CS-C3100 Computer Graphics

4.1 Tensor Product Spline Surfaces

Representing Surfaces

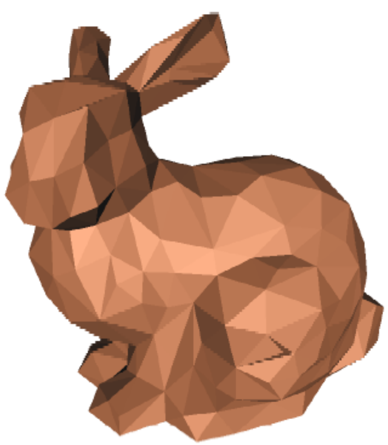
- Triangle meshes ✔
  - Surface analogue of polylines, this is what GPUs draw
- **Tensor Product Splines (this video)**
  - **Surface analogue of spline curves**
- Subdivision surfaces (next video)
- Implicit surfaces
  - $f(x,y,z)=0$
- Procedural
  - e.g. surfaces of revolution, generalized cylinder
- From volume data (medical images, etc.)

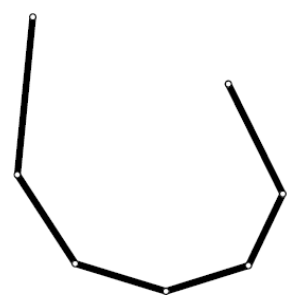
In This Video

- Tensor product spline patches
  - Bézier and B-spline
- Tangents and normals from partial derivatives
- Displacement mapping
- *Extra: matrix & tensor notation for spline patches*
  - Slides only

Triangle Meshes

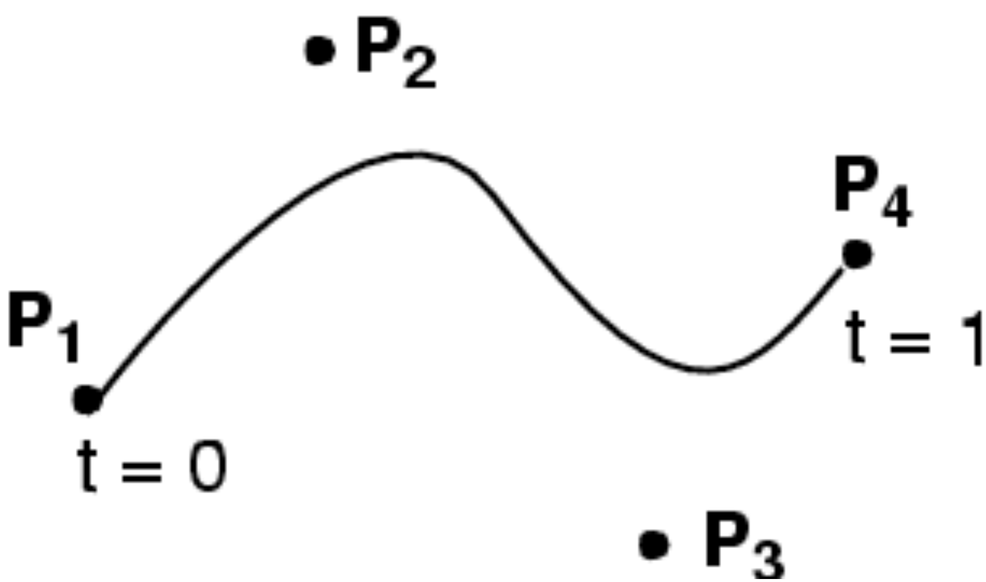
- What you've used so far in Asst 1
- Triangle represented by 3 vertices
- **Pro**: simple, can be rendered directly
- **Cons**: not smooth, needs many triangles to approximate smooth surfaces (tessellation)





Smooth Surfaces?

- $\mathbf{P}(t) = (1-t)^3 \mathbf{P}_1 + 3t(1-t)^2 \mathbf{P}_2 + 3t^2(1-t) \mathbf{P}_3 + t^3 \mathbf{P}_4$



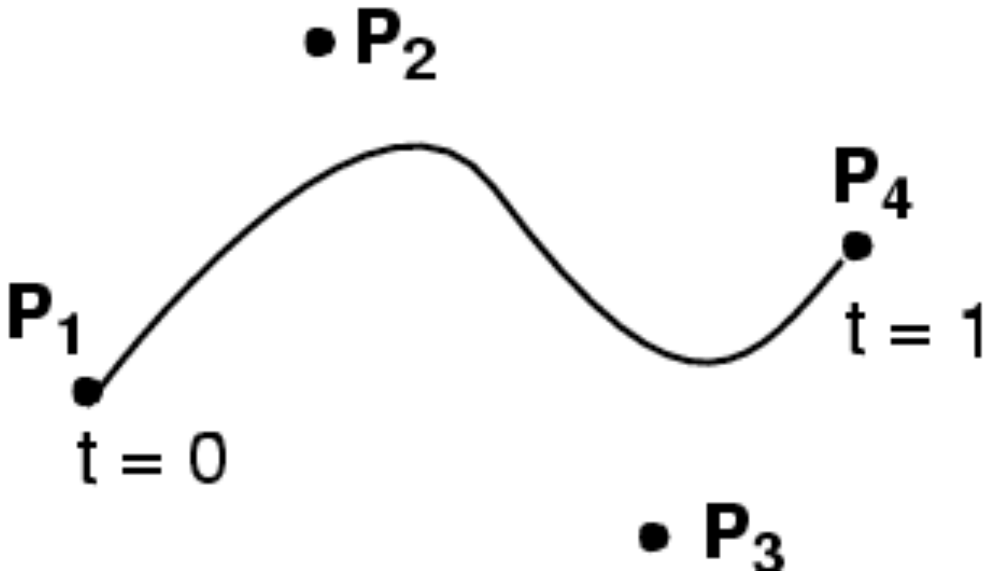
What's the dimensionality of a curve? 1D!

Why? Just one scalar parameter, *t*

What about a surface?

Smooth Surfaces?

- $$\mathbf{P}(t) = (1-t)^3 \, \mathbf{P}_1 + 3t(1-t)^2 \, \mathbf{P}_2 + 3t^2(1-t) \, \mathbf{P}_3 + t^3 \, \mathbf{P}_4$$



What's the dimensionality of a curve? 1D!

Why? Just one scalar parameter, *t*

What about a surface?

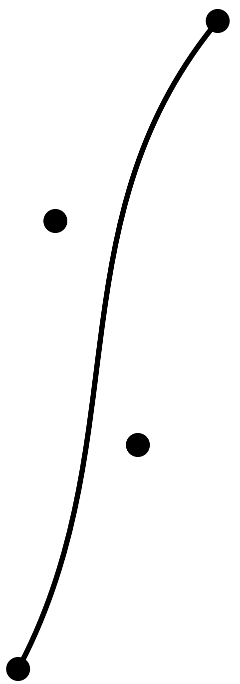
2D!

How to Build Them?

