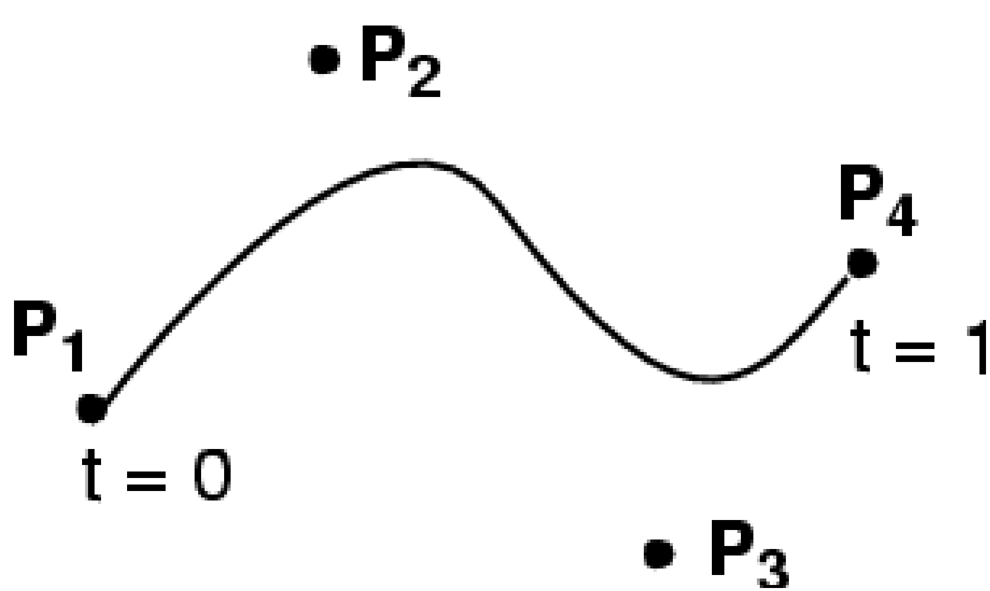
6.837 Computer Graphics

# Curve Properties & Conversion, Surface Representations

vectorportal.com

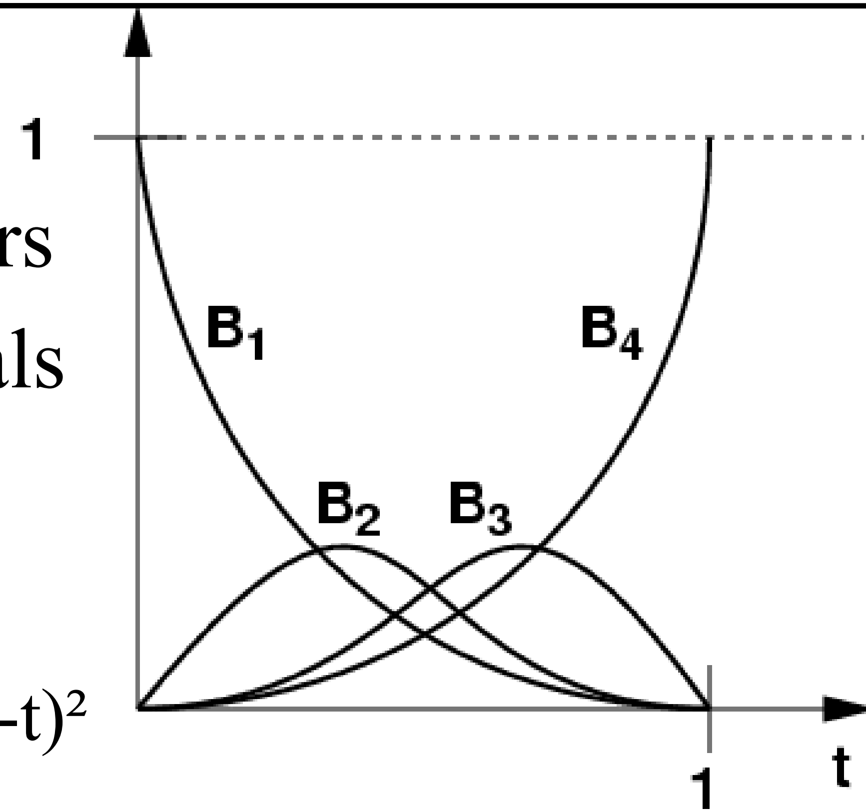
# Cubic Bezier Splines

- $$\begin{aligned} P(t) = & (1-t)^3 && P1 \\ + & 3t(1-t)^2 && P2 \\ + & 3t^2(1-t) && P3 \\ + & t^3 && P4 \end{aligned}$$



# Bernstein Polynomials

- For Bézier curves, the basis polynomials/vectors are Bernstein polynomials




- For cubic Bezier curve:

  $B_1(t)=(1-t)^3$ $B_2(t)=3t(1-t)^2$

  $B_3(t)=3t^2(1-t)$ $B_4(t)=t^3$

  (careful with indices, many authors start at 0)

- Defined for any degree

# General Spline Formulation

$$Q(t) = \mathbf{GBT(t)} = \text{Geometry } \mathbf{G} \cdot \text{Spline Basis } \mathbf{B} \cdot \text{Power Basis } \mathbf{T(t)}$$

- Geometry: control points coordinates assembled into a matrix (P1, P2, …, Pn+1)
- Power basis: the monomials 1, *t*, *t*2, ...
- Cubic Bézier:

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

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

# Questions?

# Linear Transformations & Cubics

- What if we want to transform each point on the curve with a linear transformation **M**?

$$P'(t) = \mathbf{M}\left[\begin{pmatrix} P_{1,x} & P_{2,x} & P_{3,x} & P_{4,x} \\ P_{1,y} & P_{2,y} & P_{3,y} & P_{4,y} \end{pmatrix} \begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}\right]$$

# Linear Transformations & Cubics

- What if we want to transform each point on the curve with a linear transformation **M**?
  - Because everything is linear, it is the same as transforming only the control points

$$P'(t) = \mathbf{M}\left[\begin{pmatrix} P_{1,x} & P_{2,x} & P_{3,x} & P_{4,x} \\ P_{1,y} & P_{2,y} & P_{3,y} & P_{4,y} \end{pmatrix} \begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}\right]$$

$$= \left[\mathbf{M} \begin{pmatrix} P_{1,x} & P_{2,x} & P_{3,x} & P_{4,x} \\ P_{1,y} & P_{2,y} & P_{3,y} & P_{4,y} \end{pmatrix}\right] \begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}$$

# Affine Transformations

- Homogeneous coordinates also work
  - Means you can translate, rotate, shear, etc.
  - Note though that you need to normalize P’ by $1/w'$

$$P'(t) = \mathbf{M}\left[\begin{pmatrix} P_{1,x} & P_{2,x} & P_{3,x} & P_{4,x} \\ P_{1,y} & P_{2,y} & P_{3,y} & P_{4,y} \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}\right]$$

$$= \left[\mathbf{M} \begin{pmatrix} P_{1,x} & P_{2,x} & P_{3,x} & P_{4,x} \\ P_{1,y} & P_{2,y} & P_{3,y} & P_{4,y} \\ 1 & 1 & 1 & 1 \end{pmatrix}\right] \begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}$$

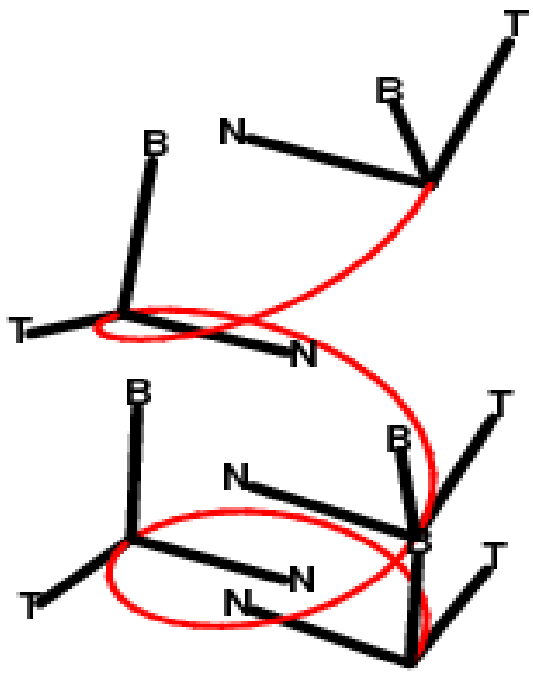
# Questions?

# The Plan for Today

- Differential Properties of Curves & Continuity
- B-Splines
- Surfaces
  - Tensor Product Splines
  - Subdivision Surfaces
  - Procedural Surfaces
  - Other

# Differential Properties of Curves

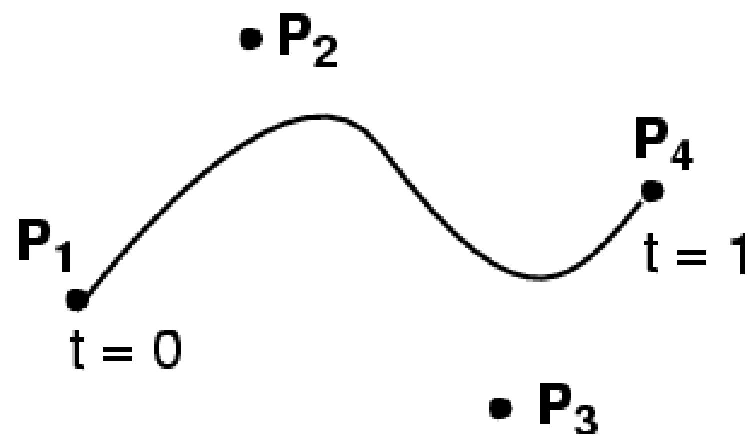
- Motivation
  - Compute normal for surfaces
  - Compute velocity for animation
  - Analyze smoothness



# Velocity

- First derivative w.r.t. *t*
- Can you compute this for Bezier curves?

$$
\begin{aligned}
P(t) = \quad & (1-t)^3 \quad P1 \\
& + 3t(1-t)^2 \quad P2 \\
& + 3t^2(1-t) \quad P3 \\
& + t^3 \quad P4
\end{aligned}
$$

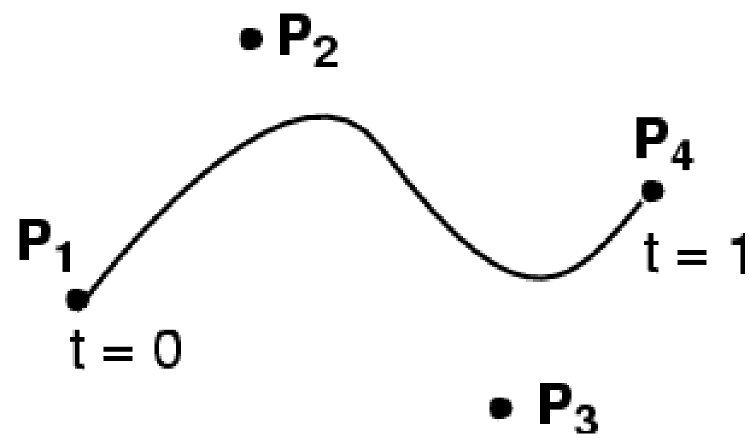



- You know how to differentiate polynomials...

# Velocity

- First derivative w.r.t. *t*
- Can you compute this for Bezier curves?

$$
\begin{aligned}
P(t) = \quad & (1-t)^3 \quad P1 \\
& + 3t(1-t)^2 \quad P2 \\
& + 3t^2(1-t) \quad P3 \\
& + t^3 \quad P4
\end{aligned}
$$




Sanity check: t=0; t=1

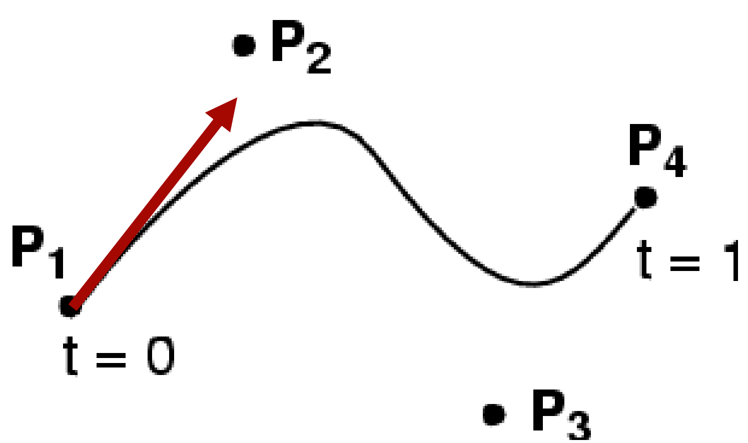
- $$
\begin{aligned}
\mathbf{P'(t) =} \quad & \mathbf{-3(1-t)2} \quad \mathbf{P1} \\
& \mathbf{+ \ [3(1-t)\ 2\ -6t(1-t)]} \quad \mathbf{P2} \\
& \mathbf{+ \ [6t(1-t)-3t\ 2]} \quad \mathbf{P3} \\
& \mathbf{+ \ 3t\ 2} \quad \mathbf{P4}
\end{aligned}
$$

# Linearity?

- Differentiation is a linear operation
  - (f+g)’=f’+g’
  - (af)’=a f’
- This means that the derivative of the basis is enough to know the derivative of any spline.
- Can be done with matrices
  - Trivial in monomial basis
  - But get lower-order polynomials

# Tangent Vector

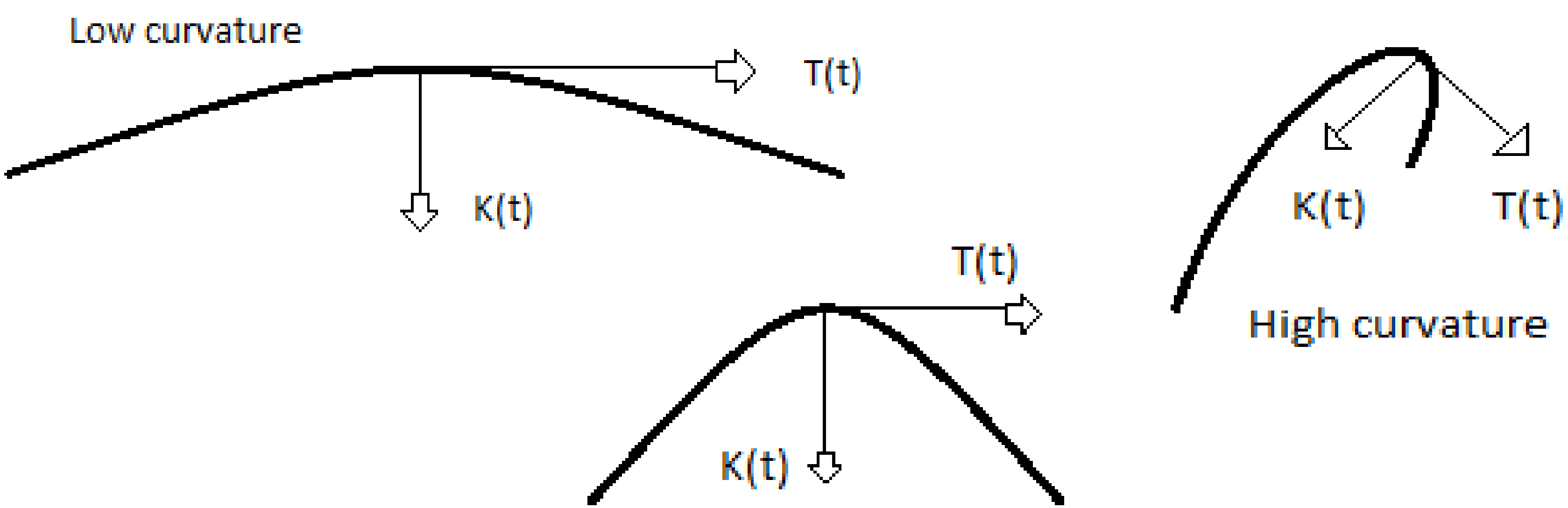
- The tangent to the curve $P(t)$ can be defined as $T(t)=P'(t)/||P'(t)||$
  - normalized velocity, $||T(t)|| = 1$
- This provides us with one orientation for swept surfaces later




Courtesy of Seth Teller.

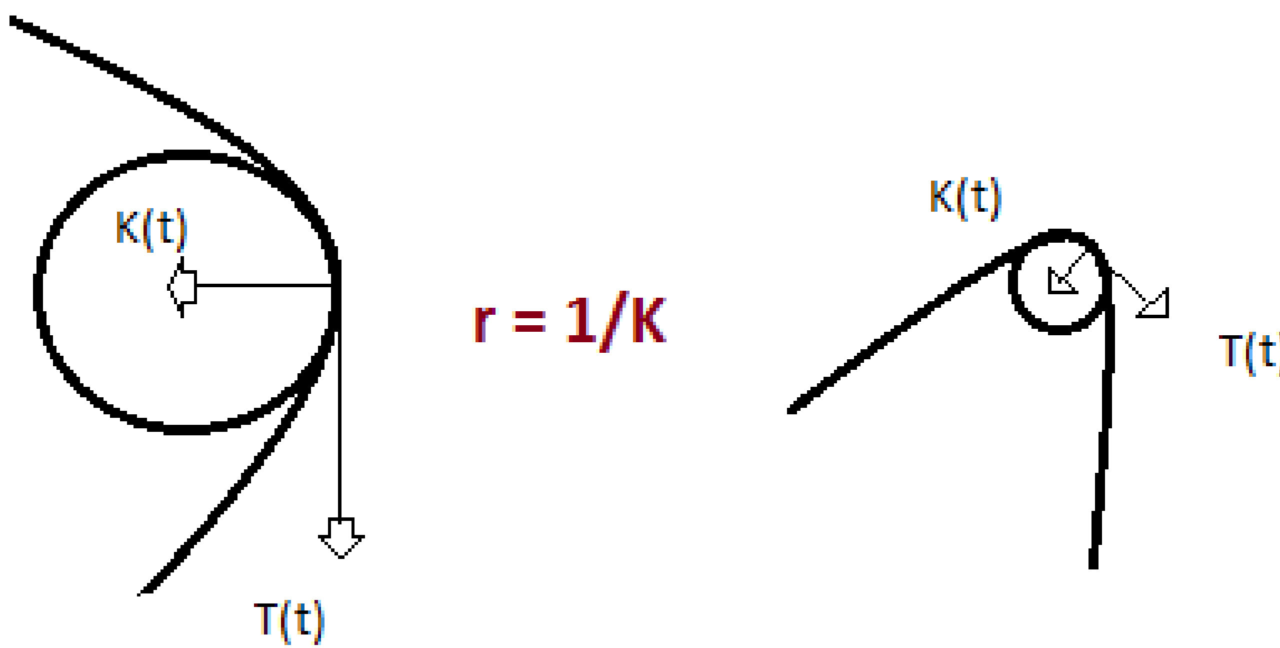
# Curvature Vector

- Derivative of unit tangent
  - $K(t)=T'(t)$
  - Magnitude $||K(t)||$ is constant for a circle
  - Zero for a straight line
- Always orthogonal to tangent, ie. $K \cdot T = 0$



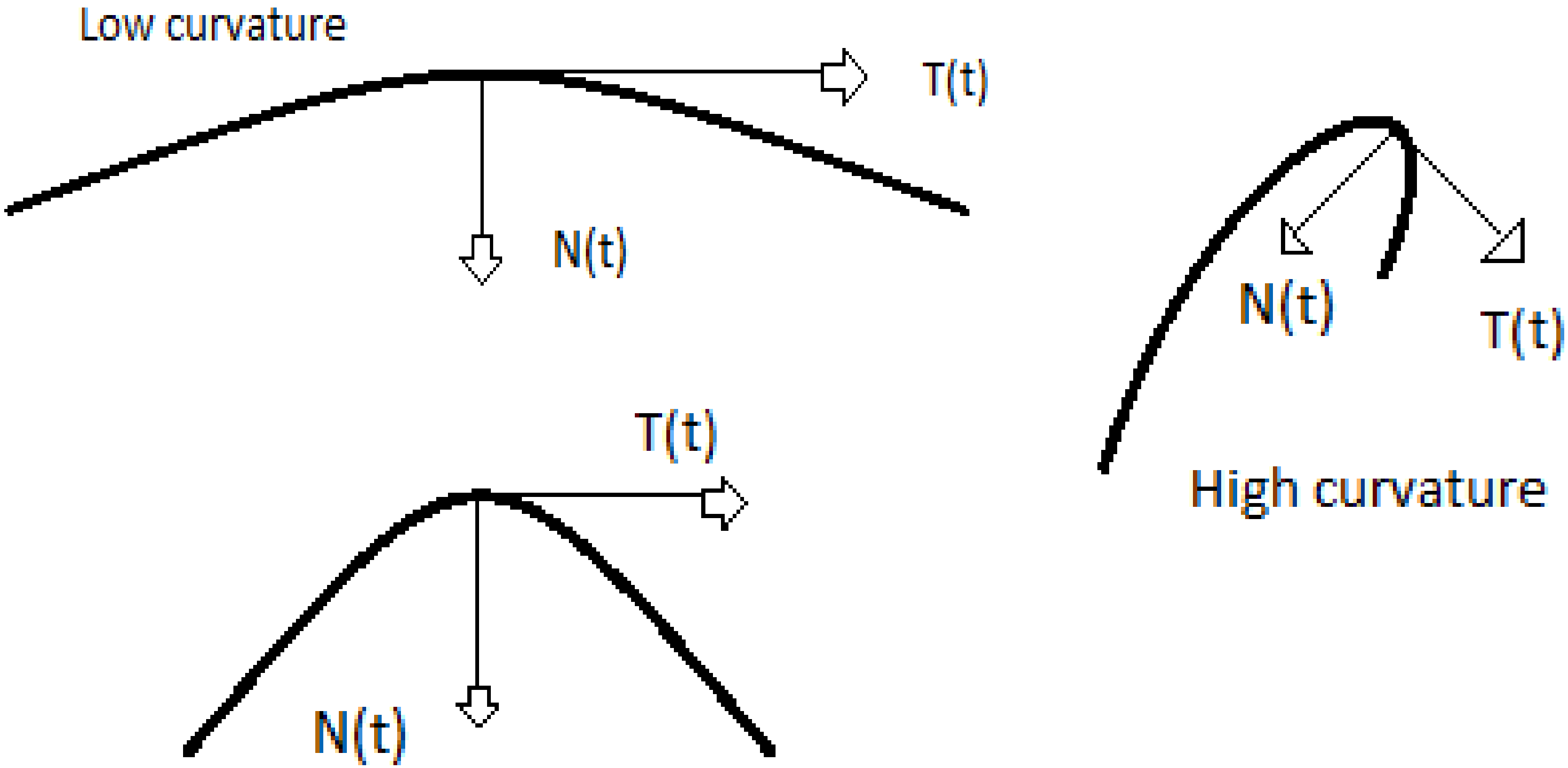
# Geometric Interpretation

- K is zero for a line, constant for circle
  - What constant? 1/r
- 1/||K(t)|| is the radius of the circle that touches P(t) at *t* and has the same curvature as the curve



# Curve Normal

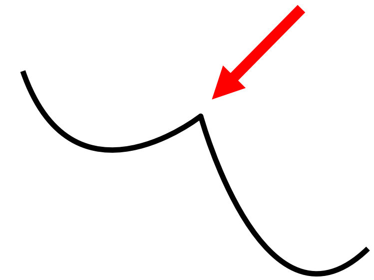
- Normalized curvature: T’(t)/||T’(t)||



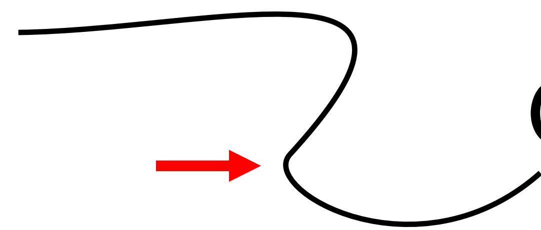
# Questions?

