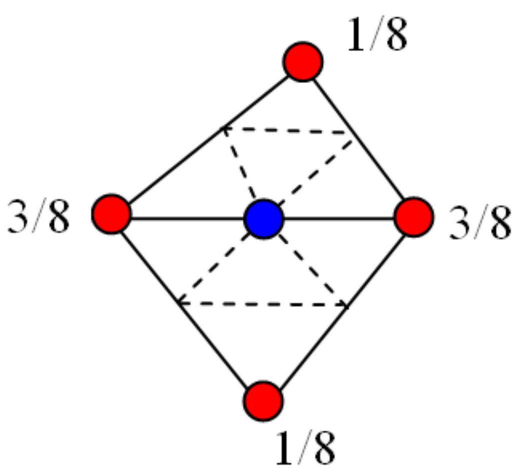
- Subdivision
  - Split each triangle into four

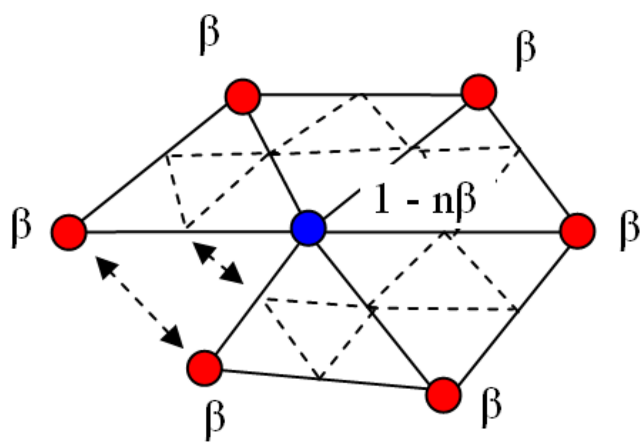


- Smoothing
  - New vertex positions as weighted average of neighbors
  - Different cases

Loop

http://en.wikipedia.org/wiki/Loop_subdivision_surface





Cases for β:

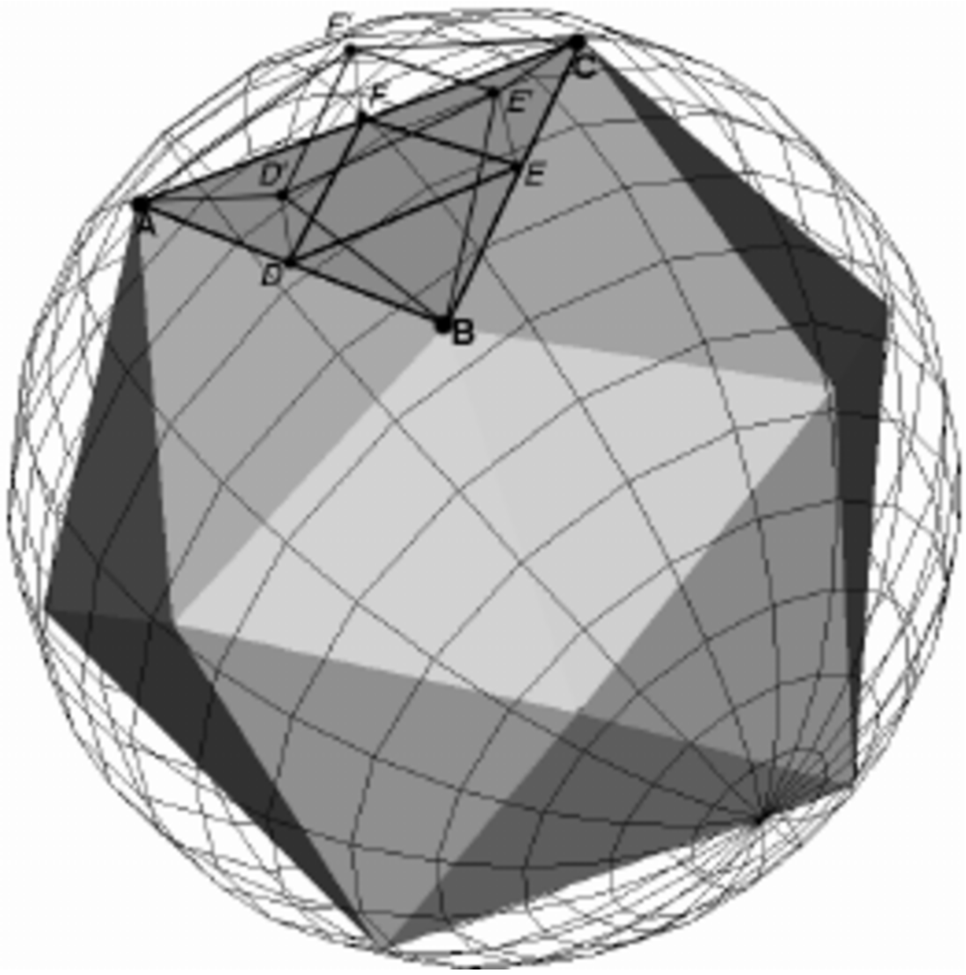
$$\beta = \begin{cases} \dfrac{3}{8n} & n > 3 \\ \dfrac{3}{16} & n = 3 \end{cases}$$