- $\mathbf{P}(u) = (1-u)^3 \mathbf{P}_1 + 3u(1-u)^2 \mathbf{P}_2 + 3u^2(1-u) \mathbf{P}_3 + u^3 \mathbf{P}_4$

$$\begin{aligned}\mathbf{P}(u) = \; & (1-u)^3 && \mathbf{P}_1 \\ + \; & 3u(1-u)^2 && \mathbf{P}_2 \\ + \; & 3u^2(1-u) && \mathbf{P}_3 \\ + \; & u^3 && \mathbf{P}_4\end{aligned}$$

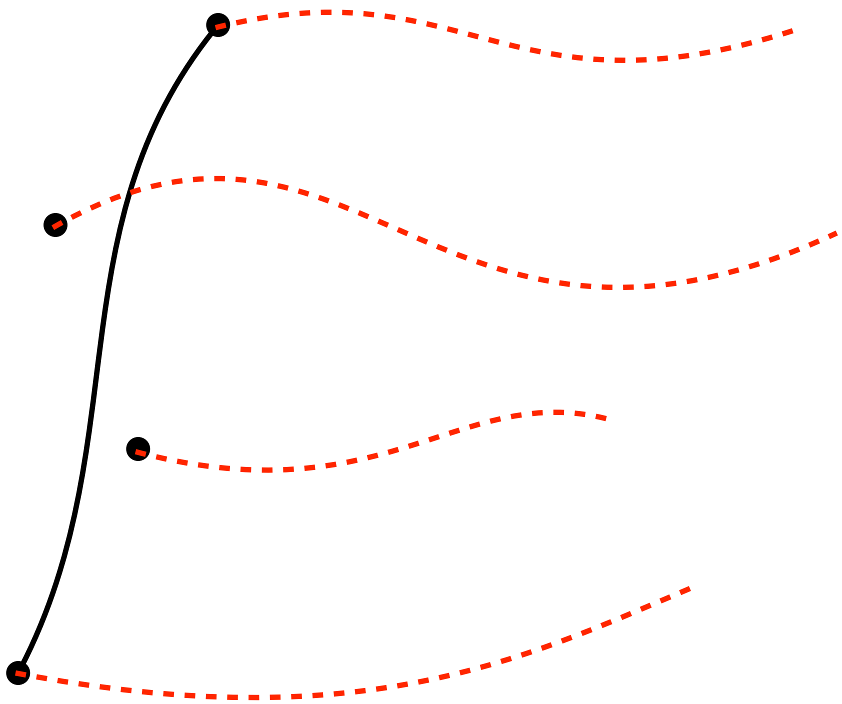
(Note! We relabeled *t* to *u*)



How to Build Them? Here's an Idea

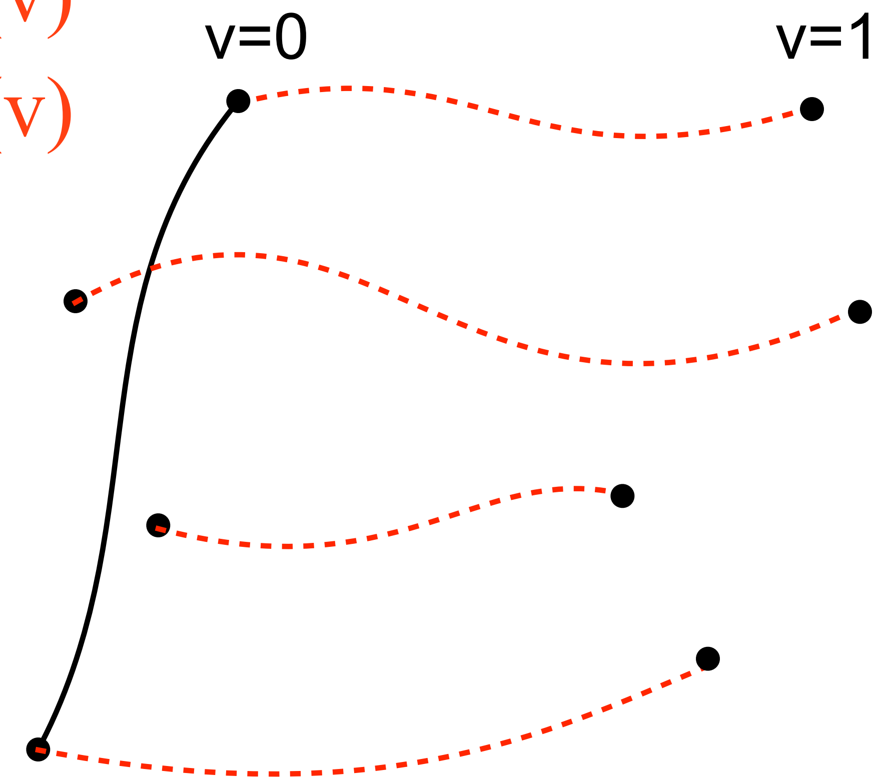
- $$\mathbf{P}(u) = (1-u)^3 \mathbf{P}_1 + 3u(1-u)^2 \mathbf{P}_2 + 3u^2(1-u) \mathbf{P}_3 + u^3 \mathbf{P}_4$$

(Note! We relabeled *t* to *u*)



Here's an Idea

- $$\mathbf{P}(u, v) = (1-u)^3 \mathbf{P}_1(v) + 3u(1-u)^2 \mathbf{P}_2(v) + 3u^2(1-u) \mathbf{P}_3(v) + u^3 \mathbf{P}_4(v)$$
- Let's make the P_is move along curves!

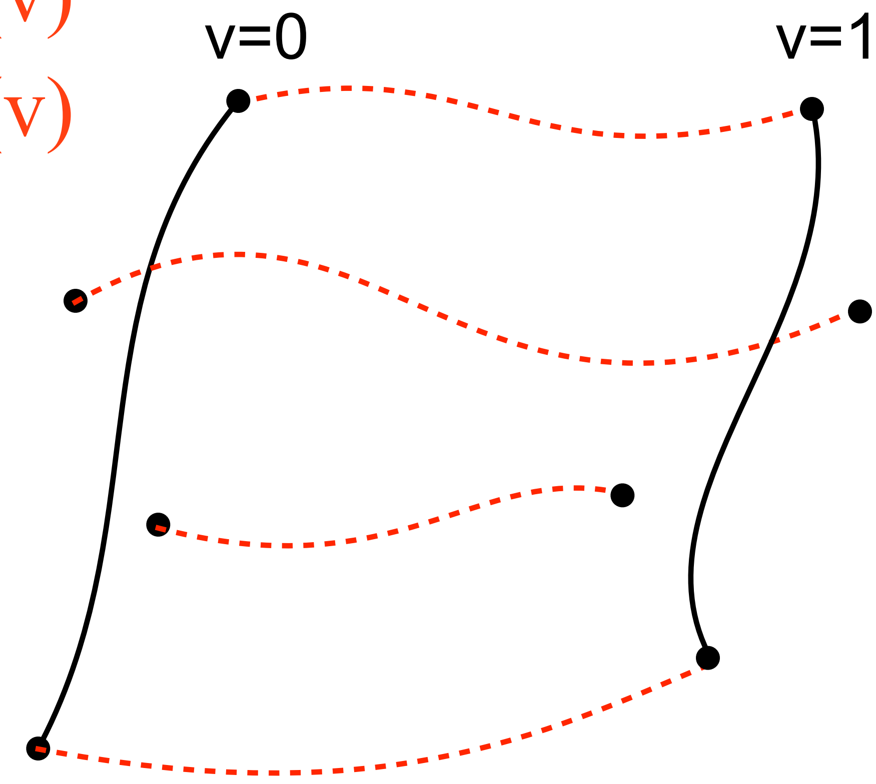


Here's an Idea

- $\mathbf{P}(u, v) = (1-u)^3 \mathbf{P}_1(v) + 3u(1-u)^2 \mathbf{P}_2(v) + 3u^2(1-u) \mathbf{P}_3(v) + u^3 \mathbf{P}_4(v)$

$$\mathbf{P}(u, v) = (1-u)^3\,\mathbf{P}_1(v) + 3u(1-u)^2\,\mathbf{P}_2(v) + 3u^2(1-u)\,\mathbf{P}_3(v) + u^3\,\mathbf{P}_4(v)$$