# Orders of Continuity

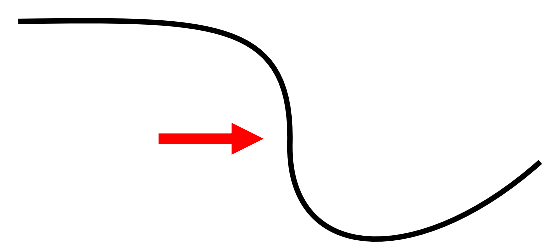
- C0 = continuous
  - The seam can be a sharp kink
- G1 = geometric continuity
  - Tangents **point to the same direction** at the seam
- C1 = parametric continuity
  - Tangents **are the same** at the seam, implies G1
- C2 = curvature continuity
  - Tangents and their derivatives are the same



C0



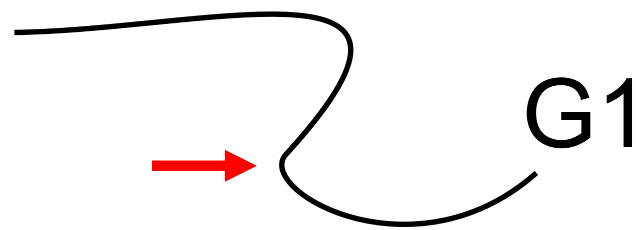
G1

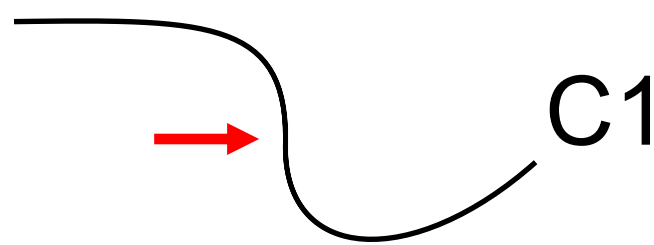


C1

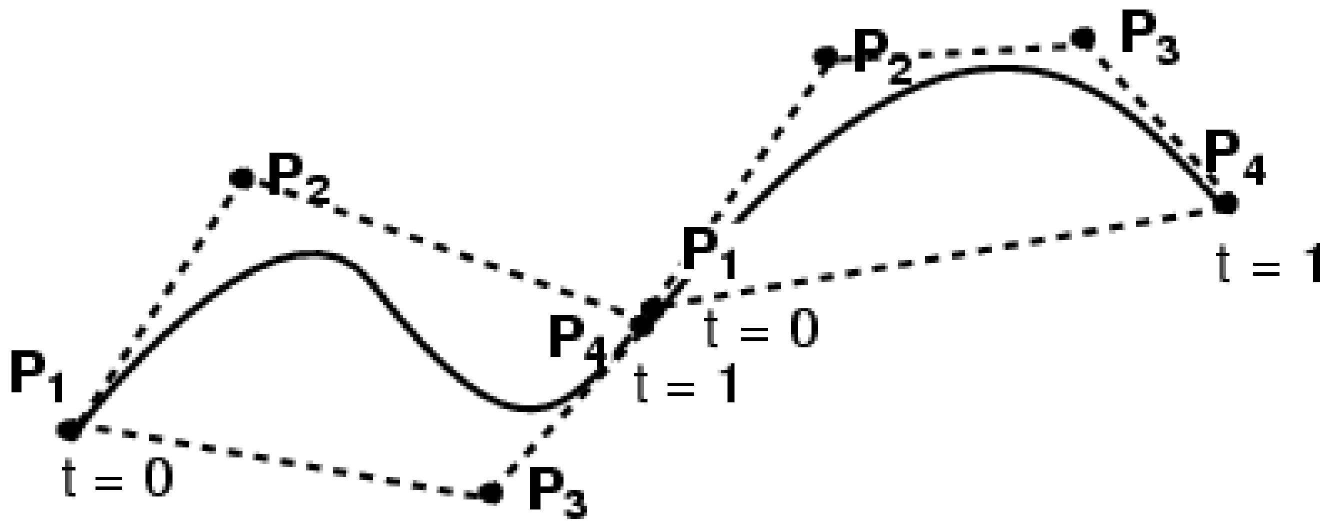
# Orders of Continuity

- G1 = geometric continuity
  - Tangents **point to the same direction** at the seam
  - good enough for modeling
- C1 = parametric continuity
  - Tangents **are the same** at the seam, implies G1
  - often necessary for animation



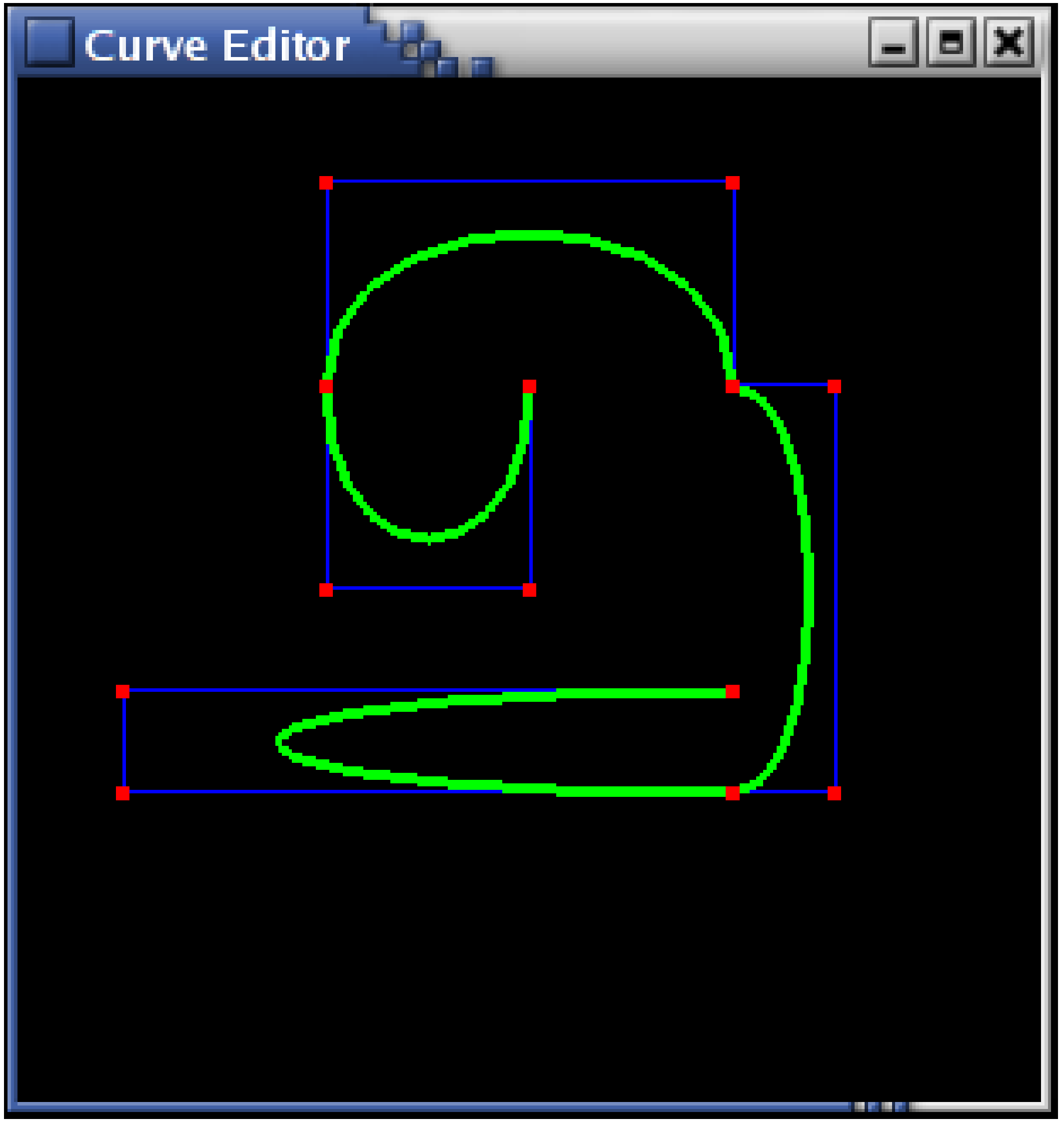



# Connecting Cubic Bézier Curves




- How can we guarantee C0 continuity?
- How can we guarantee G1 continuity?
- How can we guarantee C1 continuity?
- C2 and above gets difficult

# Connecting Cubic Bézier Curves

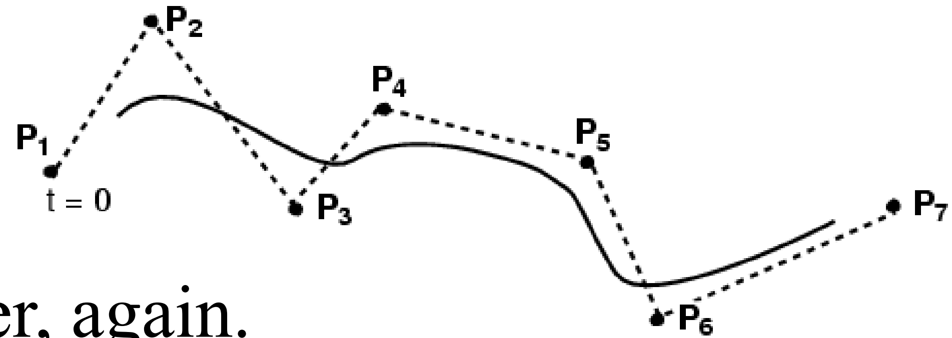



- Where is this curve
  - C0 continuous?
  - G1 continuous?
  - C1 continuous?
- What's the relationship between:
  - the # of control points, and the # of cubic Bézier subcurves?

# Questions?

# Cubic B-Splines

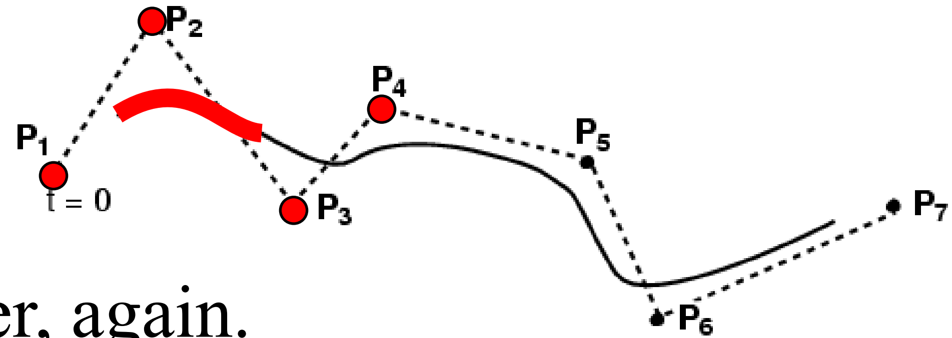
- $\geq$ 4 control points
- Locally cubic
  - Cubics chained together, again.




Courtesy of Seth Teller.

# Cubic B-Splines

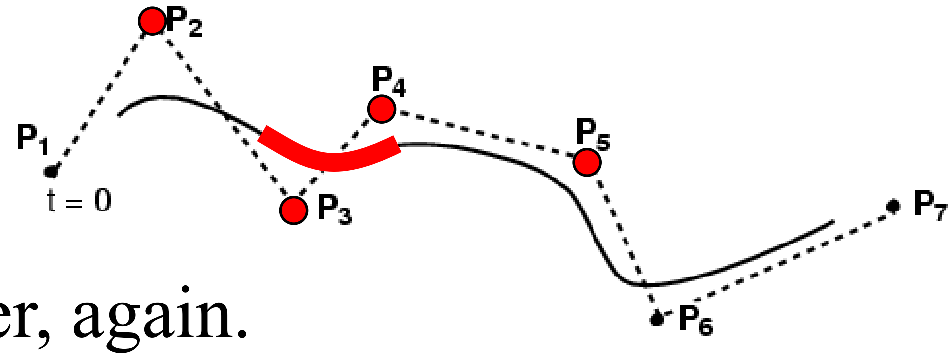
- $\geq 4$ control points
- Locally cubic
  - Cubics chained together, again.




Courtesy of Seth Teller.

# Cubic B-Splines

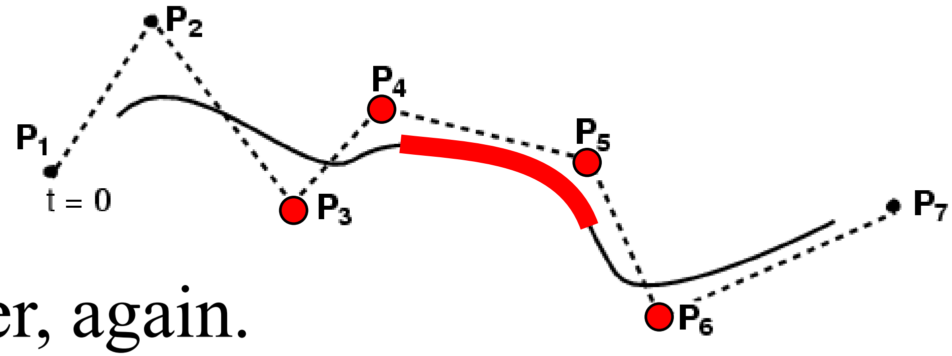
- $\geq$ 4 control points
- Locally cubic
  - – Cubics chained together, again.




Courtesy of Seth Teller.

# Cubic B-Splines

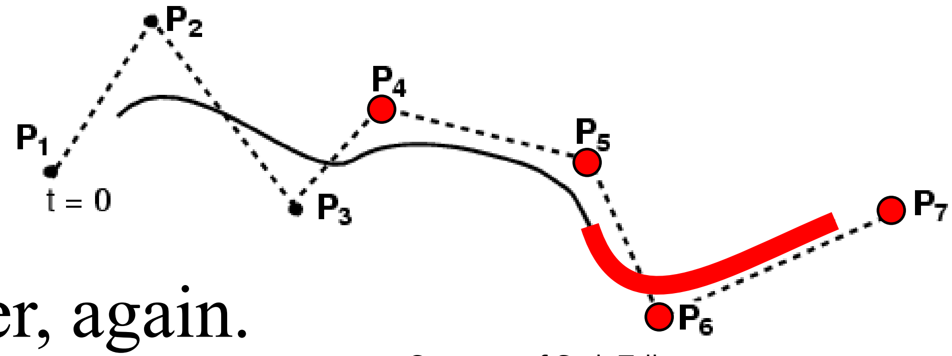
- $\geq 4$ control points
- Locally cubic
  - Cubics chained together, again.




Courtesy of Seth Teller.

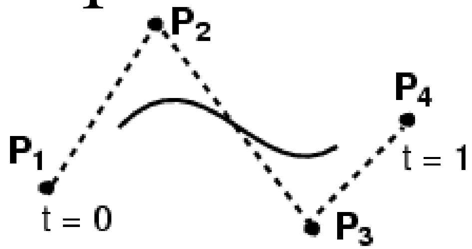
# Cubic B-Splines

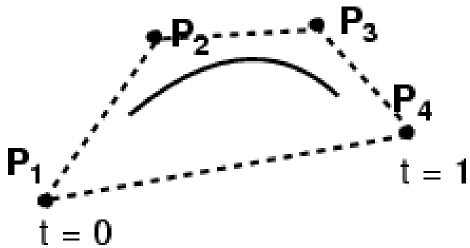
- $\geq$ 4 control points
- Locally cubic
  - Cubics chained together, again.

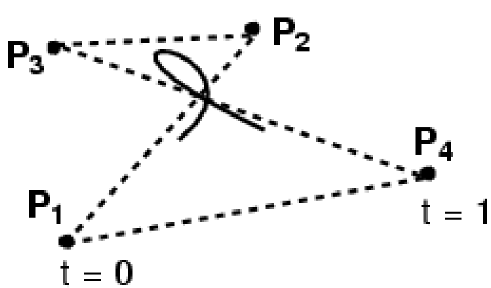



Courtesy of Seth Teller.

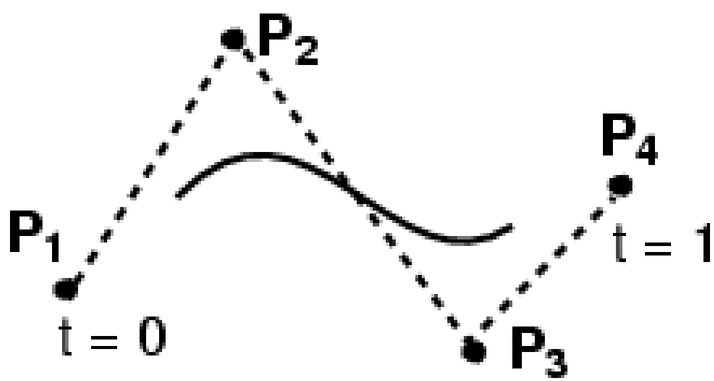
- Curve is not constrained to pass through any control points



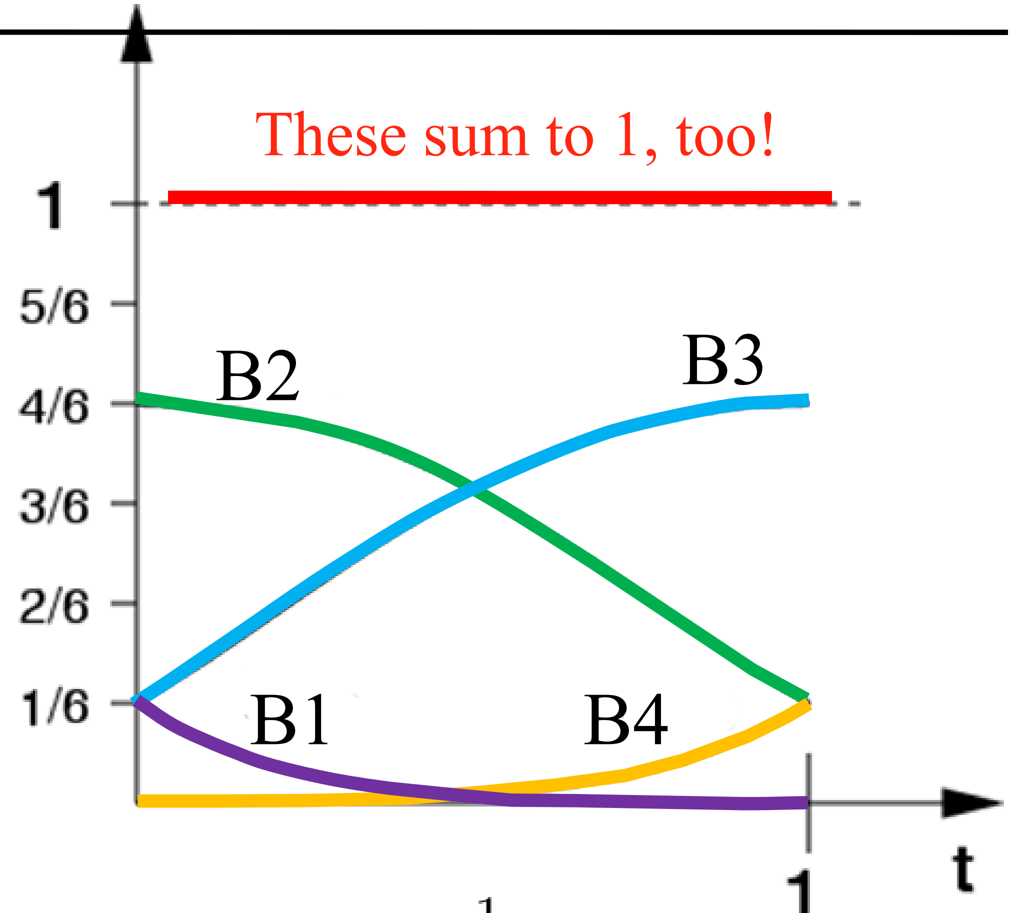



# Cubic B-Splines: Basis




A B-Spline curve is also bounded by the convex hull of its control points.



