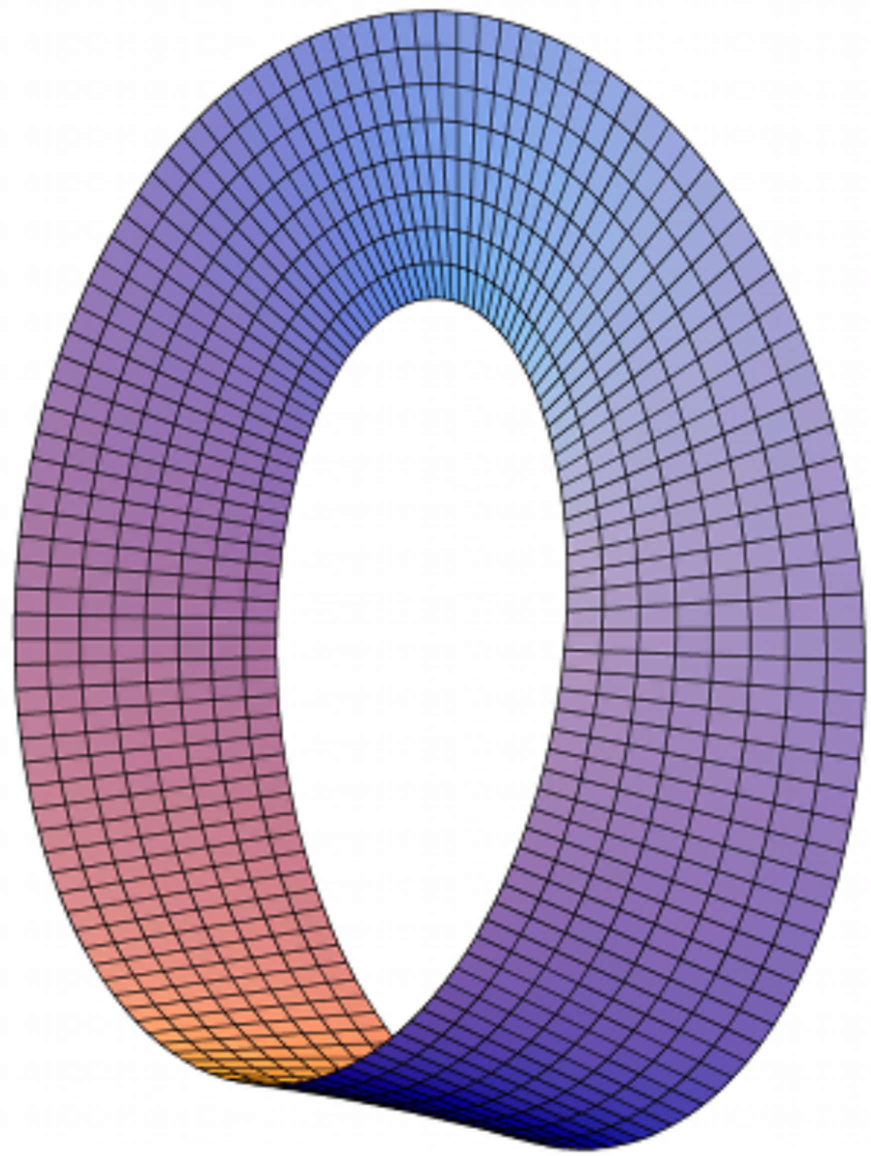
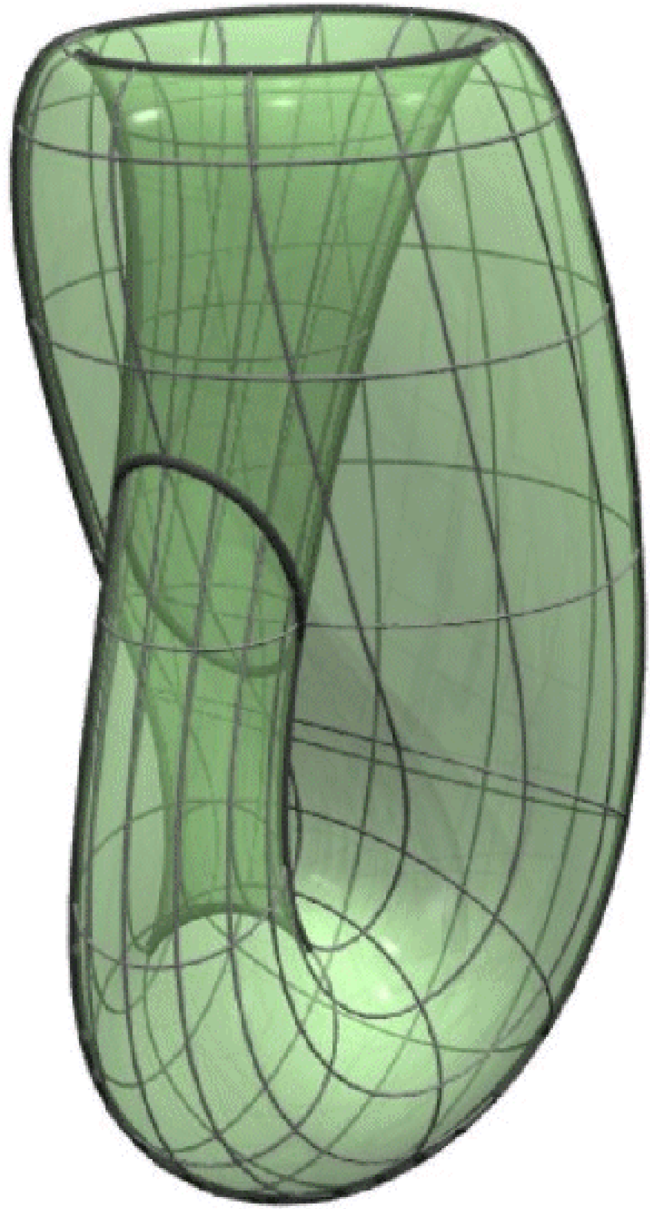
Curves and Surfaces