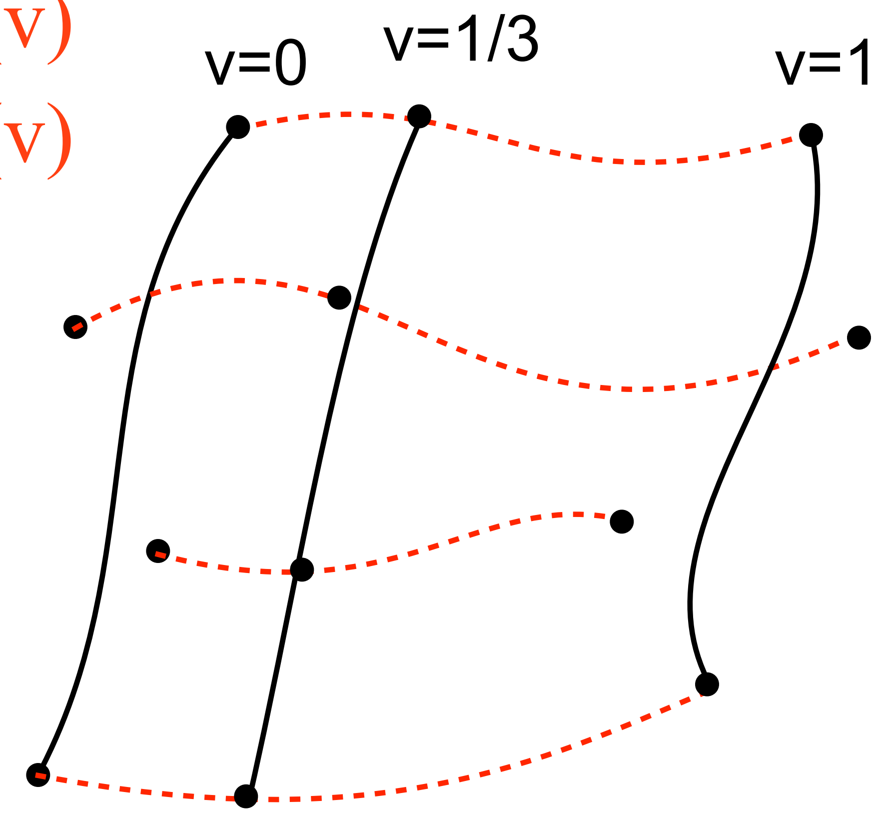
- Let's make the P_is move along curves!



Here's an Idea

- $\mathbf{P}(u, v) = (1-u)^3 \mathbf{P}_1(v) + 3u(1-u)^2 \mathbf{P}_2(v) + 3u^2(1-u) \mathbf{P}_3(v) + u^3 \mathbf{P}_4(v)$

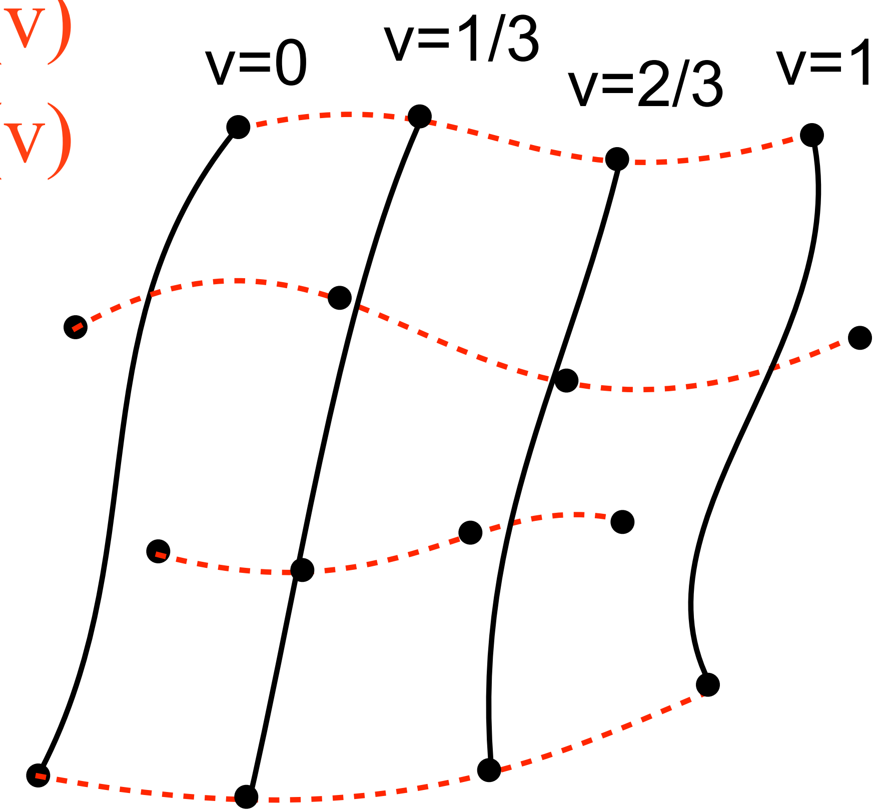
- Let's make the P_is move along curves!



Here's an Idea

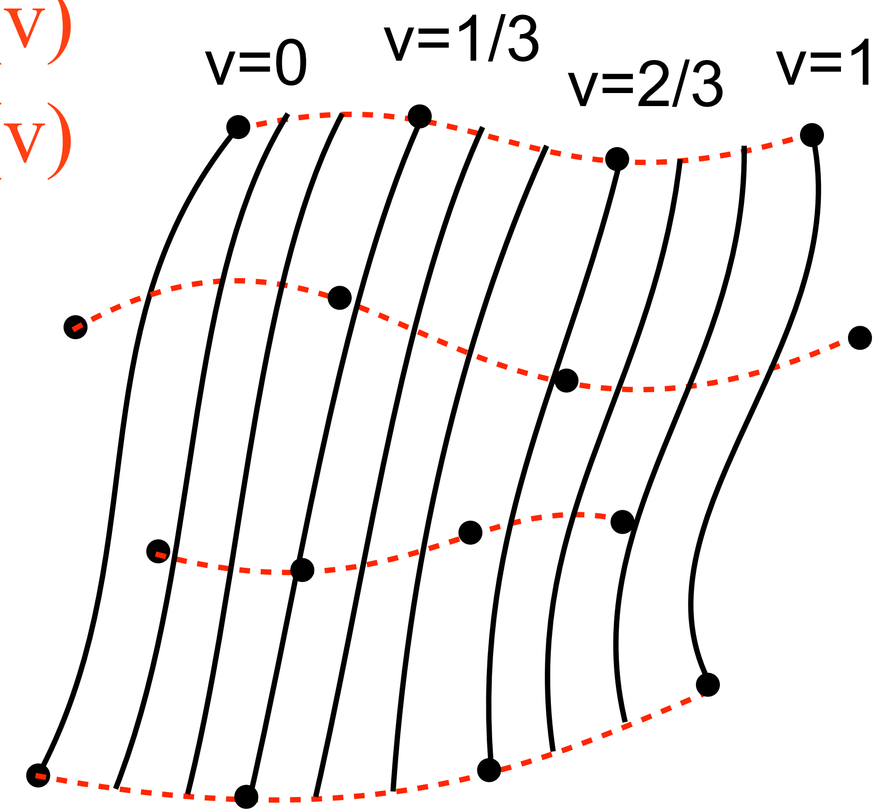
- $\mathbf{P}(u, v) = (1\text{-}u)^3 \quad \mathbf{P}_1(v)$
  $+ \quad 3u(1\text{-}u)^2 \quad \mathbf{P}_2(v)$
  $+ \quad 3u^2(1\text{-}u) \quad \mathbf{P}_3(v)$
  $+ \quad u^3 \quad \mathbf{P}_4(v)$

- Let's make the P_is move along curves!