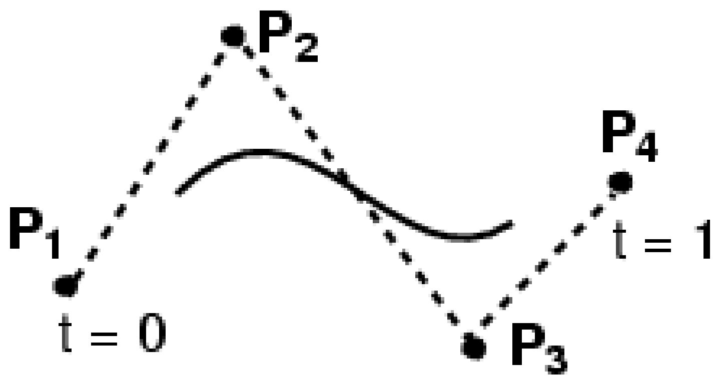

$$B_1(t) = \frac{1}{6}(1-t)^3$$

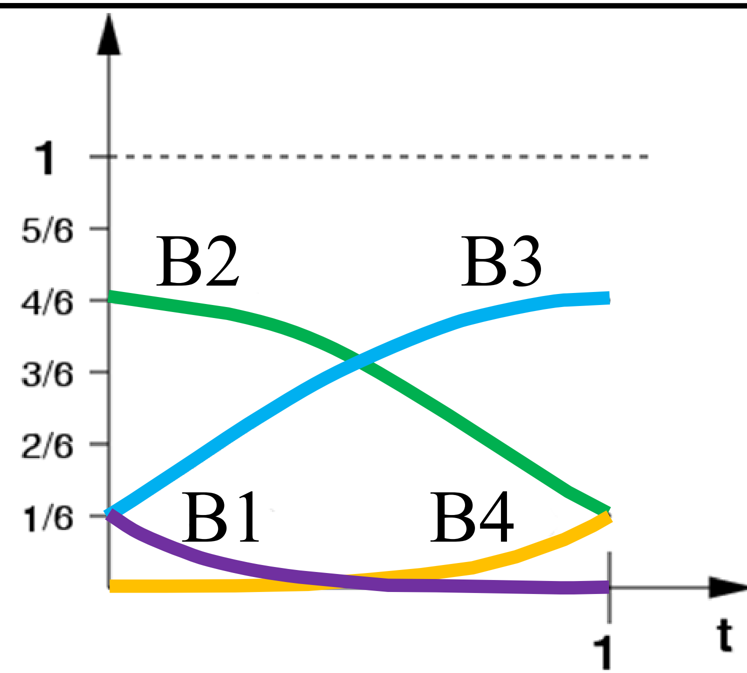
$$B_3(t) = \frac{1}{6}(-3t^3 + 3t^2 + 3t + 1)$$

$$B_2(t) = \frac{1}{6}(3t^3 - 6t^2 + 4)$$

$$B_4(t) = \frac{1}{6}t^3$$

# Cubic B-Splines: Basis



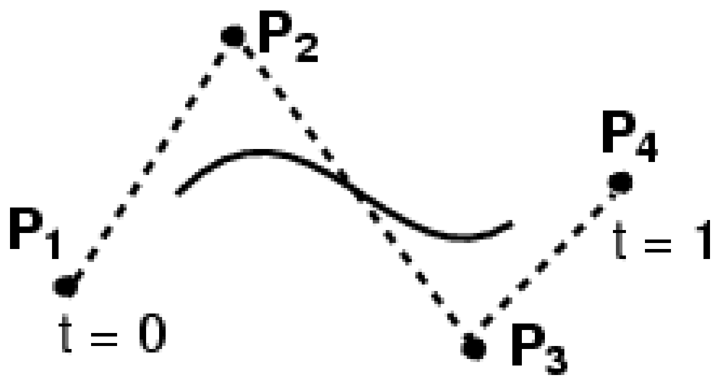



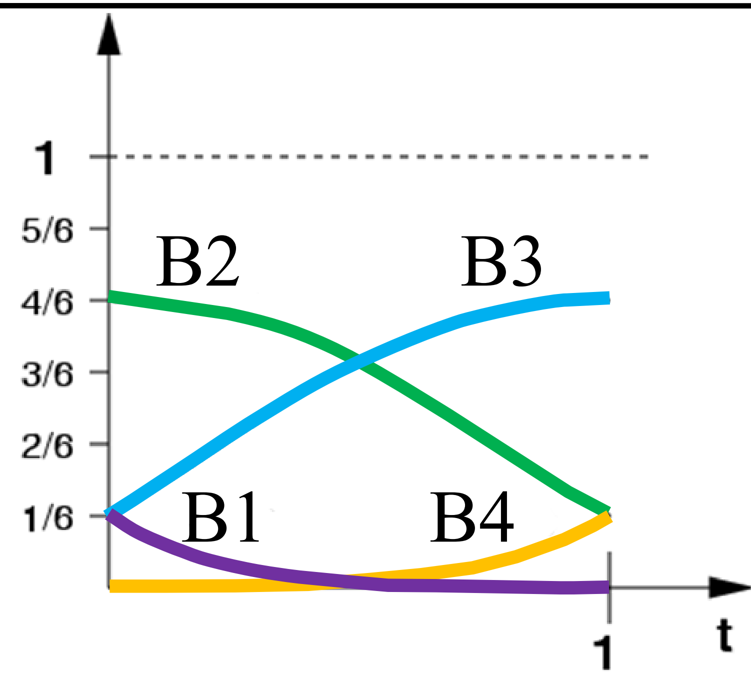

$$Q(t) = \frac{(1-t)^3}{6} P_1 + \frac{3t^3 - 6t^2 + 4}{6} P_2 + \frac{-3t^3 + 3t^2 + 3t + 1}{6} P_3 + \frac{t^3}{6} P_4$$

$$Q(t) = \mathbf{GBT(t)}$$

$$\boxed{B_{B-Spline} = \frac{1}{6} \begin{pmatrix} 1 & -3 & 3 & -1 \\ 4 & 0 & -6 & 3 \\ 1 & 3 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}}$$

# Cubic B-Splines: Basis



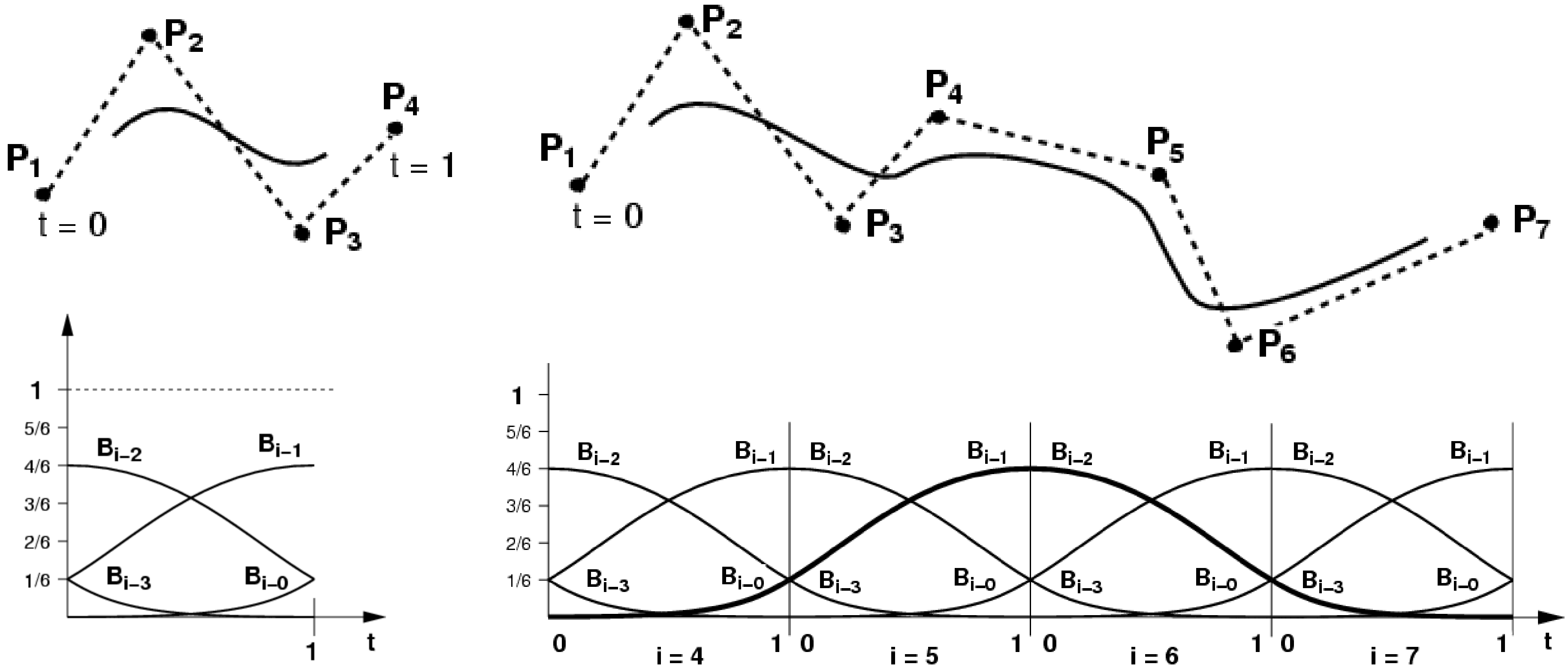

$$Q(t) = \frac{(1-t)^3}{6} P_1 + \frac{3t^3 - 6t^2 + 4}{6} P_2 + \frac{-3t^3 + 3t^2 + 3t + 1}{6} P_3 + \frac{t^3}{6} P_4$$

$$Q(t) = \mathbf{GBT(t)}$$

$$\boxed{B_{B-Spline} = \frac{1}{6} \begin{pmatrix} 1 & -3 & 3 & -1 \\ 4 & 0 & -6 & 3 \\ 1 & 3 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}}$$

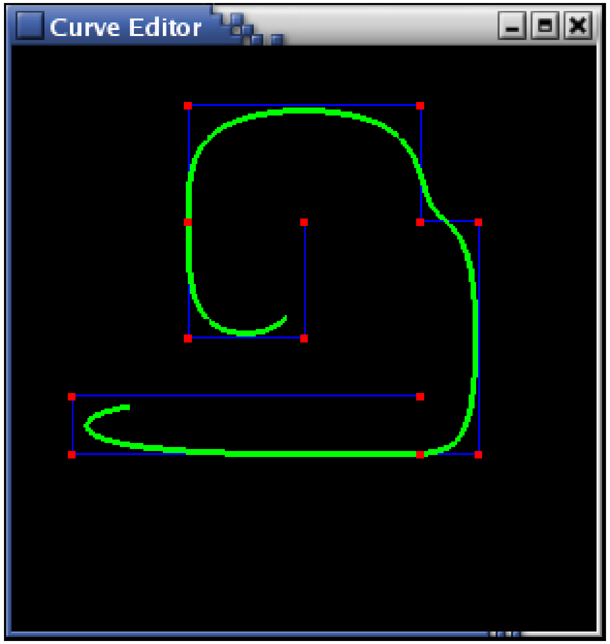
# Cubic B-Splines

- Local control (windowing)
- Automatically C2, and no need to match tangents!

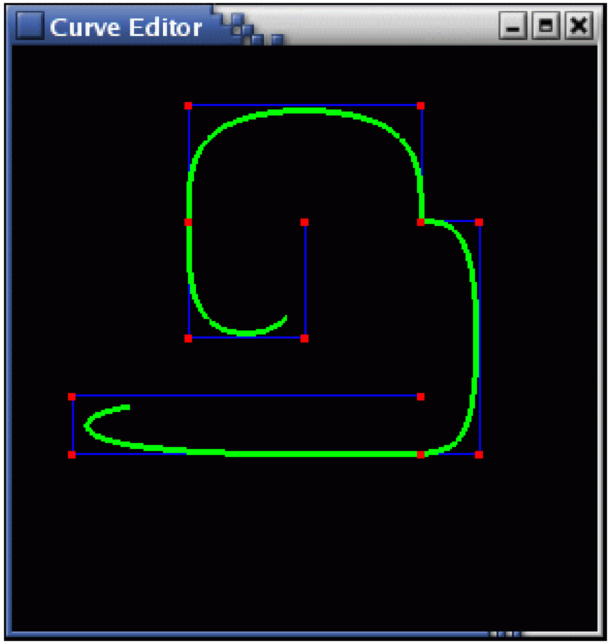



Courtesy of Seth Teller. Used with permission.

# B-Spline Curve Control Points

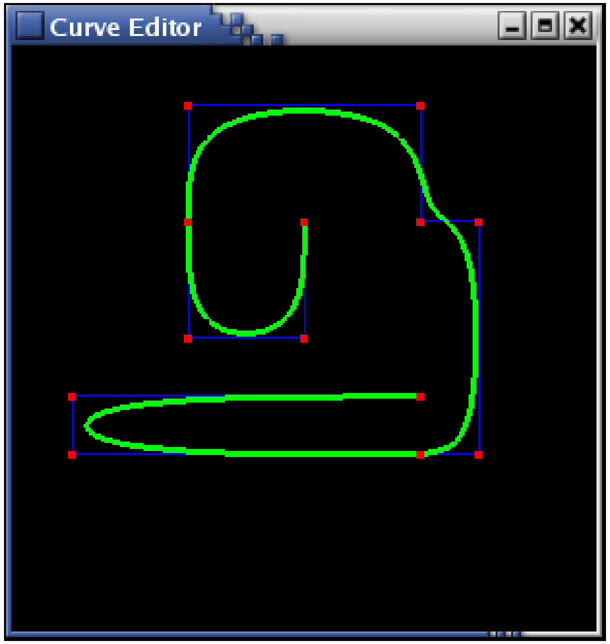



Default B-Spline




B-Spline with derivative discontinuity

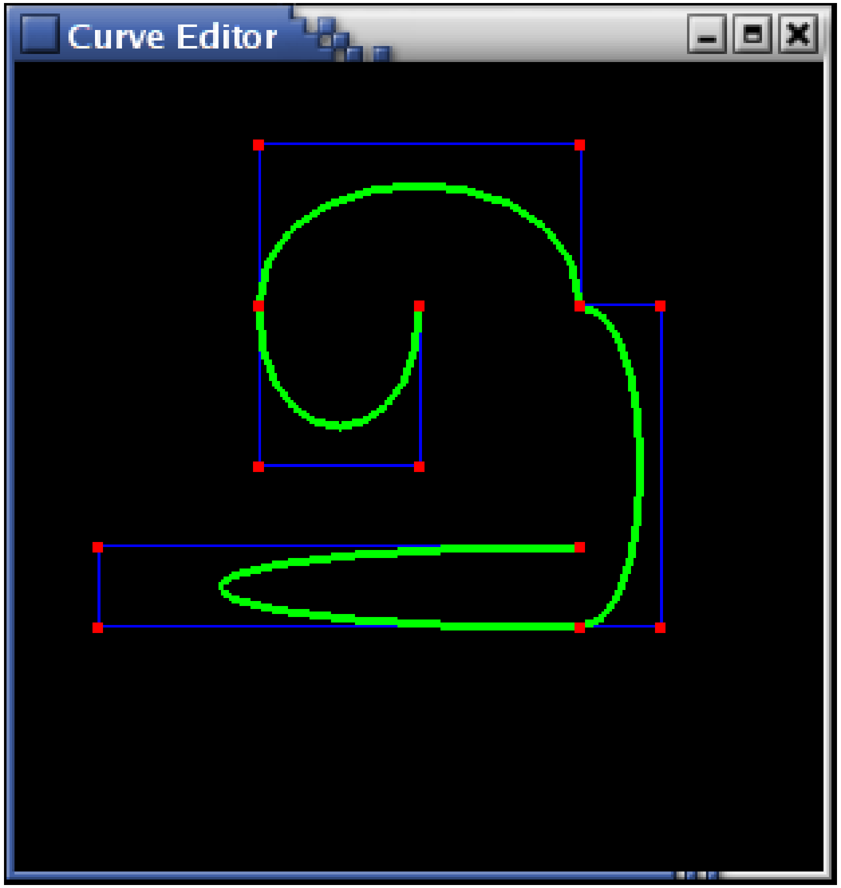
Repeat interior control point



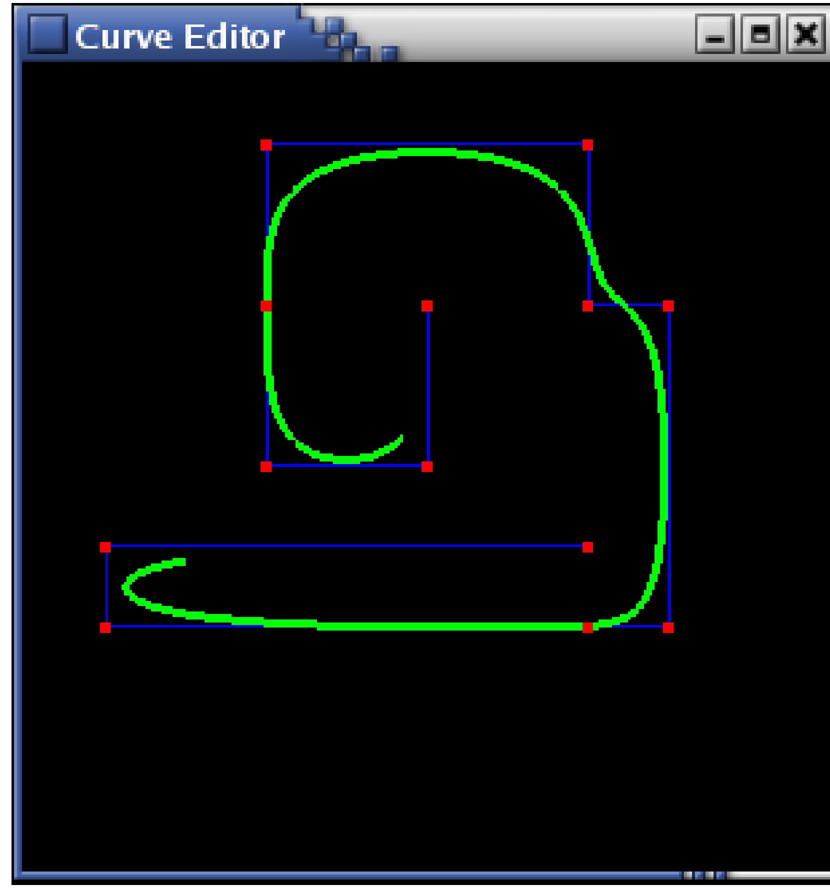

B-Spline which passes through end points

Repeat end points

# Bézier ≠ B-Spline




Bézier




B-Spline

**But both are cubics, so one can be converted into the other!**

# Converting between Bézier & BSpline

$$Q(t) = \mathbf{GBT(t)} = \text{Geometry } \mathbf{G} \cdot \text{Spline Basis } \mathbf{B} \cdot \text{Power Basis } \mathbf{T(t)}$$