Number of neighbors $\mathbf{n}$

http://graphics.stanford.edu/~mdfisher/subdivision.html

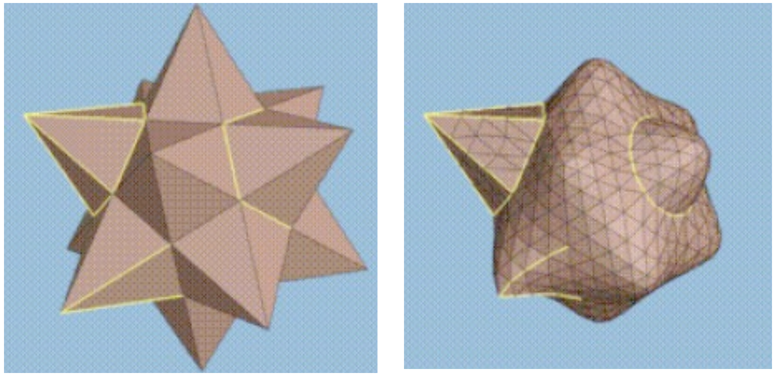
Subdividing sphere

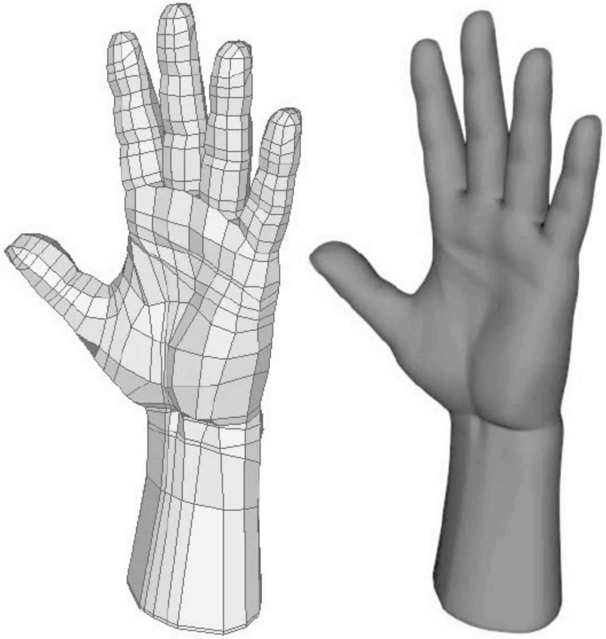
- Divide triangle ABC into four new triangles
- Extend rays to sphere surface to compute new vertices



Subdivision surfaces

- Arbitrary mesh of control points
- Arbitrary topology or connectivity
  - Not restricted to quadrilateral topology
  - No global u,v parameters
- Work by recursively subdividing mesh faces
- Used in particular for character animation
  - One surface rather than collection of patches
  - Can deform geometry without creating cracks





Subdivision surfaces