Here's an Idea

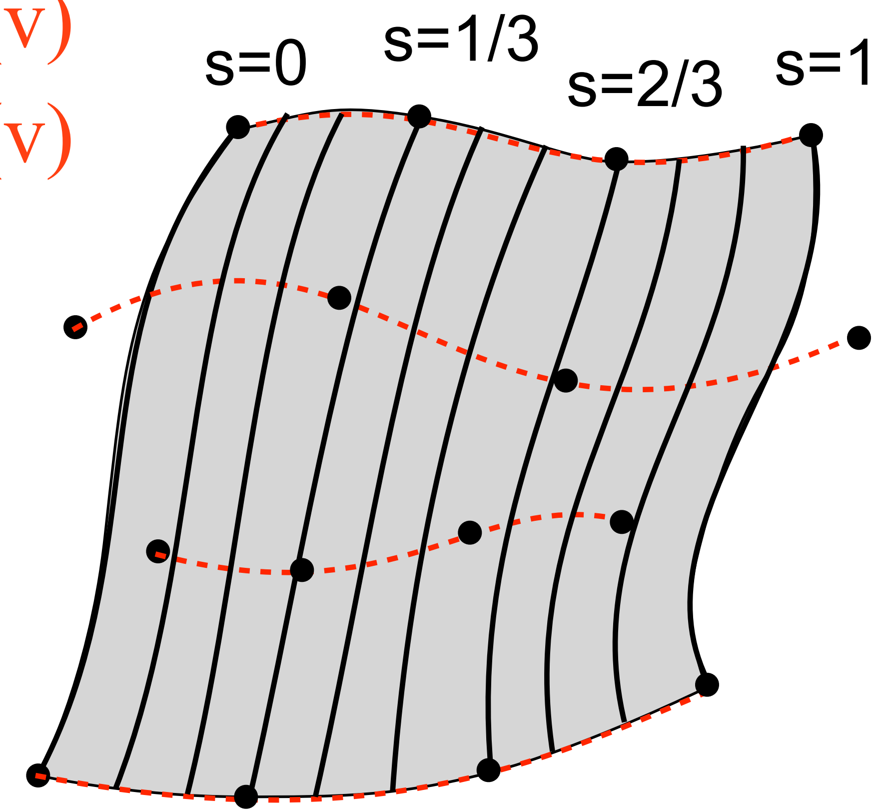
- $\mathbf{P}(u, v) = (1-u)^3 \, \mathbf{P}_1(v) + 3u(1-u)^2 \, \mathbf{P}_2(v) + 3u^2(1-u) \, \mathbf{P}_3(v) + u^3 \, \mathbf{P}_4(v)$
- Let's make the P_is move along curves!



Here's an Idea

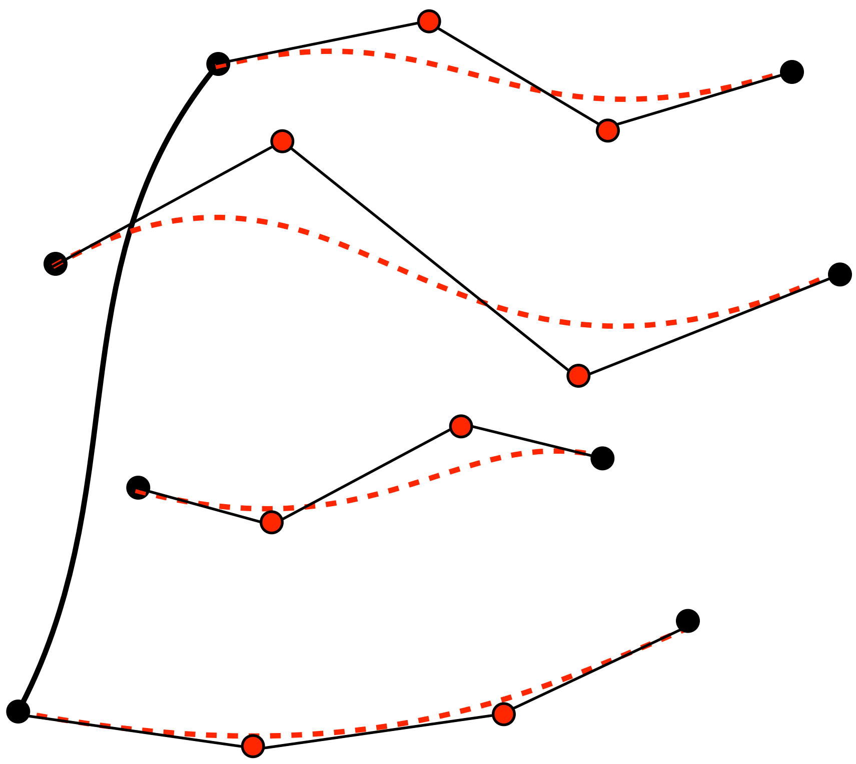
- $\mathbf{P}(u, v) = (1-u)^3 \quad \mathbf{P}_1(v)$
  $+ \quad 3u(1-u)^2 \quad \mathbf{P}_2(v)$
  $+ \quad 3u^2(1-u) \quad \mathbf{P}_3(v)$
  $+ \quad u^3 \quad \mathbf{P}_4(v)$
- Let's make the P_is move along curves!

A 2D surface patch!



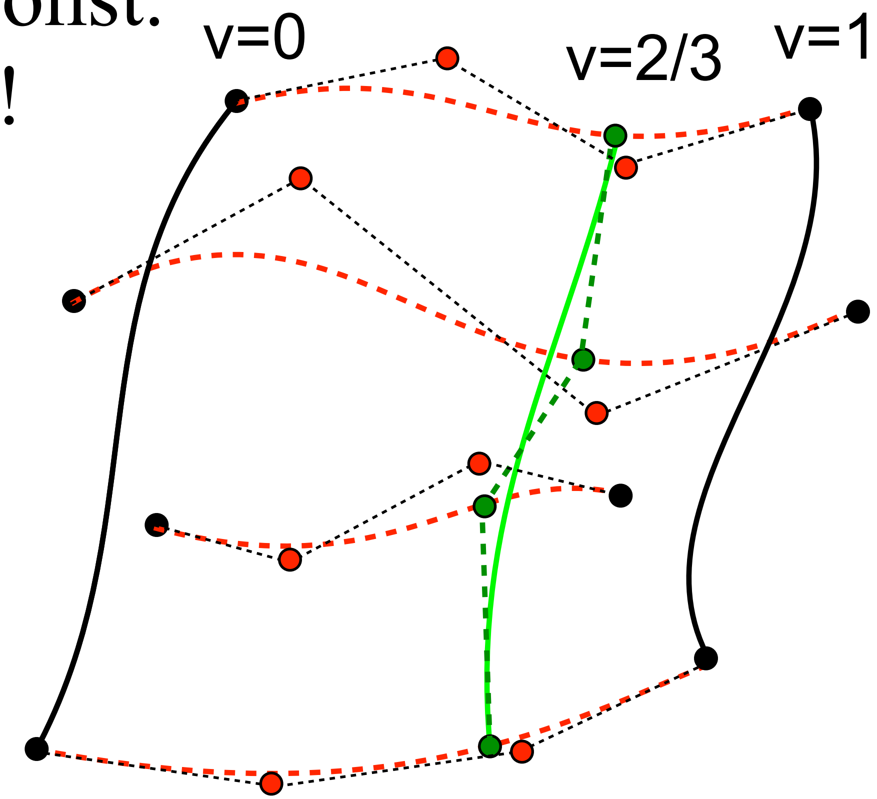
“Tensor Product Bézier Patches”

- In the previous, $\mathbf{P}_i(v)$ were just some curves
- What if we make **them** Bézier curves too?



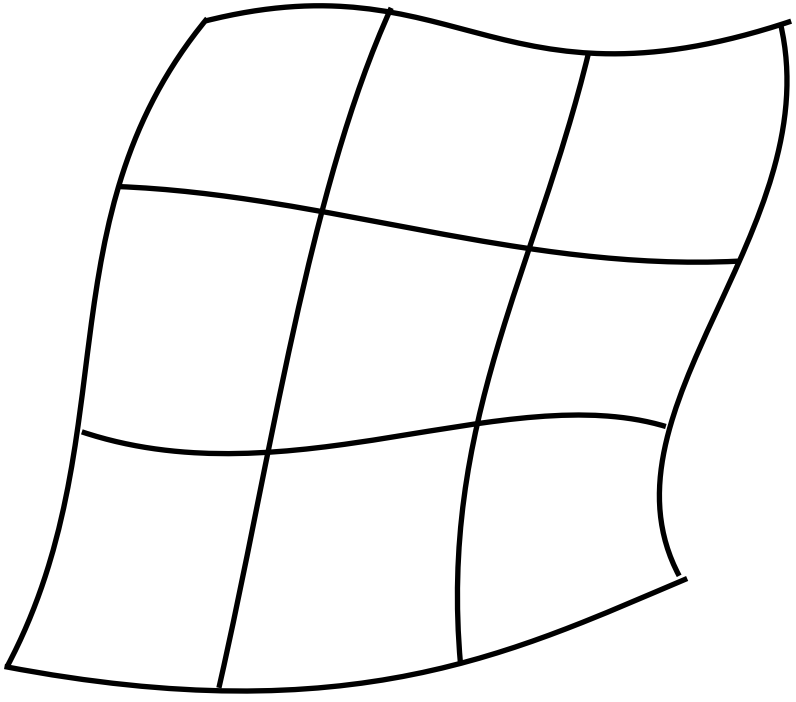
Tensor Product Bézier Patches

- In the previous, $\mathbf{P}_i(v)$ were just some curves
- What if we make **them** Bézier curves too?
- Each *u*=const. **and** *v*=const. curve is a Bézier curve!
- Note that the boundary control points (except corners) are NOT interpolated!