- Simple with the basis matrices!
  - Note that this only works for a single segment of 4 control points

$$B_{Bezier} = \begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

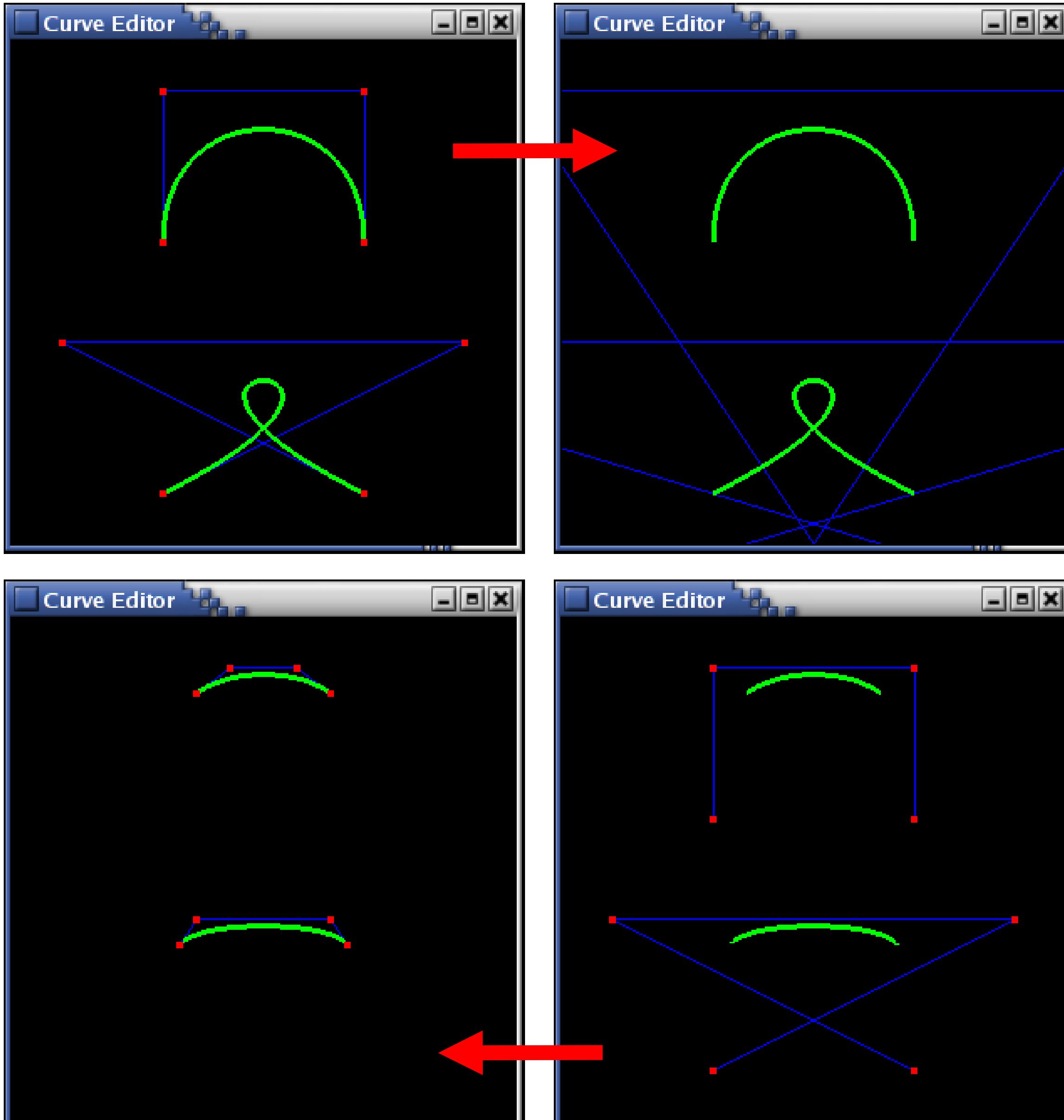
- P(t) = **G B**1 **T**(t) = **G B**1 **(B**2-1**B**2**)** **T**(t)= (**G B**1 **B**2-1) **B**2 **T**(t)

$$B_{B-Spline} = \frac{1}{6} \begin{pmatrix} 1 & -3 & 3 & -1 \\ 4 & 0 & -6 & 3 \\ 1 & 3 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- **G B**1 **B**2-1 are the control points for the segment in new basis.

# Converting between Bézier & B-Spline

original control points as Bézier




new BSpline control points to match Bézier

new Bézier control points to match B-Spline

original control points as B-Spline

# NURBS (Generalized B-Splines)

- Rational cubics
  - Use homogeneous coordinates, just add $w$ !
    - Provides an extra weight parameter to control points

- NURBS: Non-Uniform Rational B-Spline
  - **non-uniform** = different spacing between the blending functions, a.k.a. "knots"
  - **rational** = ratio of cubic polynomials (instead of just cubic)
    - implemented by adding the homogeneous coordinate $w$ into the control points.

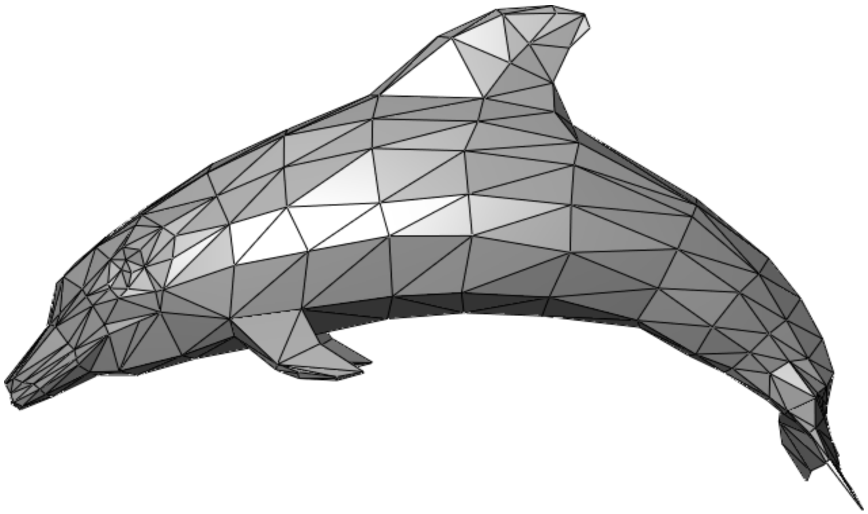
# Questions?

# Representing Surfaces

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

# Triangle Meshes

- What you've used so far in Assignment 0
- Triangle represented by 3 vertices
- **Pro**: simple, can be rendered directly
- **Cons**: not smooth, needs many triangles to approximate smooth surfaces (tessellation)

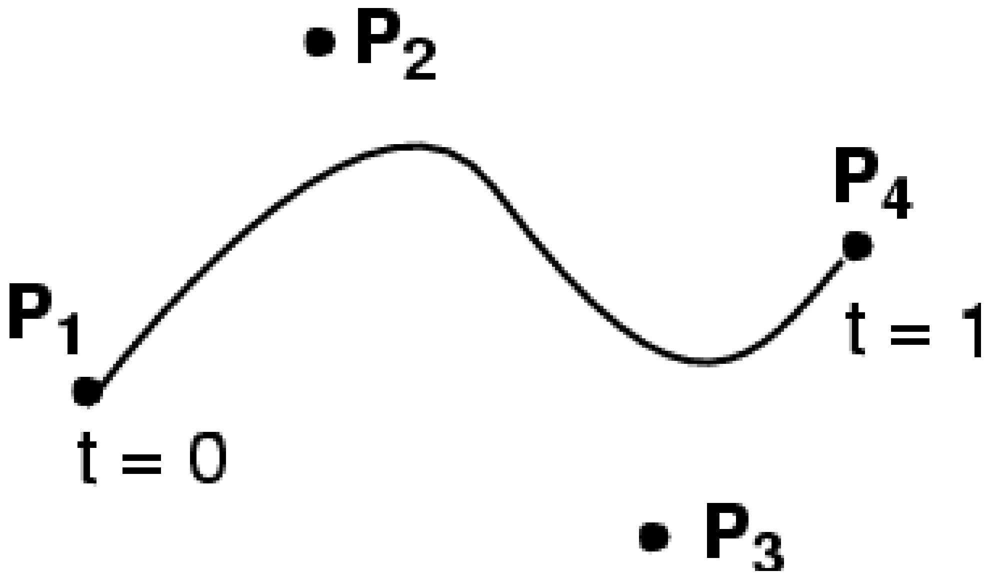


# Smooth Surfaces?

- $P(t) = (1-t)^3 \quad P1$
  $+ \quad 3t(1-t)^2 \quad P2$
  $+ \quad 3t^2(1-t) \quad P3$
  $+ \quad t^3 \quad P4$

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

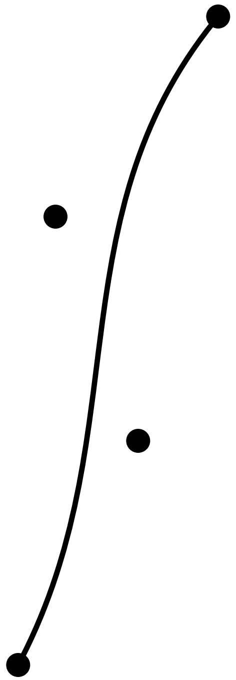
What about a surface?



# How to Build Them? Here's an Idea

- $P(u) = (1\text{-}u)^3 \quad P1$
  $+ \quad 3u(1\text{-}u)^2 \; P2$
  $+ \quad 3u^2(1\text{-}u) \; P3$
  $+ \quad u^3 \quad P4$

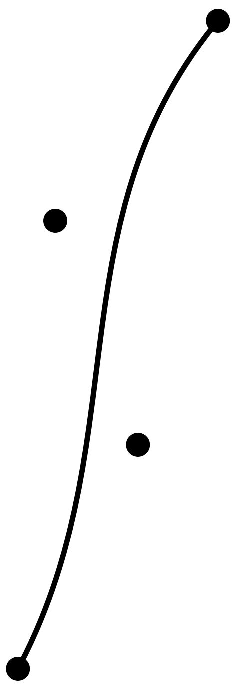
(Note! We relabeled *t* to *u*)



# How to Build Them? Here's an Idea

- $P(u) = (1\text{-}u)^3 \; P1$
  $+ \; 3u(1\text{-}u)^2 \; P2$
  $+ \; 3u^2(1\text{-}u) \; P3$
  $+ \; u^3 \; P4$

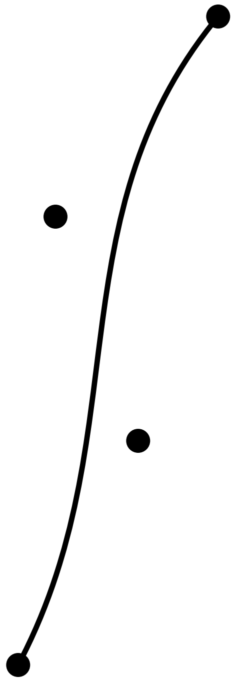
(Note! We relabeled *t* to *u*)



# How to Build Them? Here's an Idea

- $$\begin{aligned} P(u) = {} & (1-u)^3 && P1 \\ + {} & 3u(1-u)^2 && P2 \\ + {} & 3u^2(1-u) && P3 \\ + {} & u^3 && P4 \end{aligned}$$

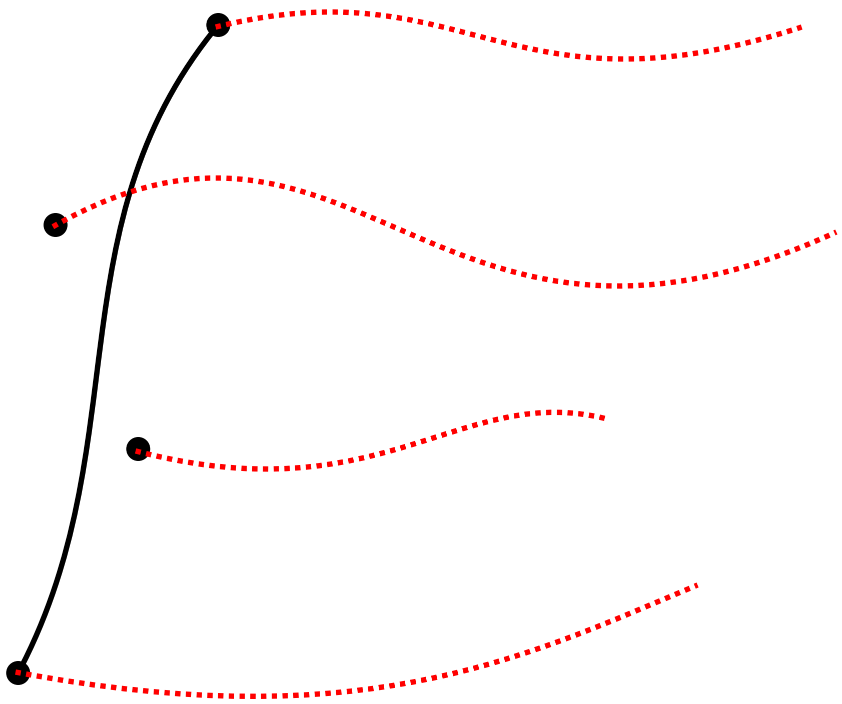
(Note! We relabeled *t* to *u*)



# How to Build Them? Here's an Idea

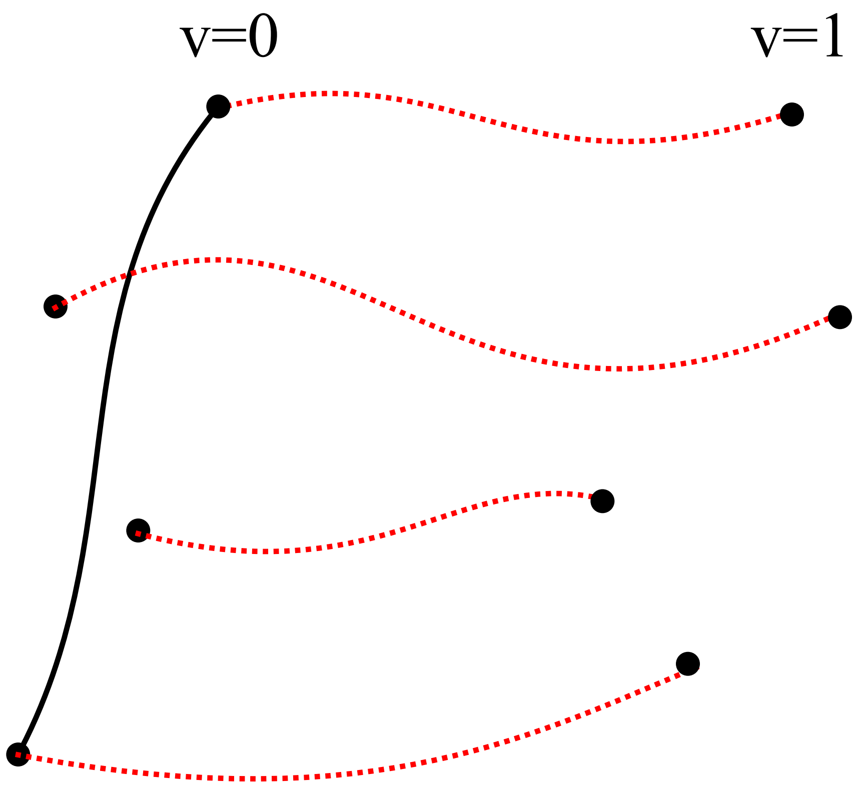
- $P(u) = (1\text{-}u)^3 \quad P1$
  $+ \quad 3u(1\text{-}u)^2 \; P2$
  $+ \quad 3u^2(1\text{-}u) \; P3$
  $+ \quad u^3 \quad P4$

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



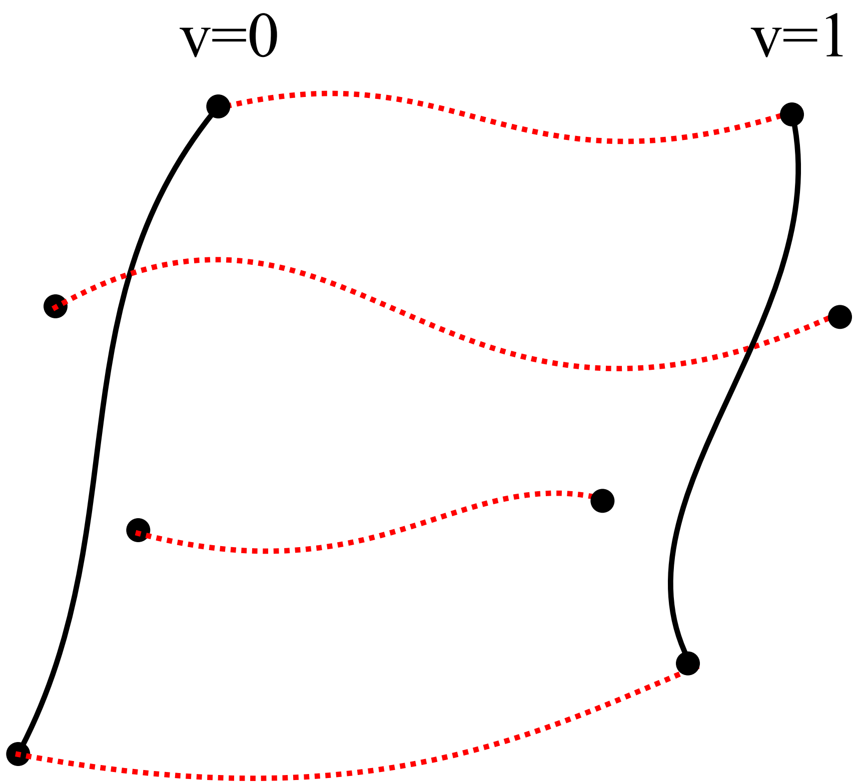
# Here's an Idea

- $P(u, v) = (1-u)^3 \quad P1(v)$
  $+ \quad 3u(1-u)^2 \quad P2(v)$
  $+ \quad 3u^2(1-u) \; P3(v)$
  $+ \quad u^3 \quad P4(v)$
- Let's make
  the Pis move along
  curves!



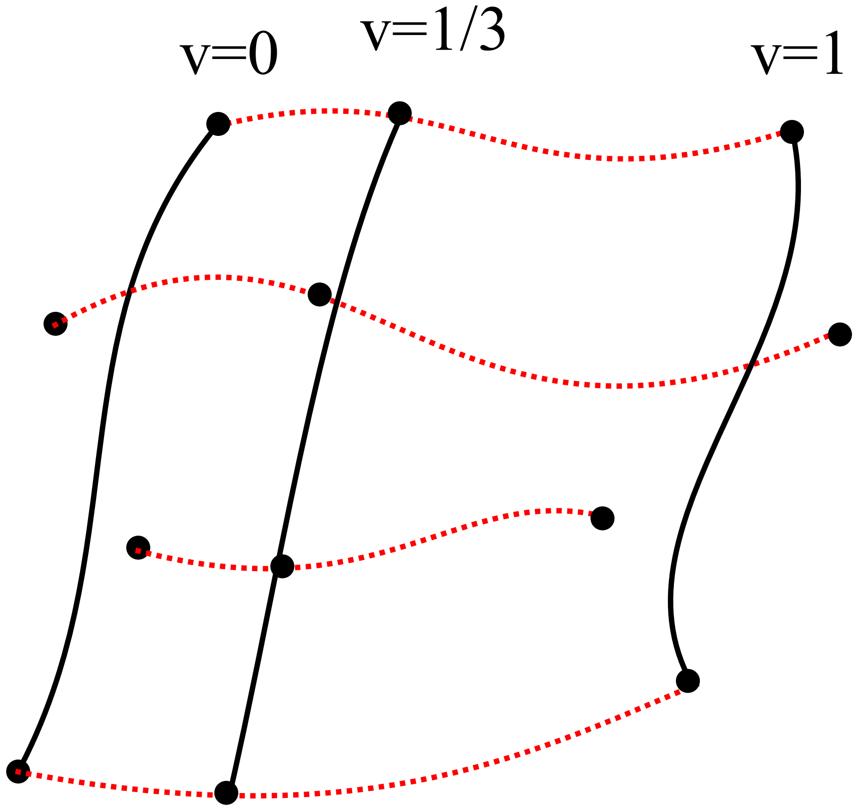
# Here's an Idea

- $P(u, v) = (1-u)^3 \quad P1(v)$
  $+ \quad 3u(1-u)^2 \quad P2(v)$
  $+ \quad 3u^2(1-u) \; P3(v)$
  $+ \quad u^3 \quad P4(v)$
- Let's make the Pis move along curves!