CS475 / 675, Fall 2016

Siddhartha Chaudhuri



Möbius strip: 1 surface, 1 edge



Klein bottle: 1 surface, no edges

(Möbius strip: Inductiveload@Wikipedia)

Curves and Surfaces

- *Curve*: 1D set
  - Generally defined as $\mathbf{f}:\mathbb{R}\rightarrow\mathrm{X}$, where X is some space
- *Surface*: 2D set
  - Generally defined as $\mathbf{f}:\mathbb{R}^2\rightarrow\mathrm{X}$
- X is the space in which the set is *embedded*
  - Dimension of curve/surface $\neq$ Dimension of X!
  - E.g. plane is 2D surface embedded in 3D

Parametric Curves

- $\mathbf{p} = \mathbf{f}(t)$
  - $\mathbf{f}(\ldots)$ is a *vector-valued function*
- Line: $\mathbf{p} = t\mathbf{u} + \mathbf{p}_0$
  - $\mathbf{u}$ is direction of line, $\mathbf{p}_0$ is any point on the line
  - Ray: $t \geq 0$
  - Line segment: $t \in [0, 1]$
- Circle: $(x, y) = (r \cos t, r \sin t)$

Parametric Curves

Parametric curve $\mathbf{f}(\mathit{time})$ traced out by a stunt plane

Parametric Surfaces

- $\mathbf{p} = \mathbf{f}(s, t)$
- Plane: $\mathbf{p} = s\mathbf{u} + t\mathbf{v} + \mathbf{p}_0$
  - $\mathbf{u}$, $\mathbf{v}$ are any two directions in the plane
  - $\mathbf{p}_0$ is any point on it
- Sphere: $(x, y, z) = (r \cos s \sin t, r \sin s \sin t, r \cos t)$
- **Note:** (s, t) provide a set of texture coordinates for the surface
- A d-dimensional set is defined with d parameters