Tensor Product Bézier Patches

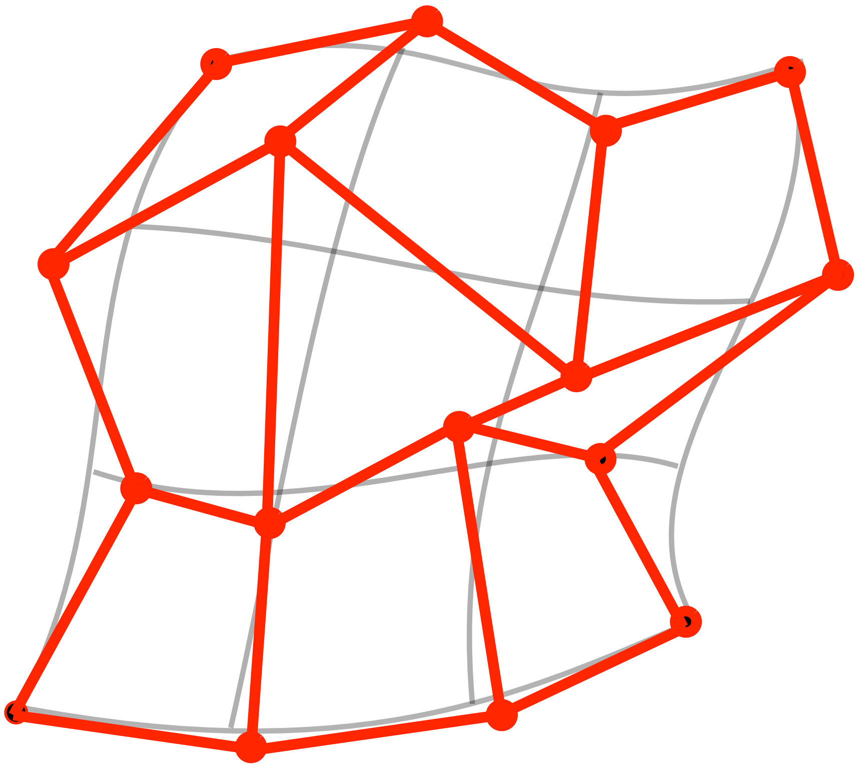
A **bicubic Bézier surface**



Tensor Product Bézier Patches

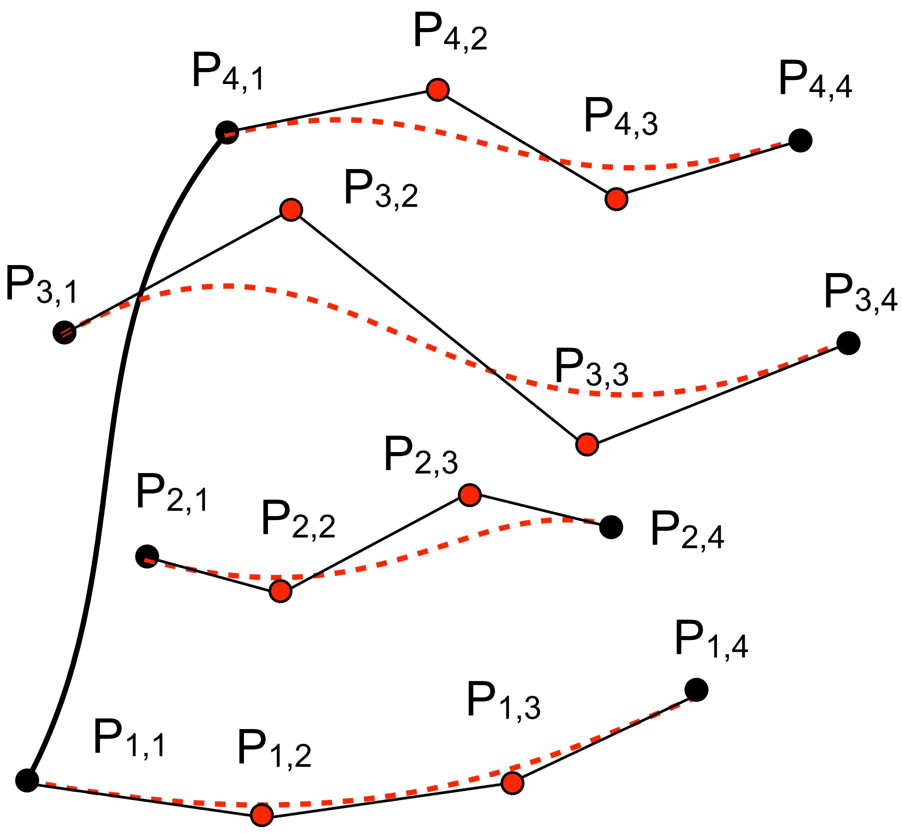
- Note that only the 4 corners are interpolated!

The "Control Mesh"
16 control points



Bicubics, Tensor Product

- $P(u,v) = B_1(u) * P_1(v)$
  $+ B_2(u) * P_2(v)$
  $+ B_3(u) * P_3(v)$
  $+ B_4(u) * P_4(v)$
- $P_i(v) = B_1(v) * P_{i,1}$
  $+ B_2(v) * P_{i,2}$
  $+ B_3(v) * P_{i,3}$
  $+ B_4(v) * P_{i,4}$



Bicubics, Tensor Product

- $P(u,v) = B_1(u) * P_1(v)$
  $+ B_2(u) * P_2(v)$
  $+ B_3(u) * P_3(v)$
  $+ B_4(u) * P_4(v)$
- $P_i(v) = B_1(v) * P_{i,1}$
  $+ B_2(v) * P_{i,2}$
  $+ B_3(v) * P_{i,3}$
  $+ B_4(v) * P_{i,4}$

$$P(u,v) = \sum_{i=1}^{4} B_i(u) \left[\sum_{j=1}^{4} P_{i,j} \, B_j(v) \right]$$

$$= \sum_{i=1}^{4} \sum_{j=1}^{4} P_{i,j} \, B_{i,j}(u,v)$$

with

$$B_{i,j}(u,v) = B_i(u) B_j(v)$$

Bicubics, Tensor Product

- $P(u,v) = B_1(u) * P_1(v) + B_2(u) * P_2(v) + B_3(u) * P_3(v) + B_4(u) * P_4(v)$
- $P_i(v) = B_1(v) * P_{i,1} + B_2(v) * P_{i,2} + B_3(v) * P_{i,3} + B_4(v) * P_{i,4}$

$$P(u,v) = \sum_{i=1}^{4} B_i(u) \left[\sum_{j=1}^{4} P_{i,j} B_j(v) \right]$$

*i*th control point for the u curve

$$= \sum_{i=1}^{4} \sum_{j=1}^{4} P_{i,j} B_{i,j}(u,v)$$

with

$$B_{i,j}(u,v) = B_i(u) B_j(v)$$

Tensor product of Bi and Bj

Bicubics, Tensor Product

- $P(u,v) = B_1(u) * P_1(v) + B_2(u) * P_2(v) + B_3(u) * P_3(v) + B_4(u) * P_4(v)$
- $P_i(v) = B_1(v) * P_{i,1} + B_2(v) * P_{i,2} + B_3(v) * P_{i,3} + B_4(v) * P_{i,4}$