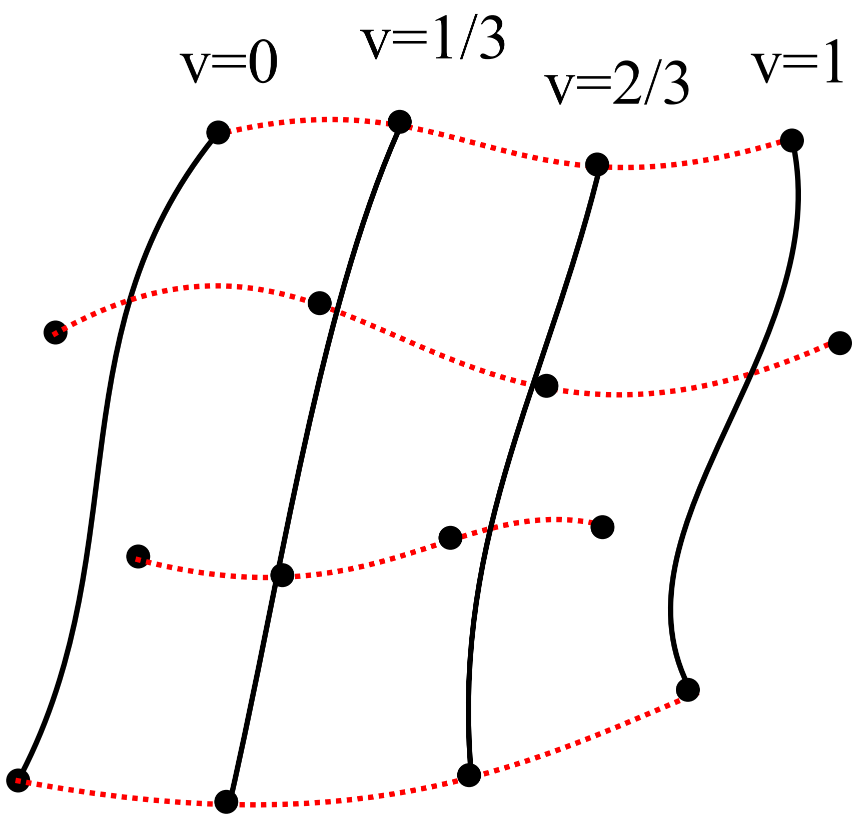
# Here's an Idea

- $$\begin{aligned} P(u, v) = \; & (1-u)^3 \; P1(v) \\ + \; & 3u(1-u)^2 \; P2(v) \\ + \; & 3u^2(1-u) \; P3(v) \\ + \; & u^3 \; P4(v) \end{aligned}$$
- Let's make the Pis move along curves!



# Here's an Idea

- $$P(u, v) = (1-u)^3 \quad P1(v) + 3u(1-u)^2 \quad P2(v) + 3u^2(1-u) \; P3(v) + u^3 \quad P4(v)$$
- Let's make the Pis move along curves!

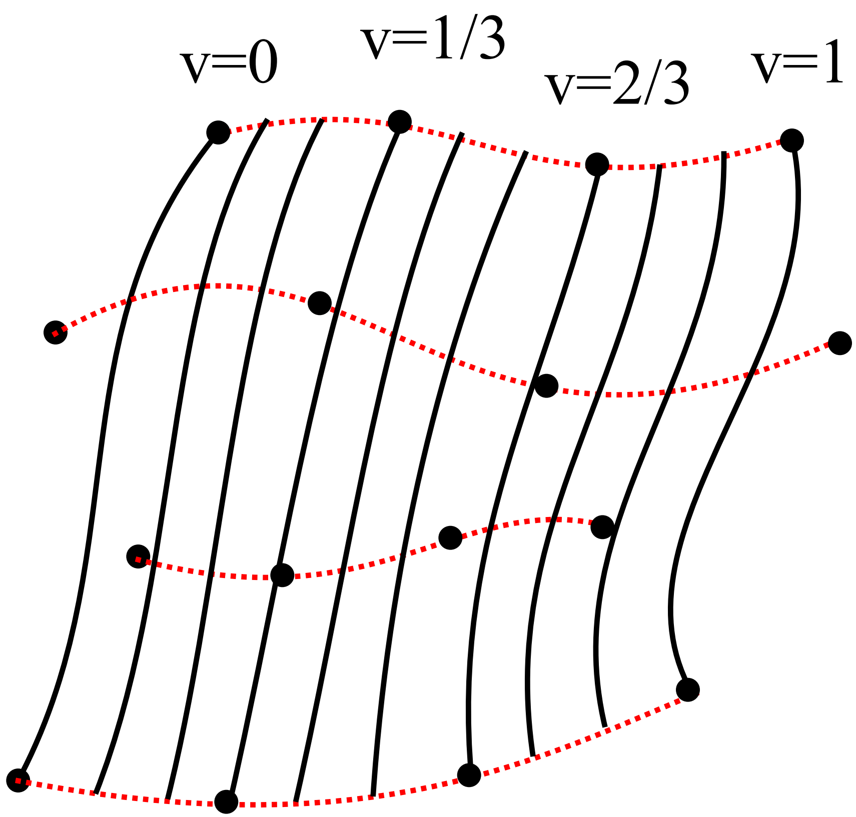


# Here's an Idea

- $P(u, v) = (1-u)^3 \; P1(v) + 3u(1-u)^2 \; P2(v) + 3u^2(1-u) \; P3(v) + u^3 \; P4(v)$

$$
\begin{aligned}
P(u, v) = \; & (1-u)^3 \; P1(v) \\
+ \; & 3u(1-u)^2 \; P2(v) \\
+ \; & 3u^2(1-u) \; P3(v) \\
+ \; & u^3 \; P4(v)
\end{aligned}
$$

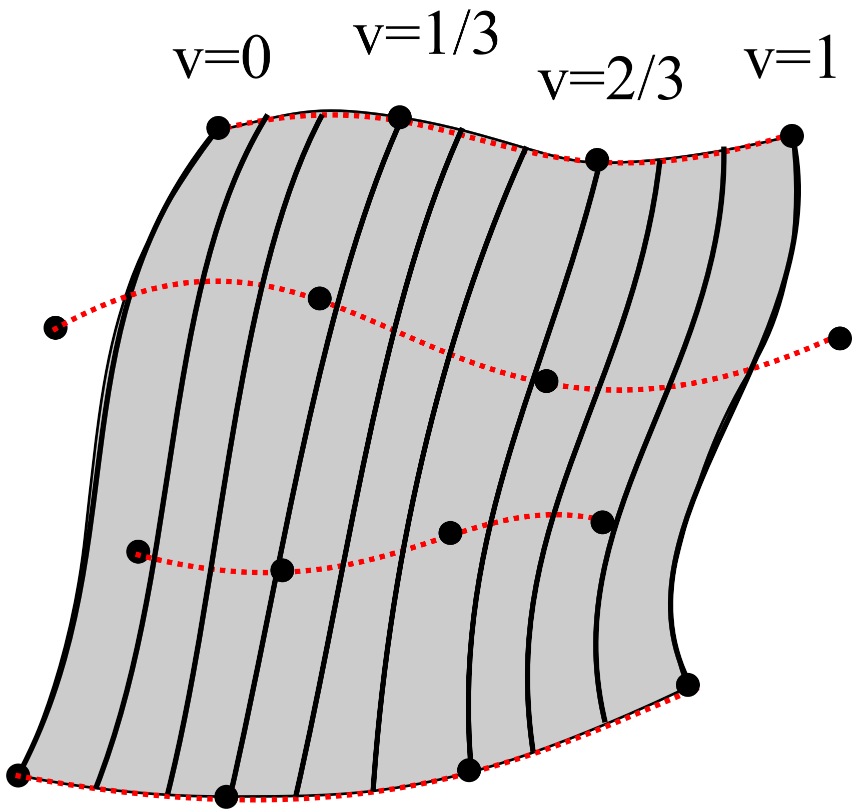
- Let's make the Pis move along curves!



# Here's an Idea

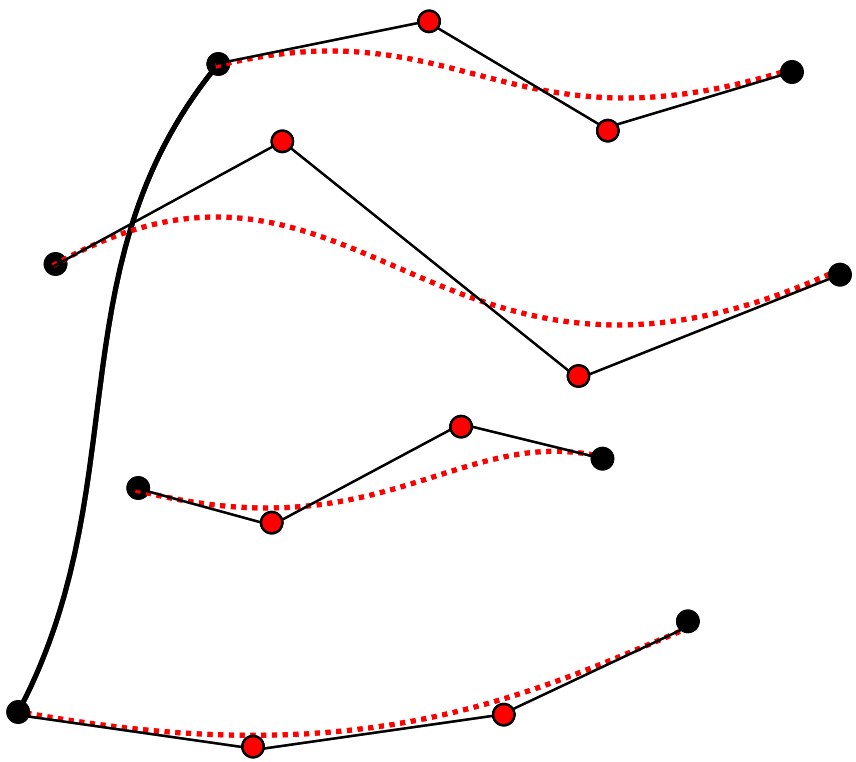
- $P(u, v) = (1-u)^3 \; P1(v) + 3u(1-u)^2 \; P2(v) + 3u^2(1-u) \; P3(v) + u^3 \; P4(v)$
- Let's make the Pis move along curves!

A 2D surface patch!



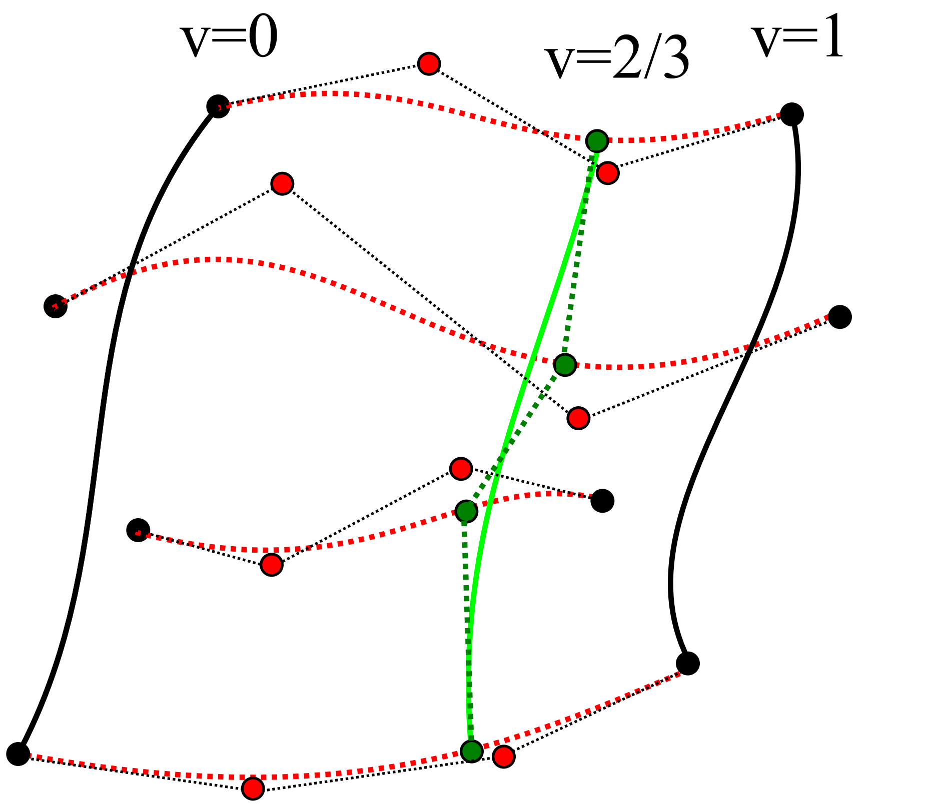
# Tensor Product Bézier Patches

- In the previous, $P_i$s were just some curves
- What if we make **them** Bézier curves?



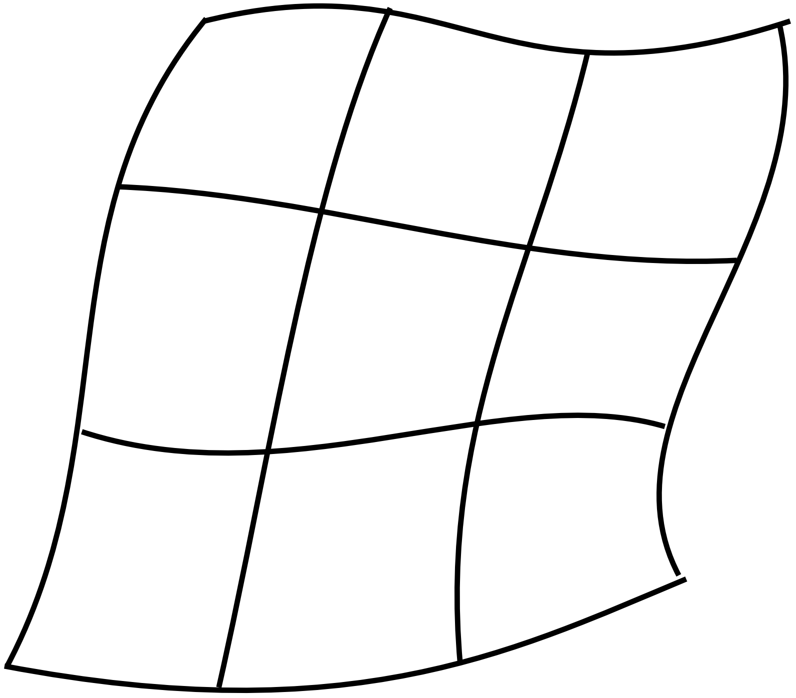
# Tensor Product Bézier Patches

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



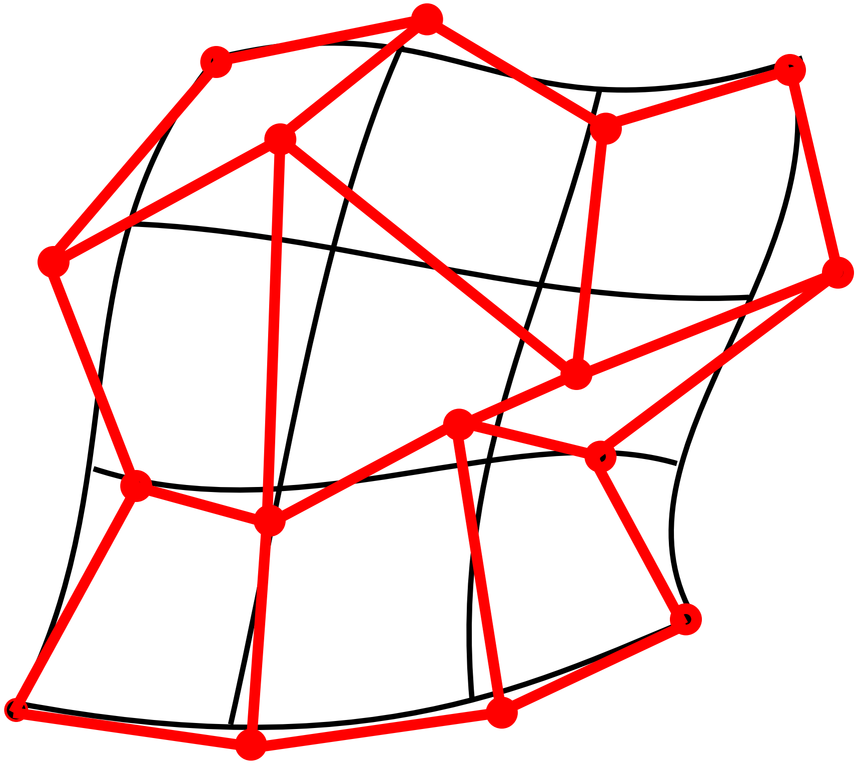
# Tensor Product Bézier Patches

**A bicubic Bézier surface**



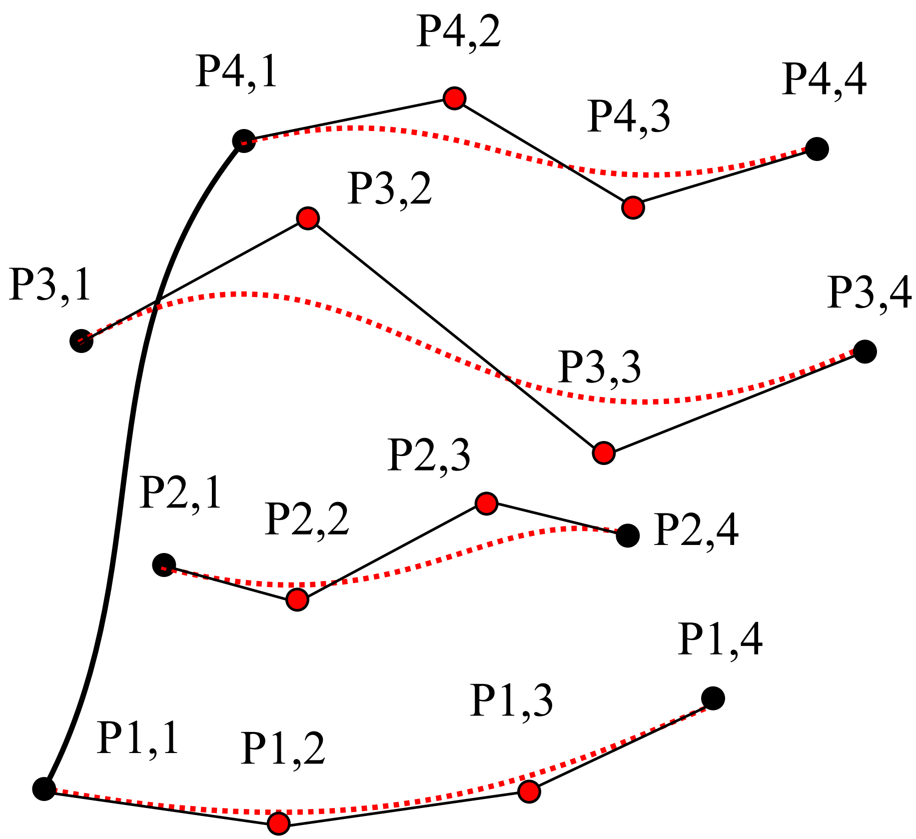
# Tensor Product Bézier Patches

The “Control Mesh”
16 control points



# Bicubics, Tensor Product

- P(u,v) = B1(u) * P1(v)
  \+ B2(u) * P2(v)
  \+ B3(u) * P3(v)
  \+ B4(u) * P4(v)
- Pi(v) = B1(v) * Pi,1
  \+ B2(v) * Pi,2
  \+ B3(v) * Pi,3
  \+ B4(v) * Pi,4



# Bicubics, Tensor Product

- P(u,v) = B1(u) * P1(v)
  + B2(u) * P2(v)
  + B3(u) * P3(v)
  + B4(u) * P4(v)
- Pi(v) = B1(v) * Pi,1
  + B2(v) * Pi,2
  + B3(v) * Pi,3
  + B4(v) * Pi,4

$$P(u,v) =$$

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

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

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

# Bicubics, Tensor Product

- P(u,v) = B1(u) * P1(v)
  + B2(u) * P2(v)
  + B3(u) * P3(v)
  + B4(u) * P4(v)
- Pi(v) = B1(v) * Pi,1
  + B2(v) * Pi,2
  + B3(v) * Pi,3
  + B4(v) * Pi,4

$$P(u, v) =$$

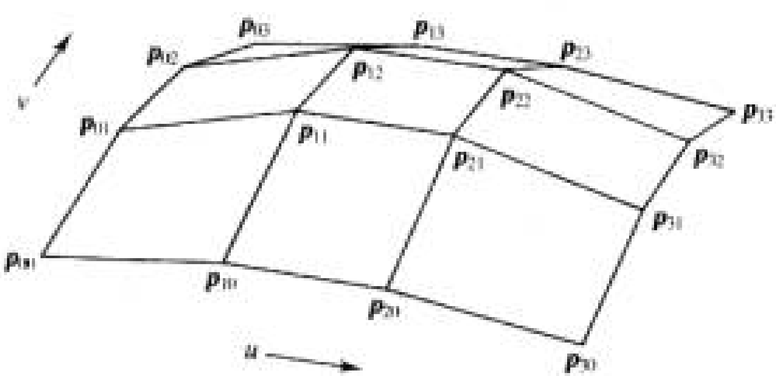
16 control points Pi,j
16 2D basis functions Bi,j

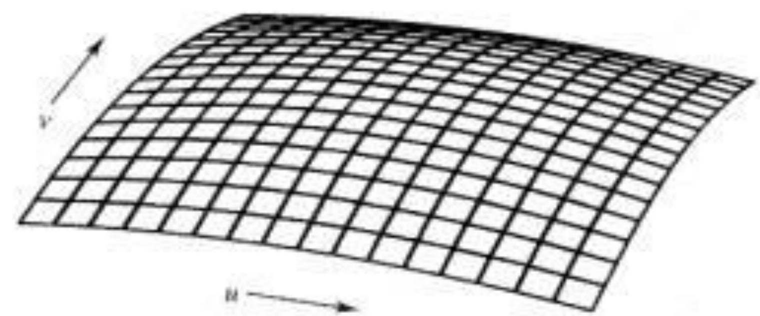
$$= \sum_{i=1}^{4} \sum_{j=1}^{4} \boxed{P_{i,j}} B_{i,j}(u, v)$$