Implicit Forms

- Curve embedded in 2D: $f(x, y) = 0$
  - $f(\ldots)$ is a *scalar-valued function*
  - Line: $ax + by + 1 = 0$
  - Circle: $x^2 + y^2 - r^2 = 0$
- Surface embedded in 3D: $f(x, y, z) = 0$
  - Plane: $ax + by + cz + 1 = 0$
  - Sphere: $x^2 + y^2 + z^2 - r^2 = 0$
- In general, an implicitly defined set consists of points $\mathbf{p}$ s.t. $f(\mathbf{p}) = 0$

Implicit Forms

- Also called *level set* or *isocontour*
  - Usually written as $f(\mathbf{p}) = c$, which can be recast to the standard form: $f(\mathbf{p}) - c = 0$

Level sets of the Earth's terrain

$$height(x, y) = constant$$

(Banaue rice terraces, the Philippines)

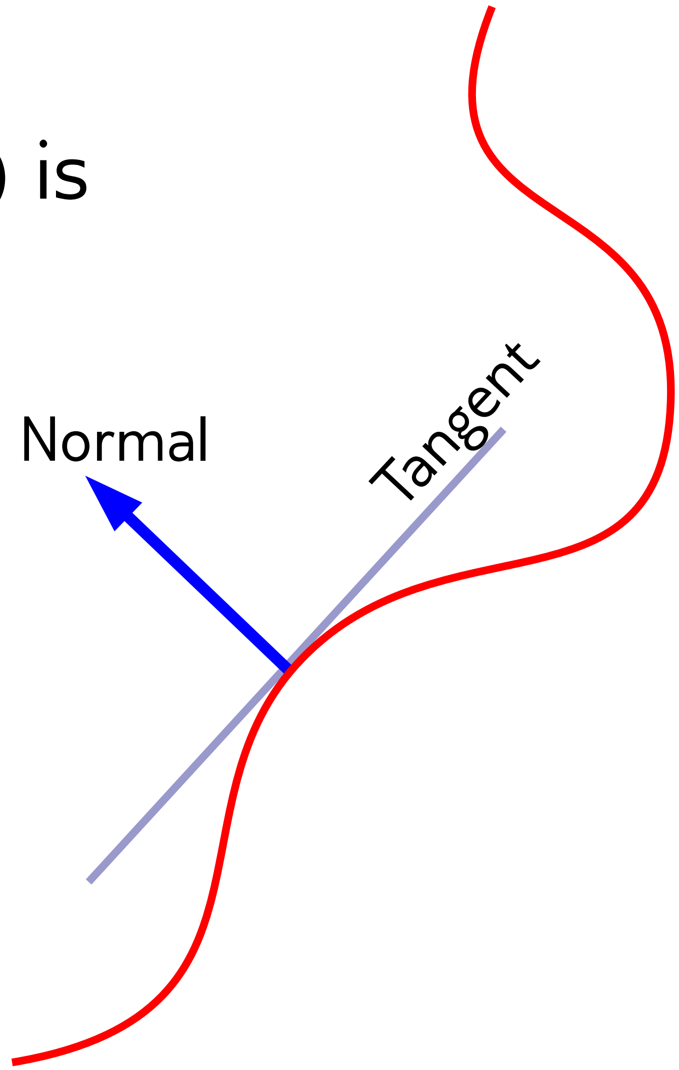
Normal to Curve Embedded in 2D

- From parametric form:
  Normal to $\mathbf{p} = \mathbf{f}(t) = (x(t), y(t))$ is

$$\left(-\frac{\mathrm{d}\,y}{\mathrm{d}\,t}, \frac{\mathrm{d}\,x}{\mathrm{d}\,t} \right)$$

- From implicit form:
  Normal to $f(x, y) = 0$ is

$$\left(\frac{\partial\, f}{\partial\, x}, \frac{\partial\, f}{\partial\, y} \right)$$