$$P(u,v) =$$

16 control points $P_{i,j}$

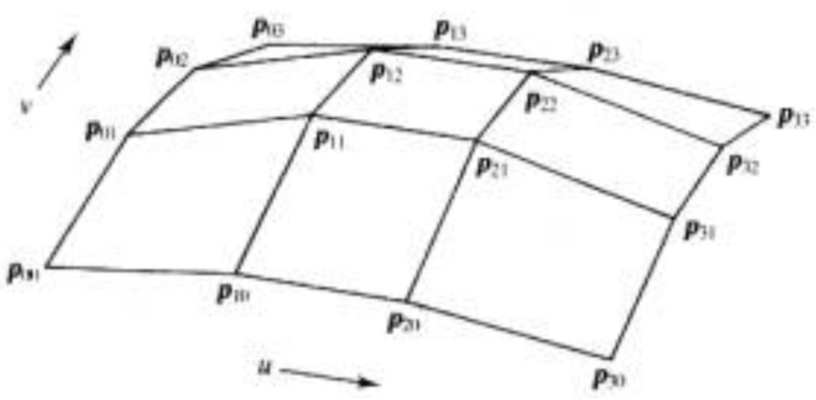
16 2D basis functions $B_{i,j}$

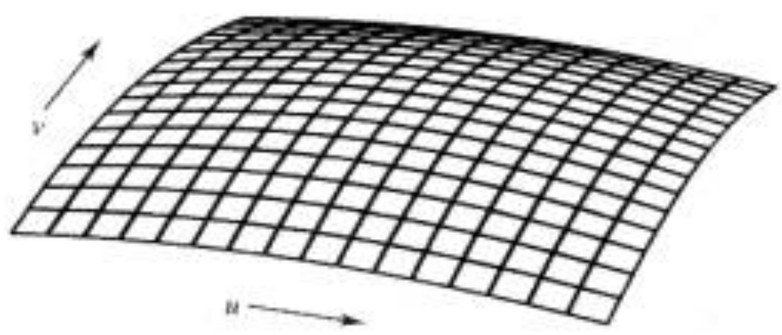
$$= \sum_{i=1}^{4} \sum_{j=1}^{4} P_{i,j} B_{i,j}(u,v)$$

$$B_{i,j}(u,v) = B_i(u) B_j(v)$$

Recap: Tensor Bézier Patches

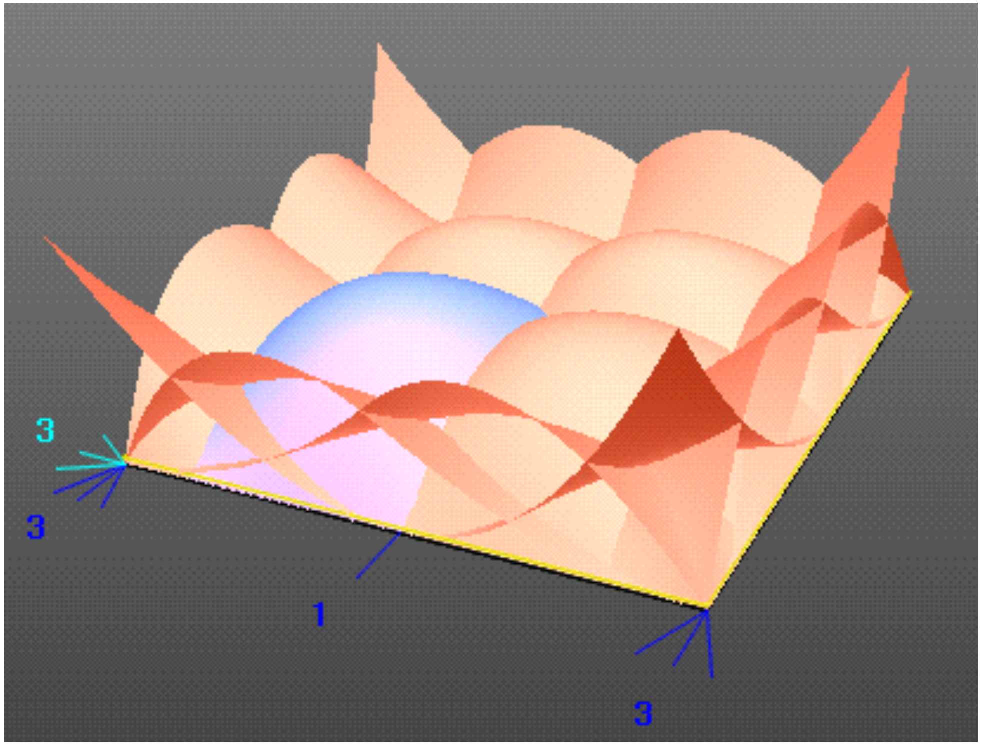
- Parametric surface P(*u*,*v*) is a cubic polynomial of two variables *u* & *v*
- Defined by 4x4=16 control points $P_{1,1}$, $P_{1,2}$.... $P_{4,4}$
- Interpolates 4 corners, approximates others





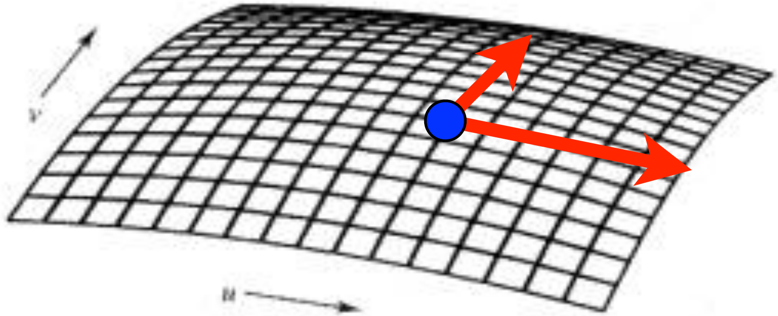
Tensor Product Bézier Patches

- Defined by 4x4=16 control points $P_{1,1}$, $P_{1,2}$.... $P_{4,4}$
- Basis functions = products of two 1D Bernsteins: $B_1(u)B_1(v)$; $B_1(u)B_2(v)$;... $B_4(u)B_4(v)$



Tangents and Normals for Patches

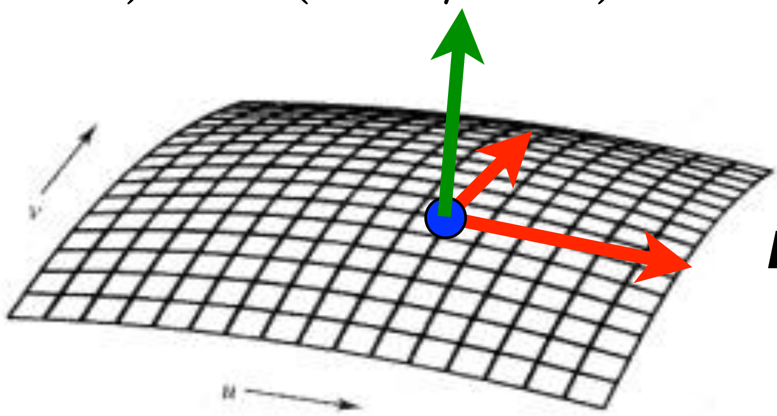
- P(u,v) is a 3D point specified by u, v
- The partial derivatives $\partial P/\partial u$ and $\partial P/\partial v$ are 3D vectors
  - Both are tangent to surface at P