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

# Recap: Tensor Bézier Patches

- Parametric surface P(*u*,*v*) is a bicubic polynomial of two variables *u* & *v*
- Defined by 4x4=16 control points P1,1, P1,2.... P4,4
- Interpolates 4 corners, approximates others
- Basis are product of two Bernstein polynomials: B1(*u*)B1(*v*); B1(*u*)B2(*v*);... B4(*u*)B4(*v*)

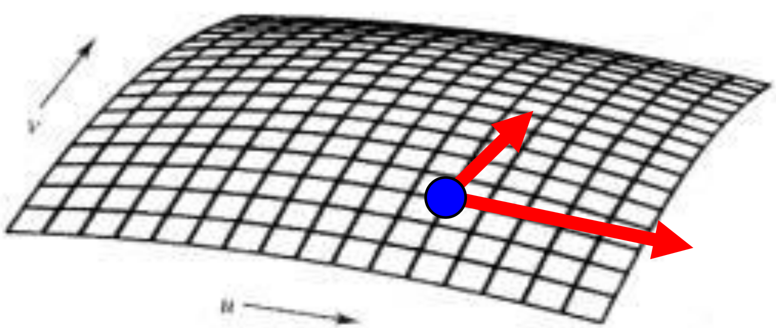


# Questions?

# Tangents and Normals for Patches

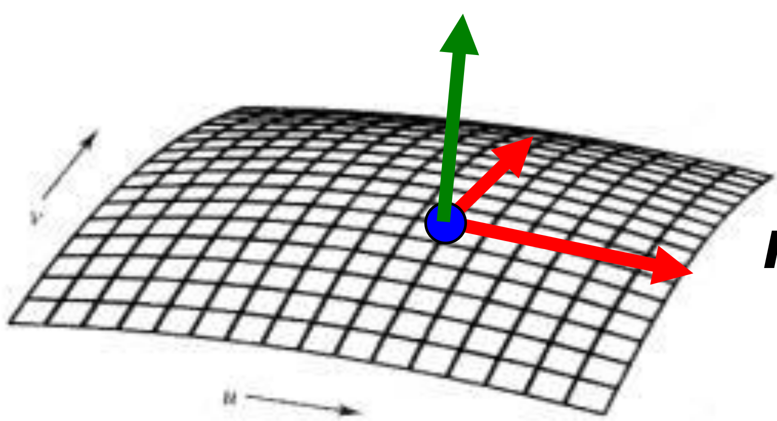
- P(u,v) is a 3D point specified by *u*, *v*
- The partial derivatives $\partial P/\partial u$ and $\partial P/\partial v$ are 3D vectors
  - Both are tangent to surface at P



# 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!**

# Questions?

# Recap: Matrix Notation for Curves

- Cubic Bézier in matrix notation

point on curve
(2x1 vector)

Canonical
"power basis"

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

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

"Geometry matrix"
of control points P1..P4
(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*)

$$P^x(u,v) =$$

Column vector of basis functions (*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 ***G***x, ***G***y, ***G***z matrices into a geometry **tensor** of control points
  - I.e., Gki,j = the *kth* coordinate of control point Pi,j
  - A cube of numbers!

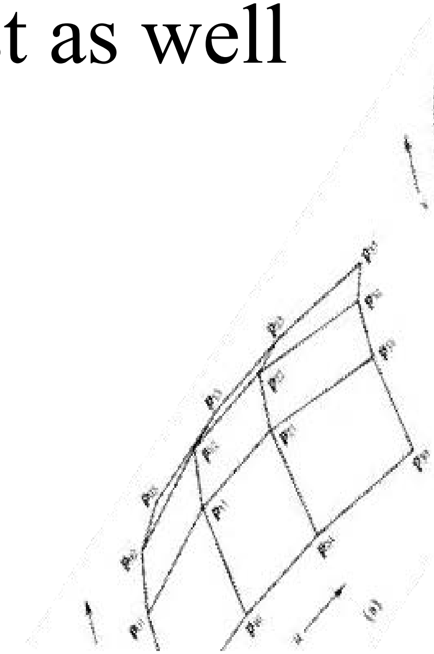
$$P^k(u,v) = \boldsymbol{T}^l(u)\,\boldsymbol{B}_l^i\,\boldsymbol{G}_{ij}^k\,\boldsymbol{B}_m^j\,\boldsymbol{T}^m(v)$$

- "Definitely not required, but nice!
  - See http://en.wikipedia.org/wiki/Multilinear_algebra

# 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
  - Yes, simple!



# Tensor Product Spline Patches

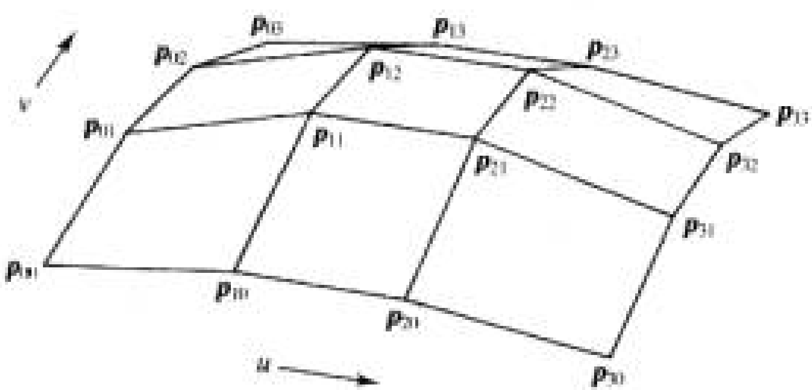
- Pros
  - Smooth
  - Defined by reasonably small set of points
- Cons
  - Harder to render (usually converted to triangles)
  - Tricky to ensure continuity at patch boundaries
- Extensions
  - Rational splines: Splines in homogeneous coordinates
  - NURBS: Non-Uniform Rational B-Splines
    - Like curves: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

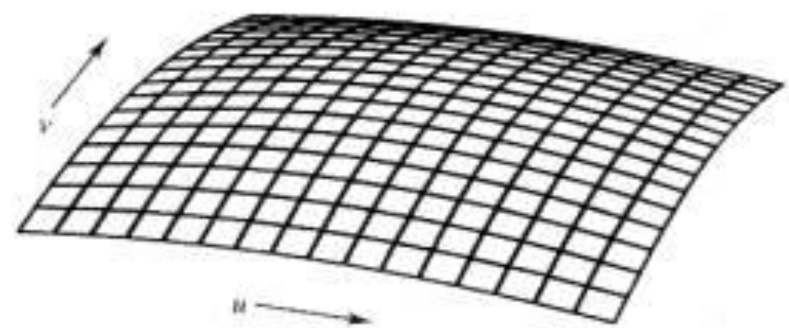
# Utah Teapot: Tensor Bézier Splines

- Designed by Martin Newell

# Cool: Displacement Mapping

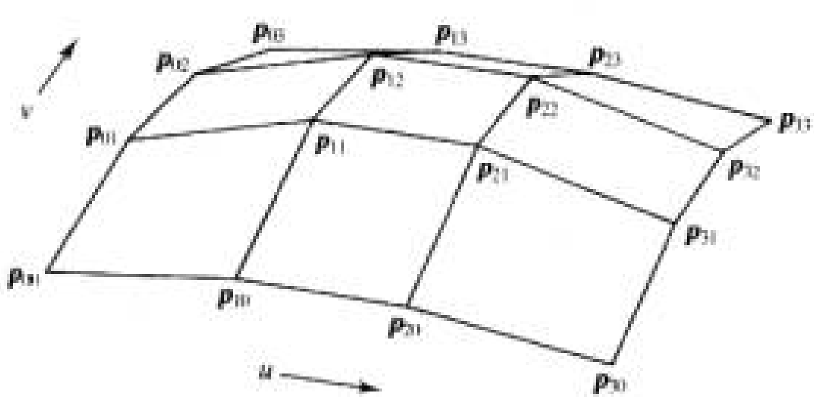
- Not all surfaces are smooth...

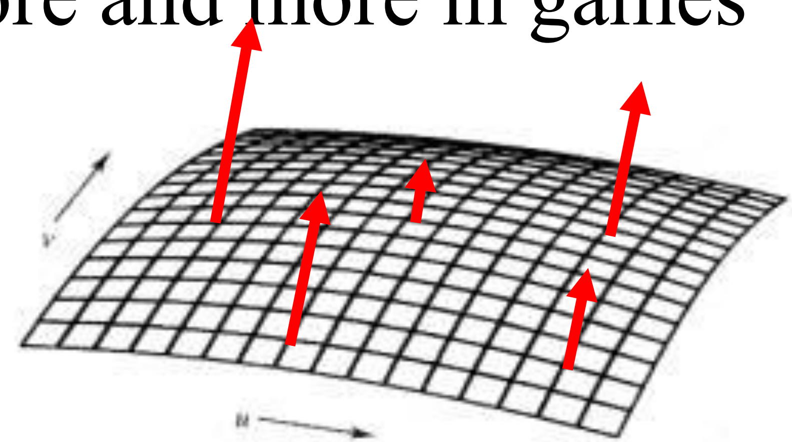




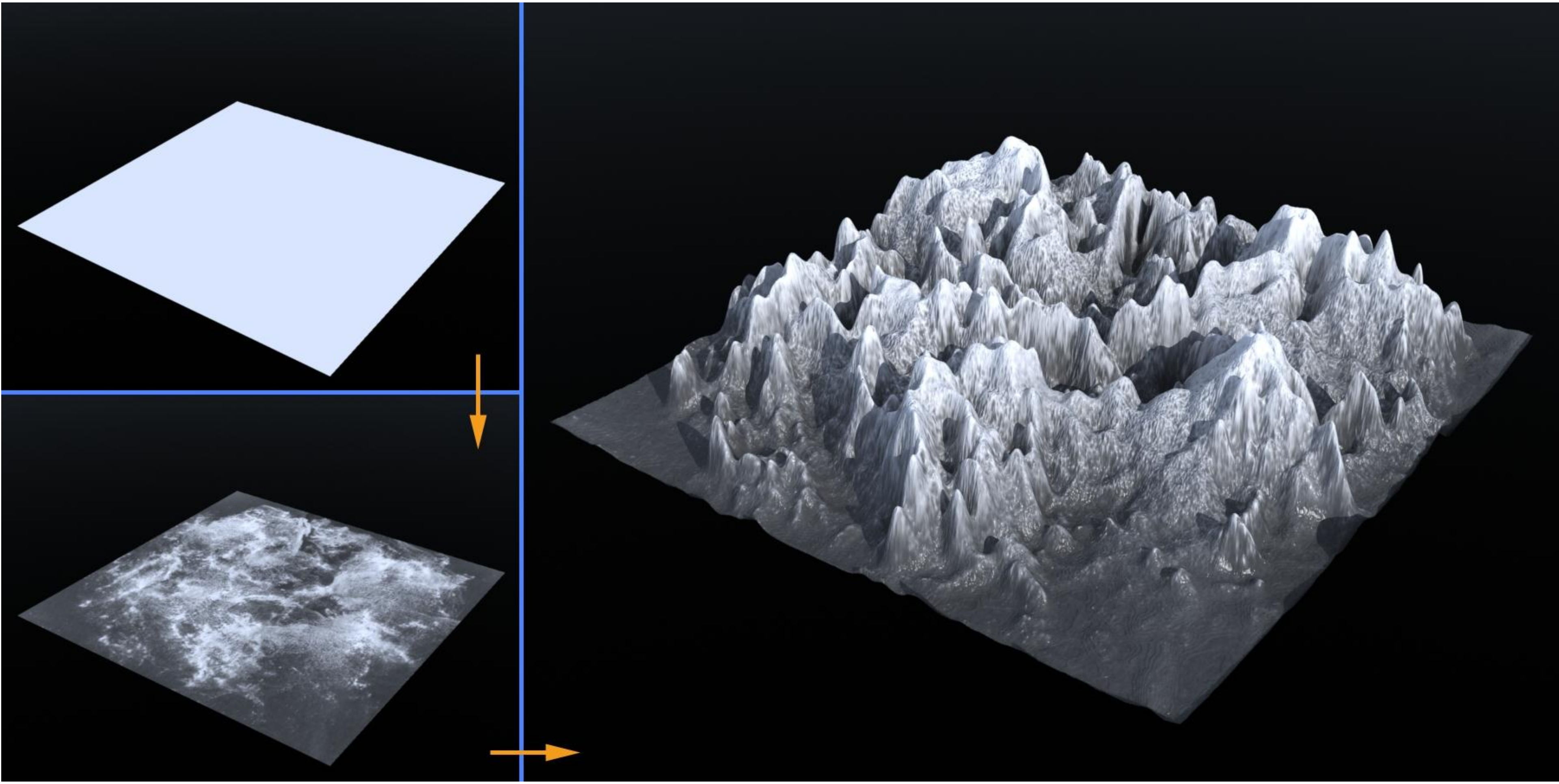
# Cool: Displacement Mapping

- 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

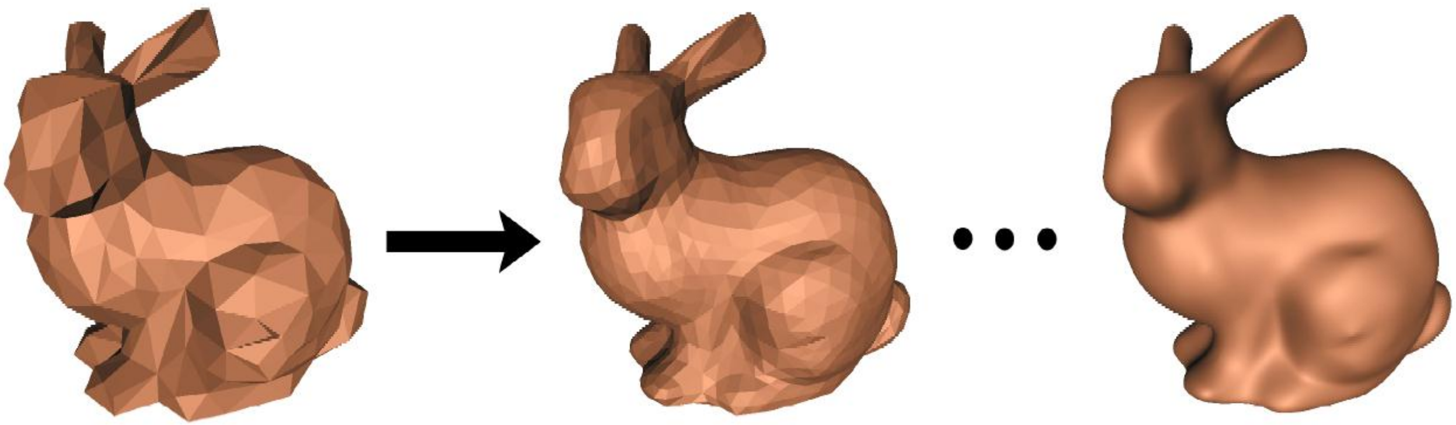


This image is in the public domain. Source: Wikimedia Commons.

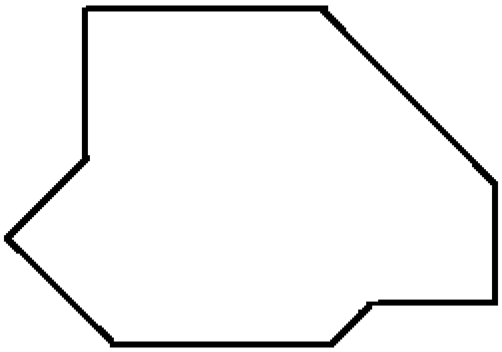
# Questions?

# Subdivision Surfaces

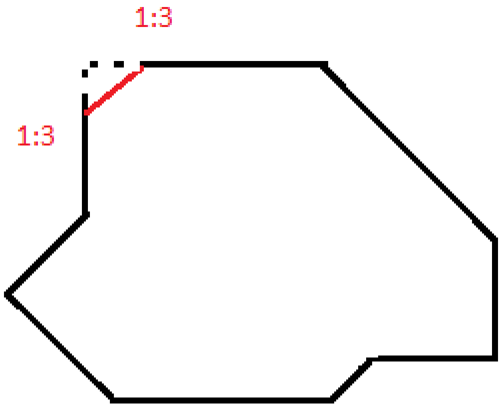
- Start with polygonal mesh
- Subdivide into larger number of polygons, smooth result after each subdivision
  - Lots of ways to do this.
- The limit surface is smooth!