Tangents and Normals for Patches

- P(u,v) is a 3D point specified by *u*, *v*
- The partial derivatives $\partial P/\partial u$ and $\partial P/\partial v$ are 3D vectors
  - Both are tangent to surface at P
  - Normal is perpendicular to both, i.e.,

$$n = (\partial P/\partial u) \times (\partial P/\partial v)$$

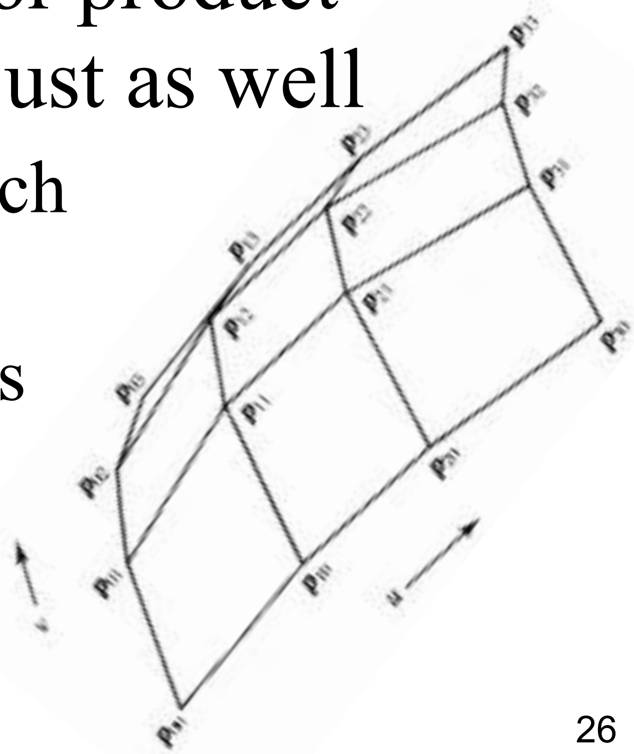


***n* is usually not unit, so must normalize!**

Tensor Product B-Spline Patches

- Bézier and B-Spline curves are both cubics
  - => Can change between representations using matrices

- Consequently, you can build tensor product surface patches out of B-Splines just as well
  - Still 4x4 control points for each patch
  - 2D basis functions are pairwise products of B-Spline basis functions
  - Sliding window of 4x4 points
  - Yes, simple!

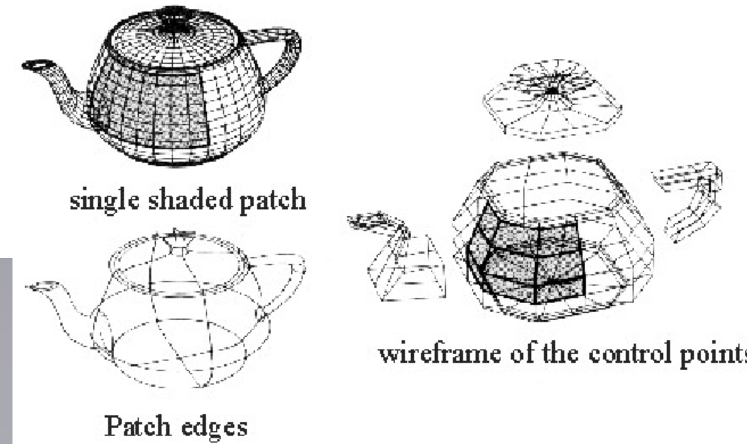


Tensor Product Spline Patches

- Pros
  - Smooth
  - Defined by reasonably small set of points
- Cons
  - Harder to render (usually converted to triangles)
  - **Tricky to ensure continuity at patch boundaries**
- Extensions
  - Rational splines: Splines in homogeneous coordinates
  - NURBS: Non-Uniform Rational B-Splines
    - Like curves: ratio of polynomials, non-uniform location of control points, etc.

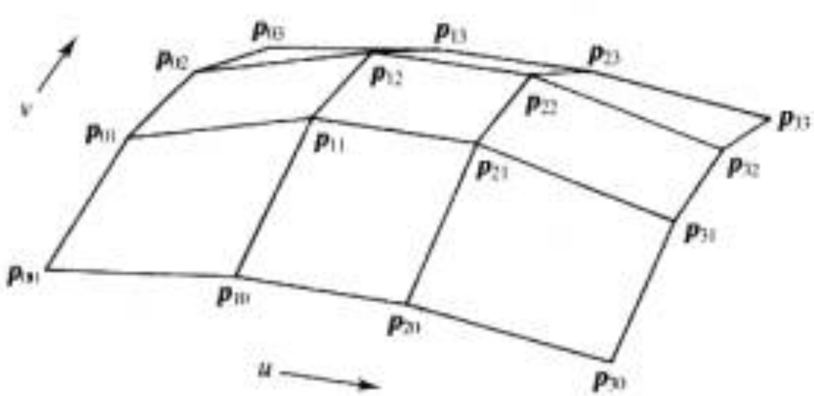
Utah Teapot: Tensor Bézier Splines

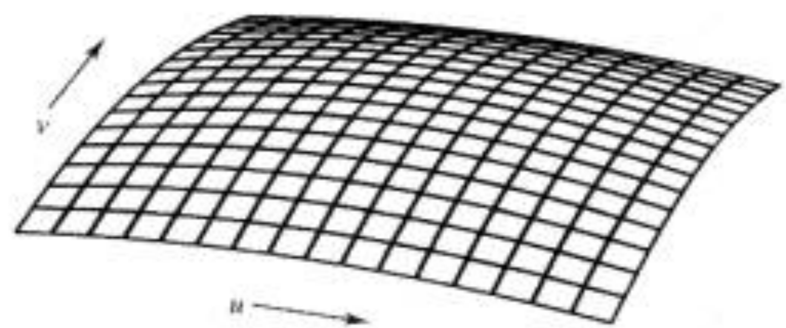
- Designed by Martin Newell



Cool: Displacement Mapping

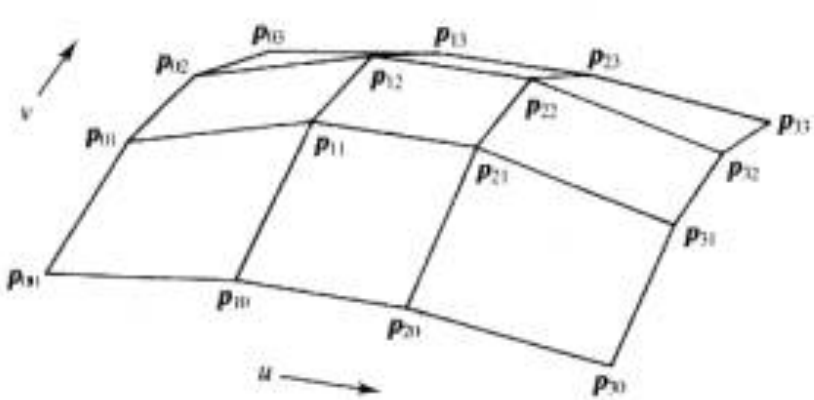
- Not all surfaces are smooth...

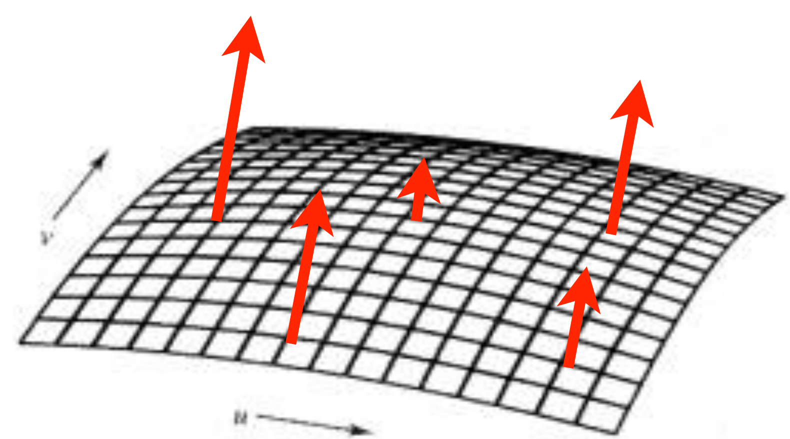




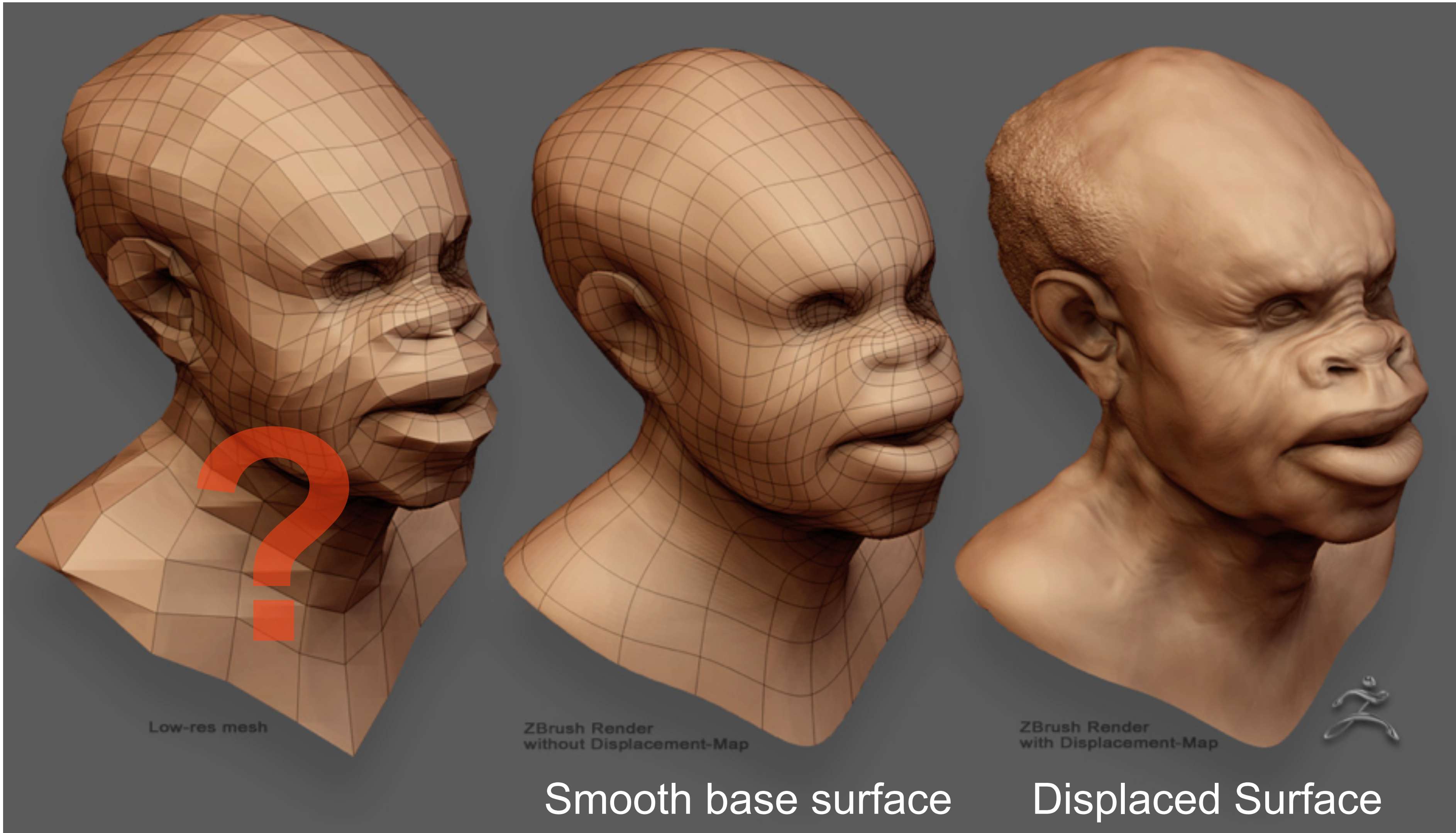
Cool: Displacement Mapping

- Not all surfaces are smooth...
- "Paint" displacements on a smooth surface
  - For example, in the direction of normal
- Tessellate smooth patch into fine grid, then add displacement D(u,v) to vertices
- Heavily used in movies, more and more in games





Displacement Mapping Example



zbrushcentral

Displacement Mapping Video

- Here, input triangles are also dynamically tessellated by the GPU

- http://www.youtube.com/watch?v=XDlC6tAqc20

Extra: Tensor Notation

- See the slides!

Recap: Matrix Notation for Curves

- Cubic Bézier in matrix notation

point on curve
(2x1 vector)

Canonical
"power basis"

$$P(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} =$$

$$\begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \end{pmatrix} \begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}$$

"Geometry matrix"
of control points $P_1..P_4$
(2 x 4)

"Spline matrix"
(Bernstein)

Hardcore: Matrix Notation for Patches

- Not required, but convenient!

$$P(u,v) = \sum_{i=1}^{4} B_i(u) \left[\sum_{j=1}^{4} P_{i,j}\, B_j(v) \right]$$

x coordinate of surface at (*u*,*v*)

Column vector of basis functions (*v*)

$$P^x(u,v) = (B_1(u), \ldots, B_4(u)) \begin{pmatrix} P^x_{1,1} & \cdots & P^x_{1,4} \\ \vdots & & \vdots \\ P^x_{4,1} & \cdots & P^x_{4,4} \end{pmatrix} \begin{pmatrix} B_1(v) \\ \vdots \\ B_4(v) \end{pmatrix}$$

Row vector of basis functions (*u*)

4x4 matrix of *x* coordinates of the control points

Hardcore: Matrix Notation for Patches

- Curves:

$$P(t) = \boldsymbol{G}\,\boldsymbol{B}\,\boldsymbol{T}(t)$$

- Surfaces:

$$P^x(u,v) = \boldsymbol{T}(u)^{\mathrm{T}}\,\boldsymbol{B}^{\mathrm{T}}\,\boldsymbol{G}^x\,\boldsymbol{B}\,\boldsymbol{T}(v)$$

A separate 4x4 geometry matrix for x, y, z

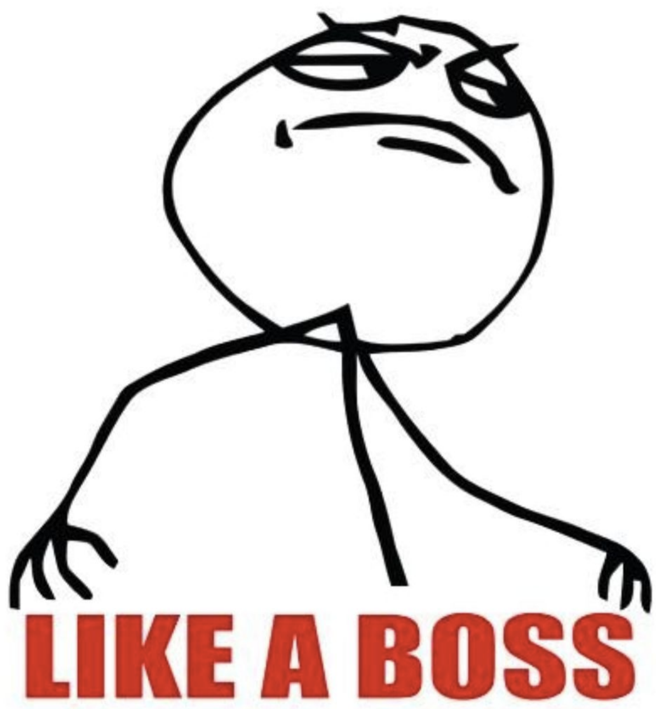
- **T** = power basis
  B = spline matrix
  G = geometry matrix

Super Hardcore: Tensor Notation

- You can stack the $\boldsymbol{G}^x$, $\boldsymbol{G}^y$, $\boldsymbol{G}^z$ matrices into a geometry **tensor** of control points
  - I.e., $G^k_{i,j}$ = the *k*:th coordinate of control point $P_{i,j}$
  - A cube of numbers!

$$P^k(u,v) = \boldsymbol{T}^l(u)\,\boldsymbol{B}_l^i\,\boldsymbol{G}_{ij}^k\,\boldsymbol{B}_m^j\,\boldsymbol{T}^m(v)$$

- "Einstein summation convention": Repeated indices are summed over (here *l, m, i, j*)
- Definitely not required, but nice! :)
  - See http://en.wikipedia.org/wiki/Multilinear_algebra


LIKE A BOSS