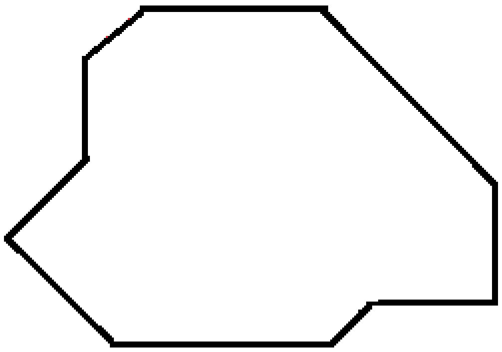
# Corner Cutting



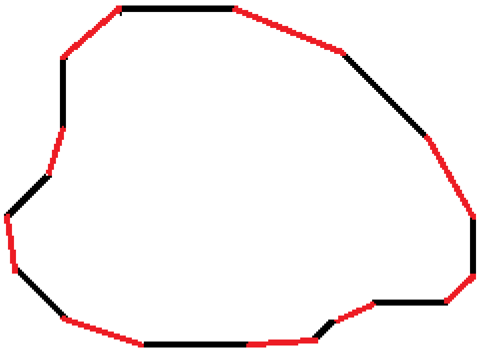
# Corner Cutting



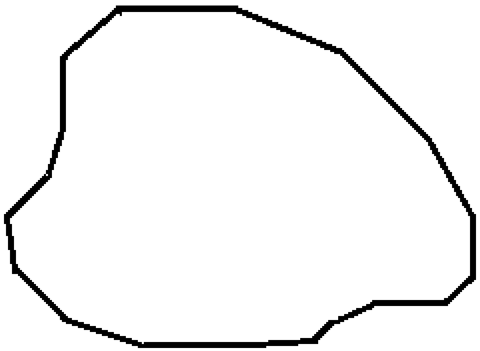
# Corner Cutting



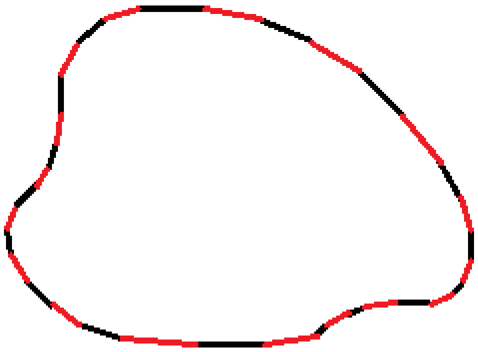
# Corner Cutting



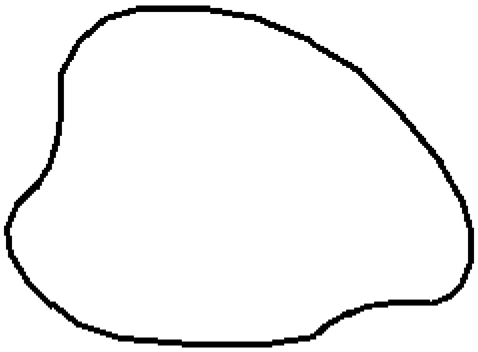
# Corner Cutting



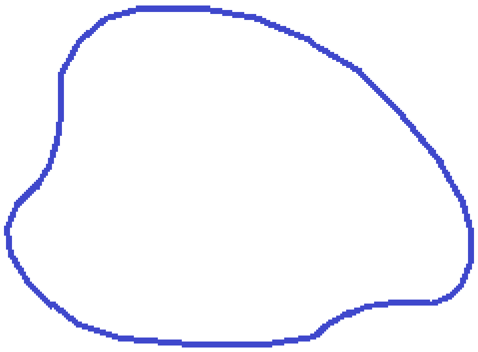
# Corner Cutting



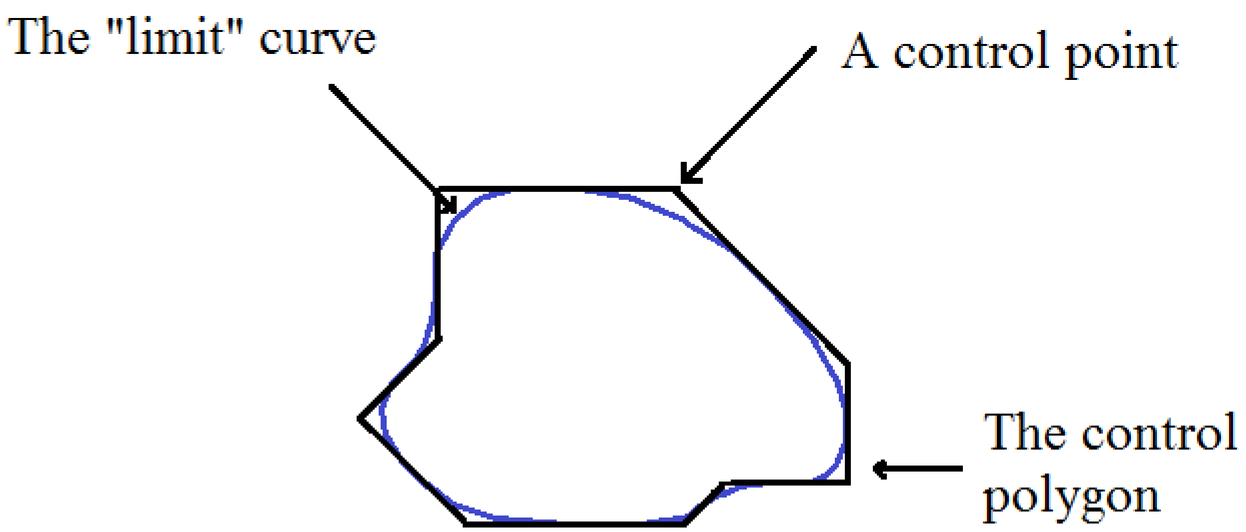
# Corner Cutting



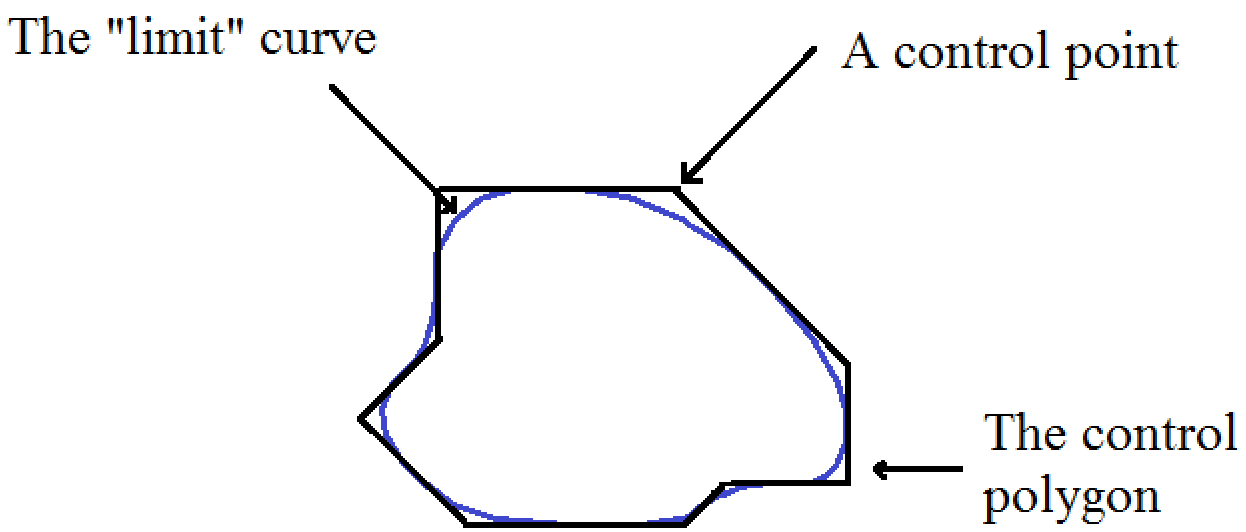
# Corner Cutting



# Corner Cutting

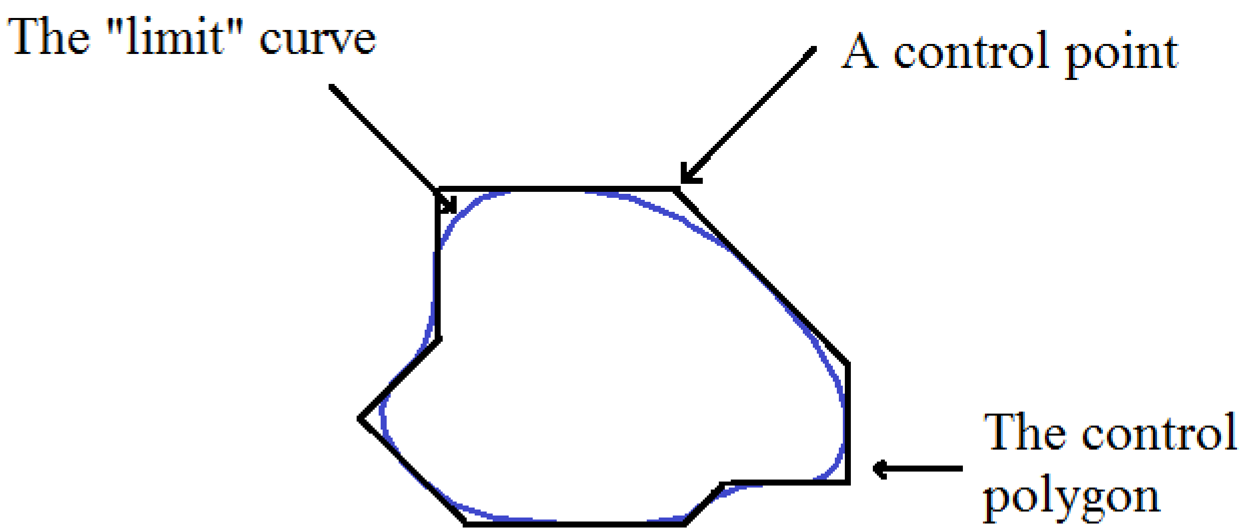


# Corner Cutting




**It turns out corner cutting (Chaikin's Algorithm) produces a quadratic B-Spline curve! (Magic!)**

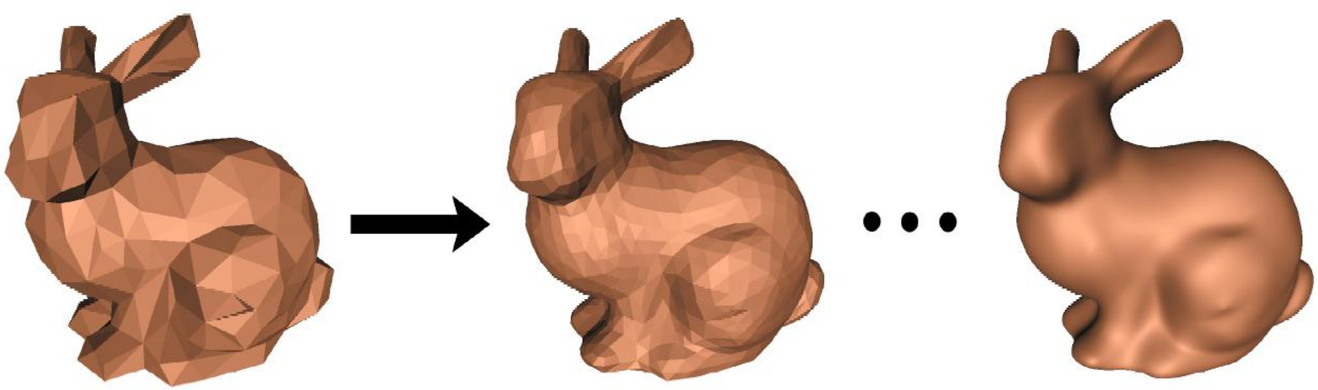
# Corner Cutting




**(Well, not totally unexpected, remember de Casteljau)**

# Subdivision Curves and Surfaces

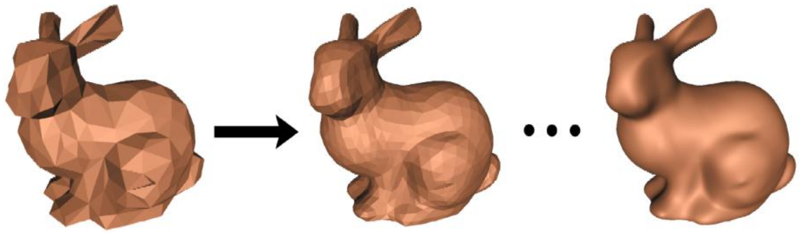
- Idea: cut corners to smooth
- Add points and compute weighted average of neighbors
- Same for surfaces
  - Special case for irregular vertices
    - vertex with more or less than 6 neighbors in a triangle mesh



Warren et al.

# Subdivision Curves and Surfaces

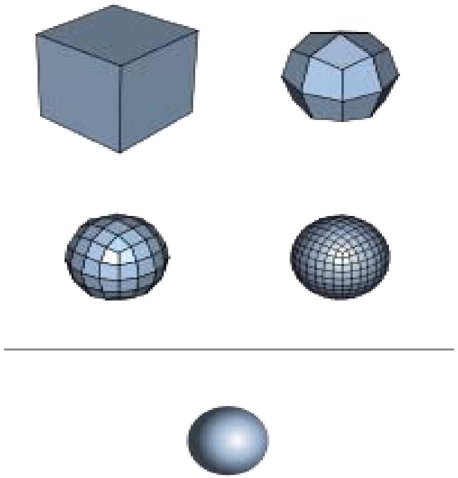
- Advantages
  - Arbitrary topology
  - Smooth at boundaries
  - Level of detail, scalable
  - Simple representation
  - Numerical stability, well-behaved meshes
  - Code simplicity
- Little disadvantage:
  - Procedural definition
  - Not parametric
  - Tricky at special vertices




Warren et al.

# Flavors of Subdivision Surfaces

- Catmull-Clark
  - Quads and triangles
  - Generalizes bicubics to arbitrary topology!
- Loop, Butterfly
  - Triangles
- Doo-Sabin, sqrt(3), biquartic...
  - and a whole host of others
- Used **everywhere** in movie and game modeling!
- See http://www.cs.nyu.edu/~dzorin/sig00course/



# Subdivision + Displacement

Original rough mesh

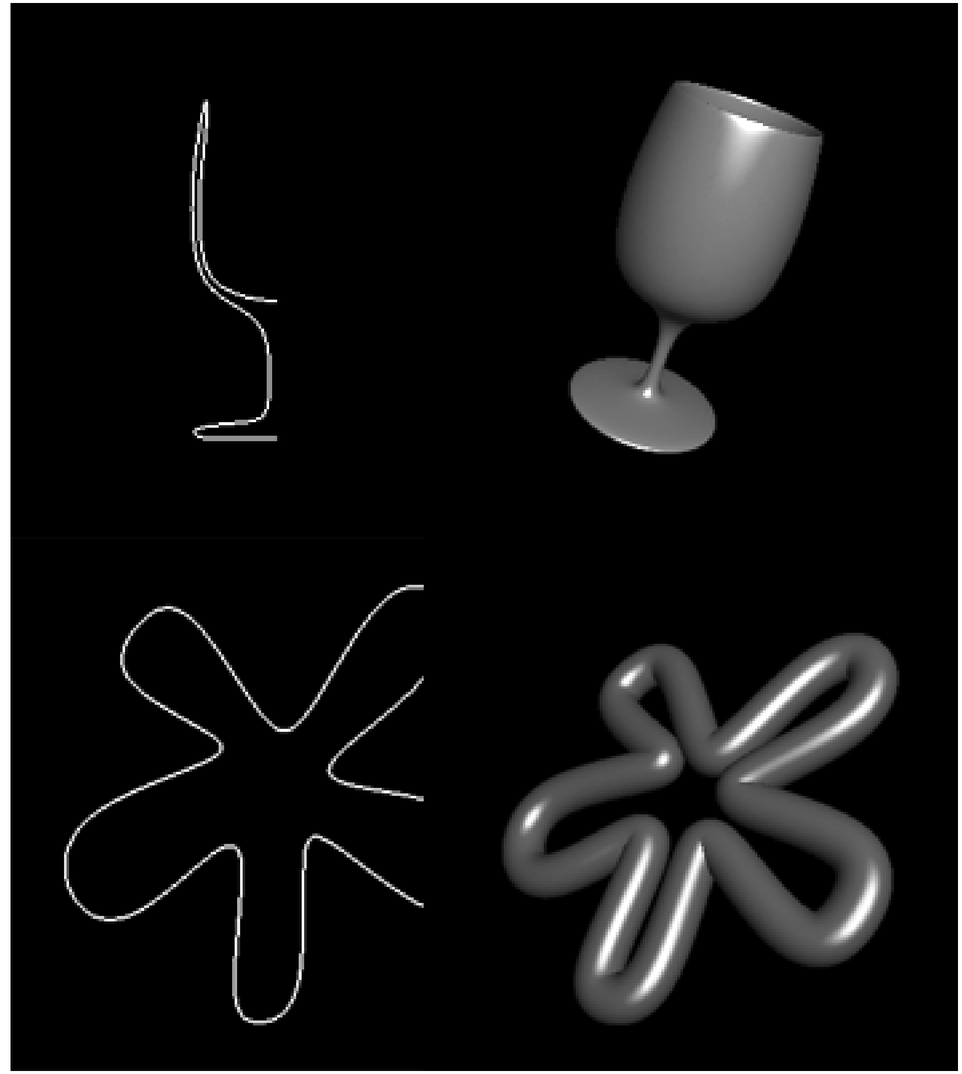
Original mesh with subdivision

Original mesh with subdivision and displacement

# Questions?

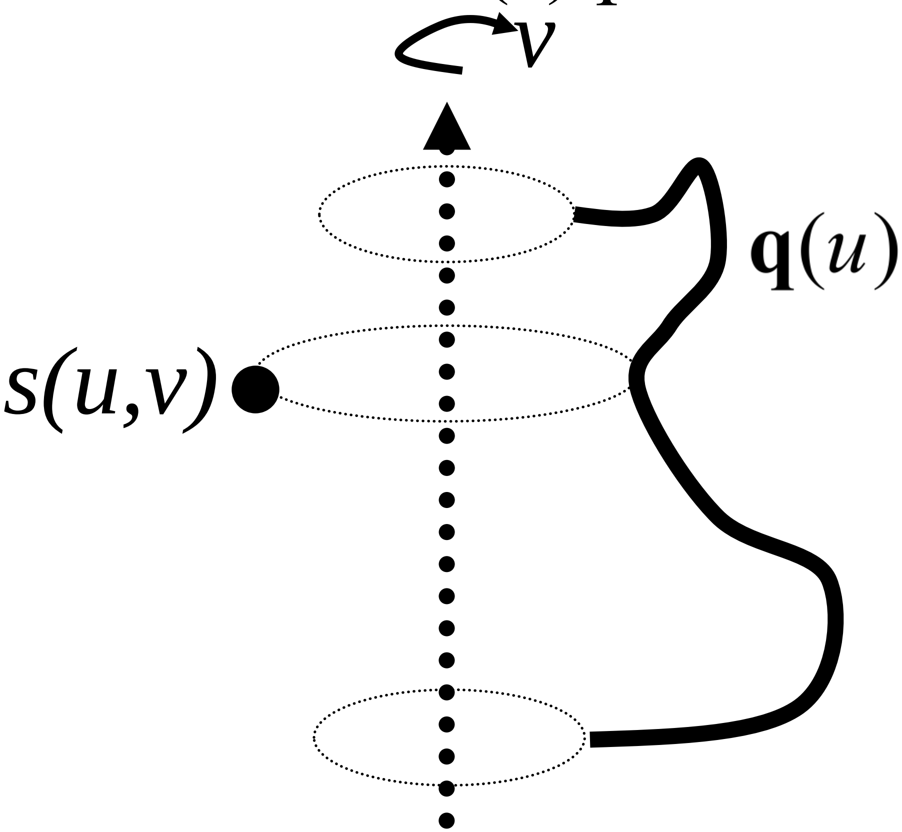
# Specialized Procedural Definitions

- Surfaces of revolution
  – Rotate given 2D profile curve
- Generalized cylinders
  – Given 2D profile and 3D curve, sweep the profile along the 3D curve
- **Assignment 1!**



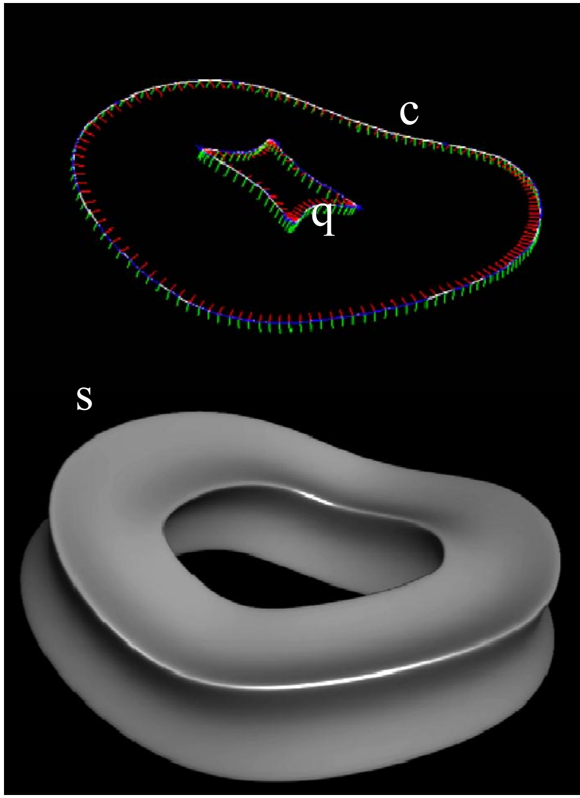
# Surface of Revolution

- 2D curve q($u$) provides one dimension
  - Note: works also with 3D curve
- Rotation R($v$) provides 2nd dimension




$\boldsymbol{s}(u,v)=\boldsymbol{R}(v)\boldsymbol{q}(u)$
where $\boldsymbol{R}$ is a matrix,
$\boldsymbol{q}$ a vector,
and $\boldsymbol{s}$ is a point on the surface

# General Swept Surfaces



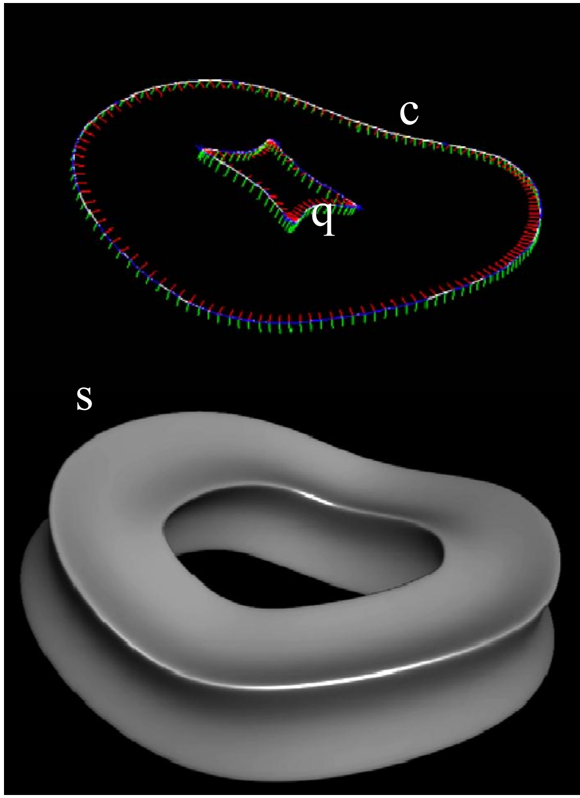

- Trace out surface by moving a profile curve along a trajectory.
  - profile curve $\mathbf{q}(u)$ provides one dim
  - trajectory $\mathbf{c}(u)$ provides the other
- Surface of revolution can be seen as a special case where trajectory is a circle

$$\boldsymbol{s}(u,v)=\boldsymbol{M}(\boldsymbol{c}(v))\boldsymbol{q}(u)$$

where $\boldsymbol{M}$ is a matrix that depends on the trajectory $\boldsymbol{c}$

# General Swept Surfaces

- How do we get **M**?
  - Translation is easy, given by ***c****(v)*
  - What about orientation?
- Orientation options:
  - Align profile curve with an axis.
  - **Better**: Align profile curve with frame that "follows" the curve

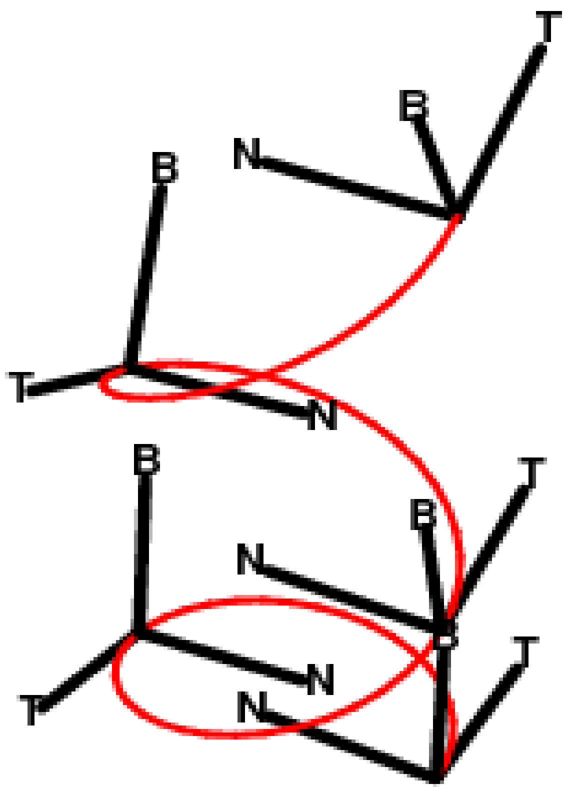



$$\boldsymbol{s}(u,v)=\boldsymbol{M}(\boldsymbol{c}(v))\boldsymbol{q}(u)$$

where $\boldsymbol{M}$ is a matrix that depends on the trajectory $\boldsymbol{c}$

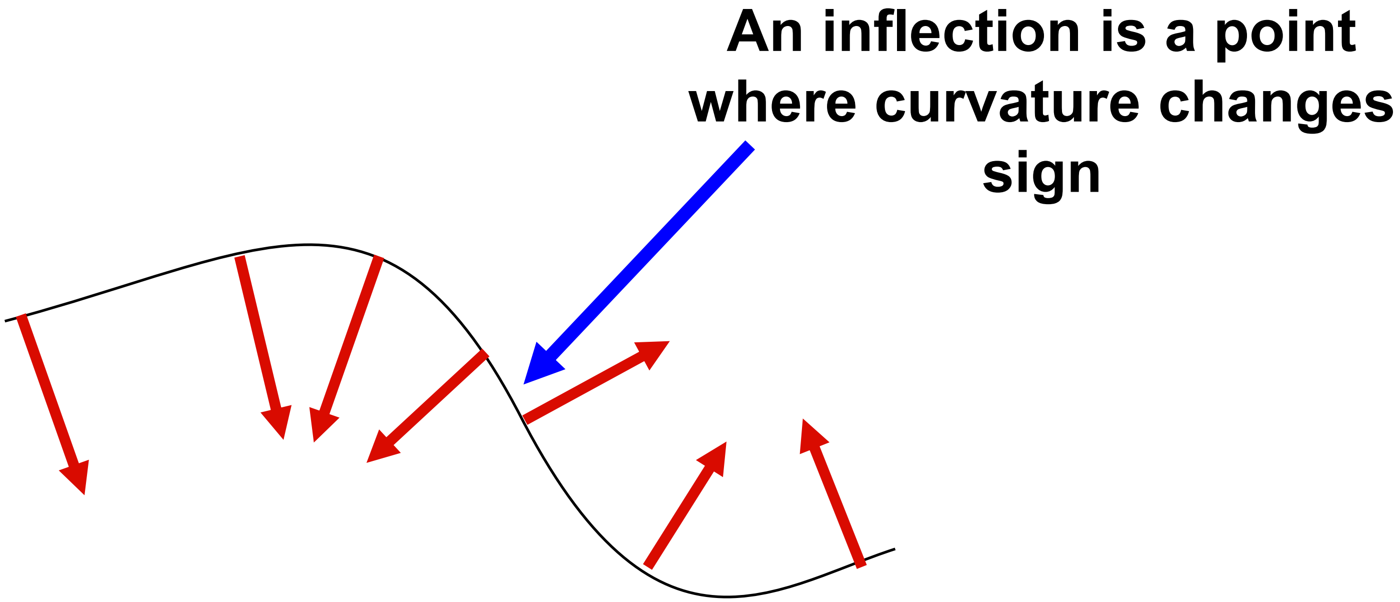
# Frames on Curves: Frenet Frame

- Frame defined by 1st (tangent), 2nd and 3rd derivatives of a 3D curve
- Looks like a good idea for swept surfaces...

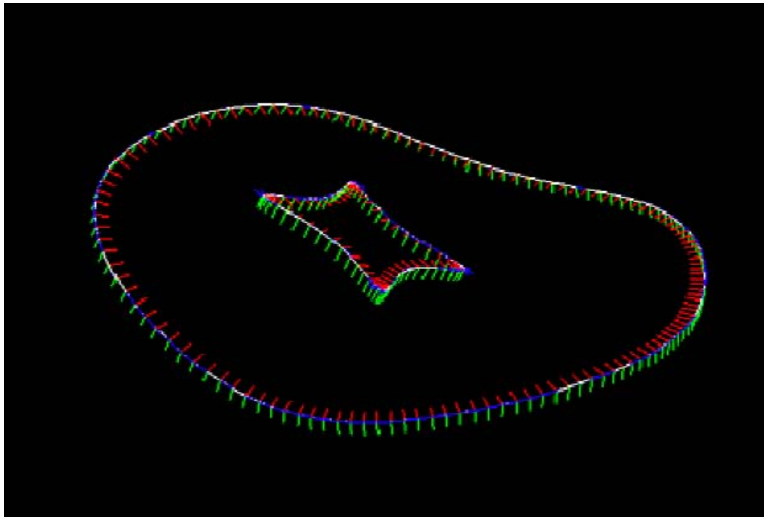


# Frenet: Problem at Inflection!

- Normal flips!
- Bad to define a smooth swept surface



# Smooth Frames on Curves



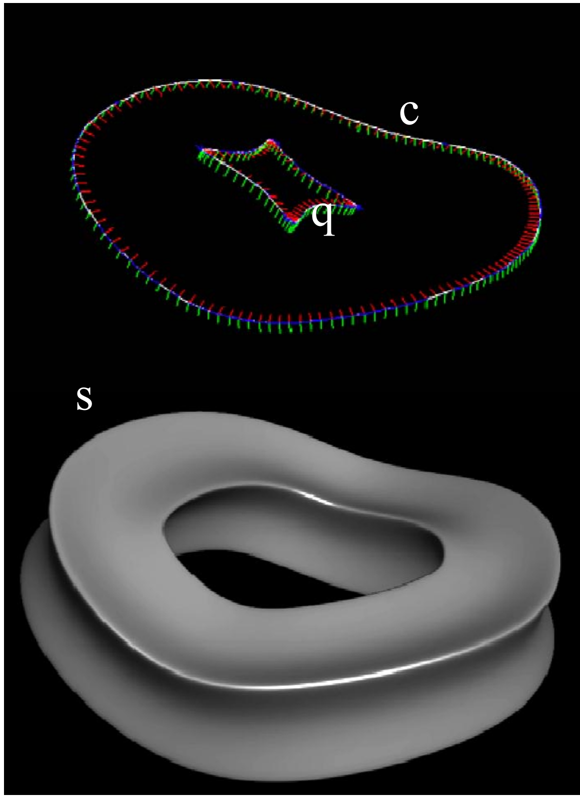
- Build triplet of vectors
  - include tangent (it is reliable)
  - orthonormal
  - coherent over the curve
- Idea:
  - use cross product to create orthogonal vectors
  - exploit discretization of curve
  - use previous frame to bootstrap orientation
  - **See Assignment 1 instructions!**

# Normals for Swept Surfaces

- Need partial derivatives w.r.t. both $u$ and $v$

$$n = (\partial \mathbf{s}/\partial u) \times (\partial \mathbf{s}/\partial v)$$

  - *Remember to normalize!*
- One given by tangent of profile curve, the other by tangent of trajectory




$$\boldsymbol{s}(u,v)=\boldsymbol{M}(\boldsymbol{c}(v))\boldsymbol{q}(u)$$

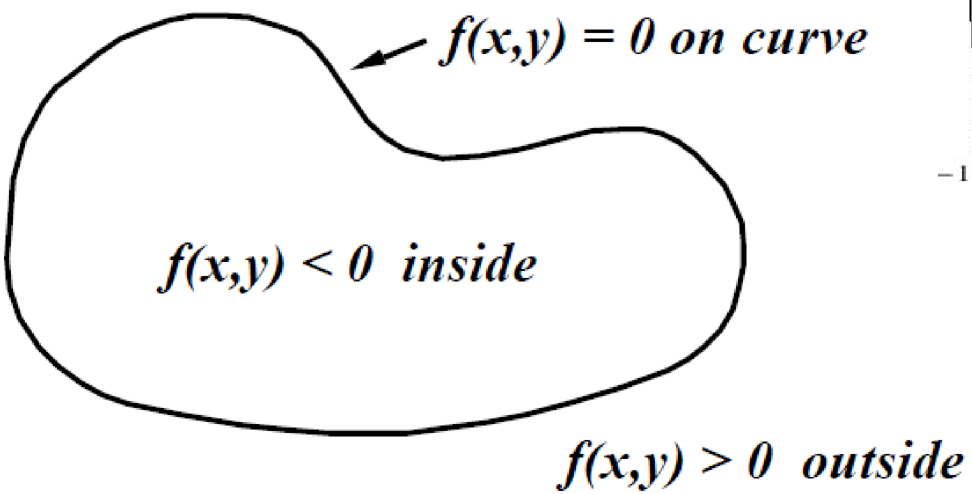
where $\boldsymbol{M}$ is a matrix that depends on the trajectory $\boldsymbol{c}$

# Questions?

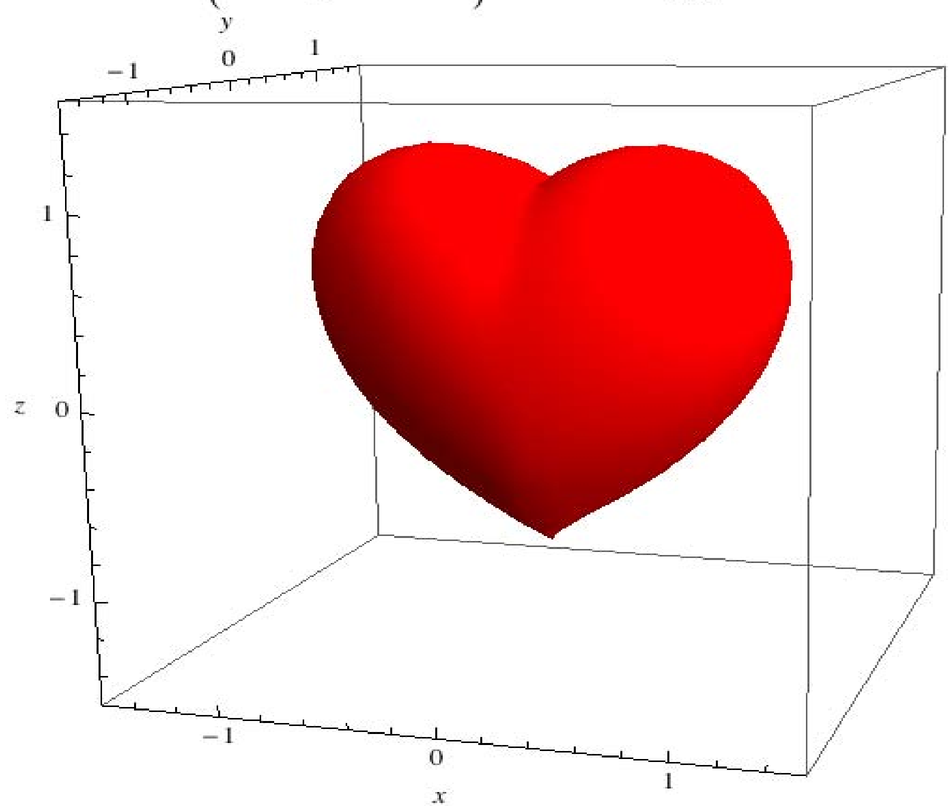
# Implicit Surfaces

- Surface defined implicitly by a function

f (x, y, z) = 0 (on surface)
f (x, y, z) < 0 (inside)
f (x, y, z) > 0 (outside)

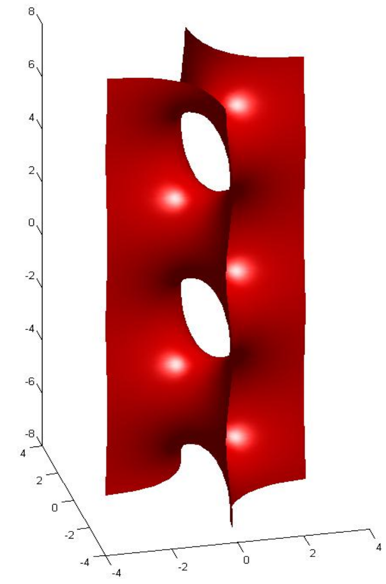



$$\left(x^2 + \frac{9y^2}{4} + z^2 - 1\right)^3 - x^2 z^3 - \frac{9y^2 z^3}{200} = 0$$



# Implicit Surfaces

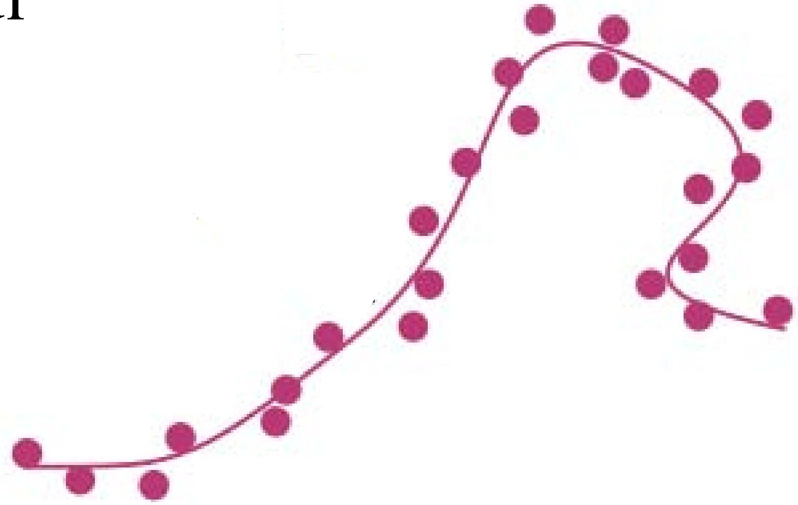
- Pros:
  - Efficient check whether  point is inside
  - Efficient Boolean operations
  - Can handle weird topology for animation
  - Easy to do sketchy modeling
- Cons:
  - Does not allow us to easily generate a point on the surface



# Questions?

# Point Set Surfaces

- Given only a noisy 3D point cloud (no connectivity), can you define a reasonable surface using only the points?
  – Laser range scans only give you points, so this is potentially useful



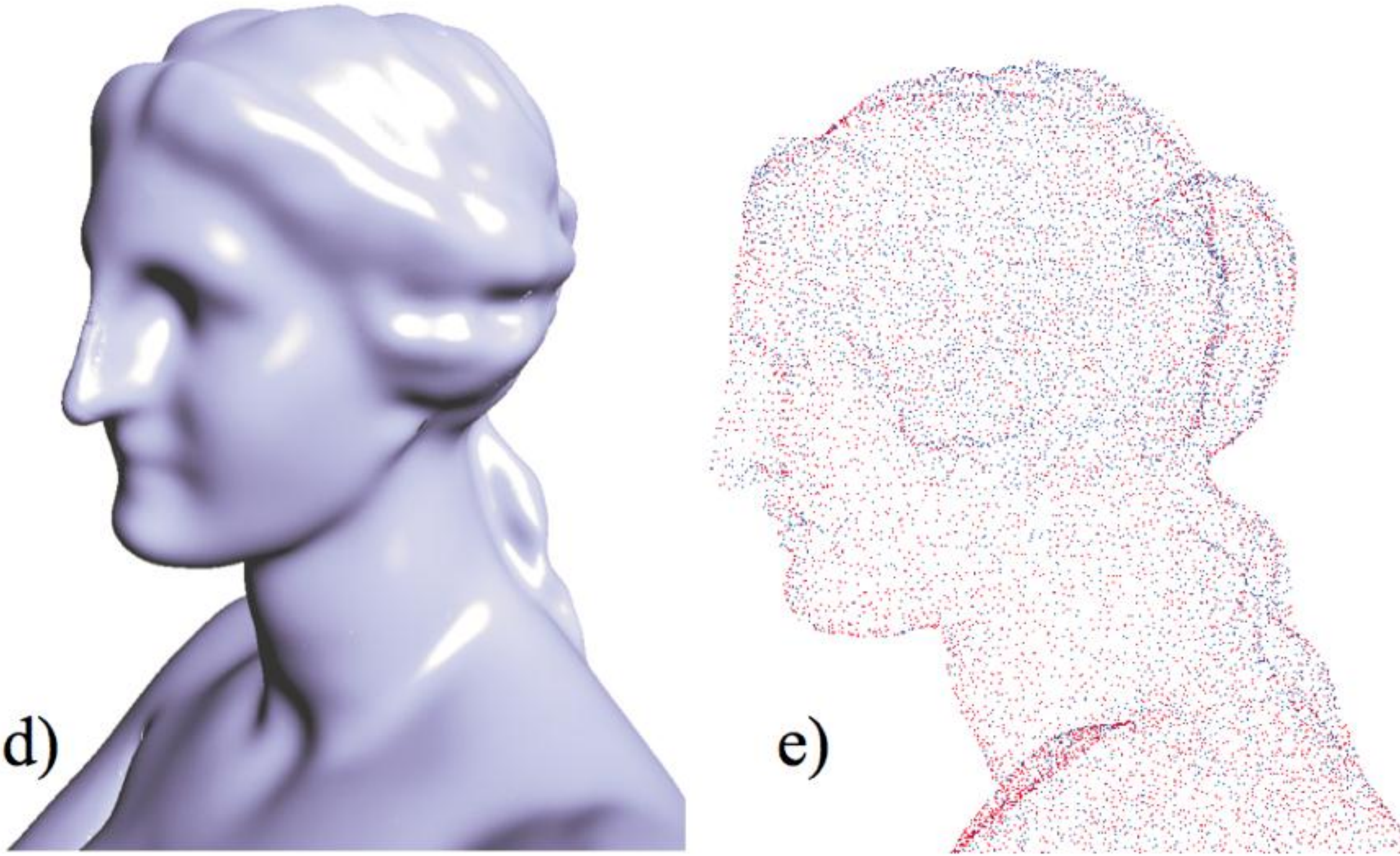


From Point Set Surfaces, (Alexa et al. 2001).

# Point Set Surfaces

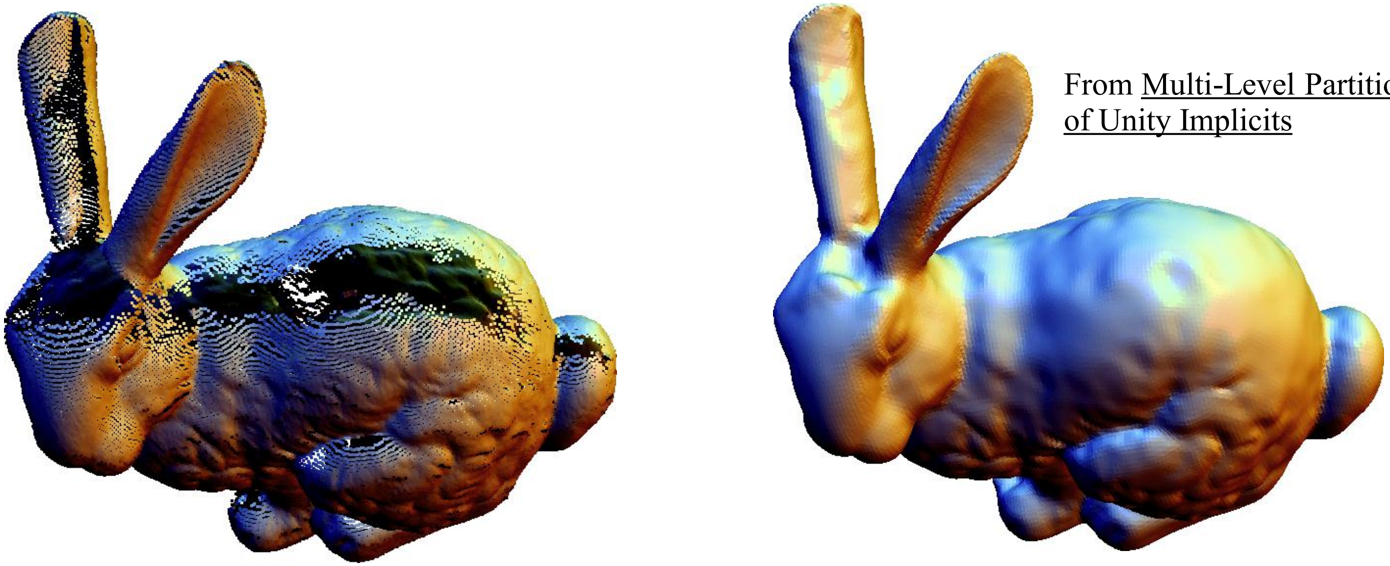
From <u>Point Set Surfaces</u>, used with permission from ACM, Inc

Alexa et al. 2001



# Point Set Surfaces

- Modern take on implicit surfaces
- Cool math: Moving Least Squares (MLS), partitions of unity, etc.



From Multi-Level Partition of Unity Implicits

Ohtake et al. 2003



- Not required in this class, but nice to know.

# Questions?

# That’s All for Today

- Further reading
  - Buss, Chapters 7 & 8
- Subvision curves and surfaces
  - http://www.cs.nyu.edu/~dzorin/sig00course/

6.837 Computer Graphics
Fall 2012