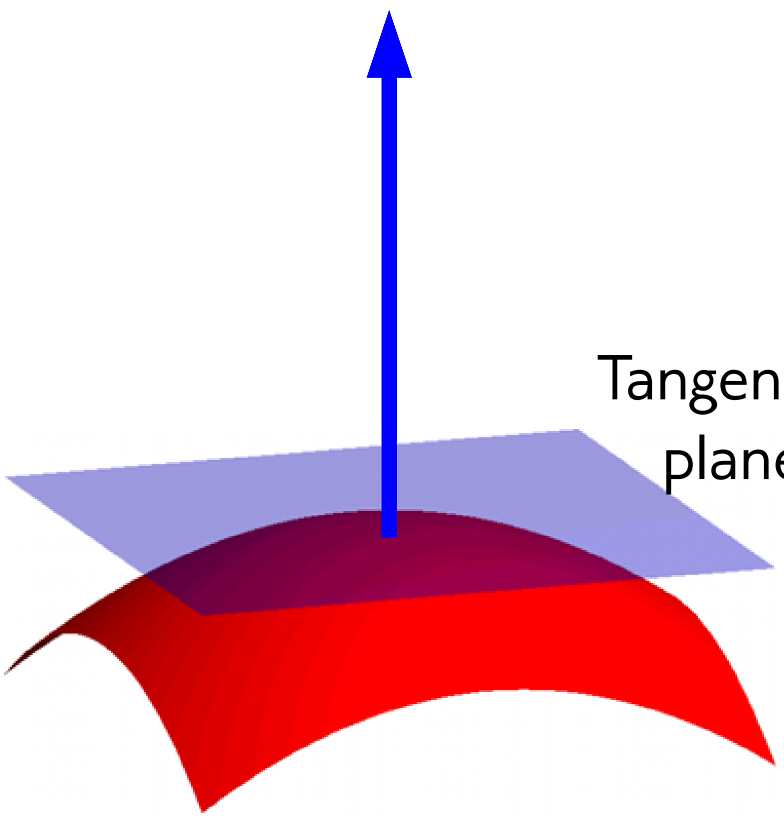
Normal to Surface Embedded in 3D

- From parametric form:
  Normal to $\mathbf{p} = \mathbf{f}(s, t)$ is

$$\frac{\partial \mathbf{f}}{\partial s} \times \frac{\partial \mathbf{f}}{\partial t}$$

- From implicit form:
  Normal to $f(x, y, z) = 0$ is

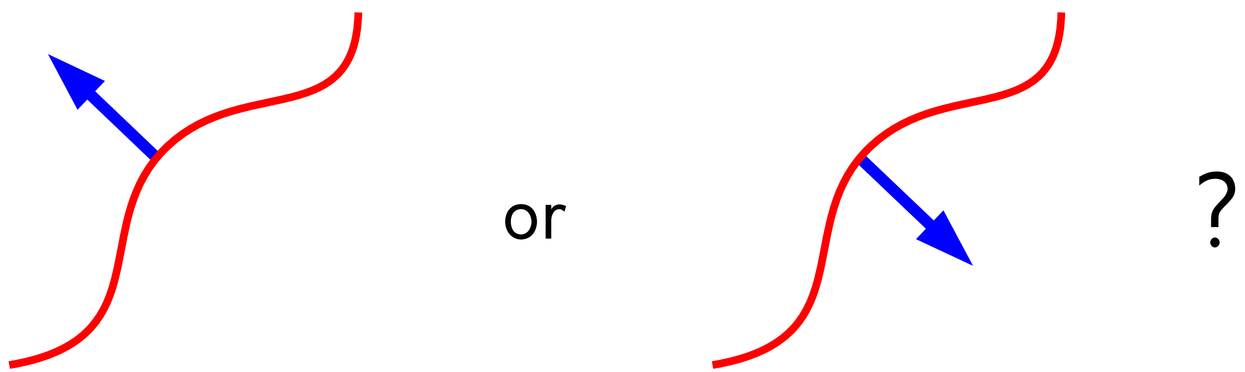
$$\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right)$$



(Oleg Alexandrov)

Caution!

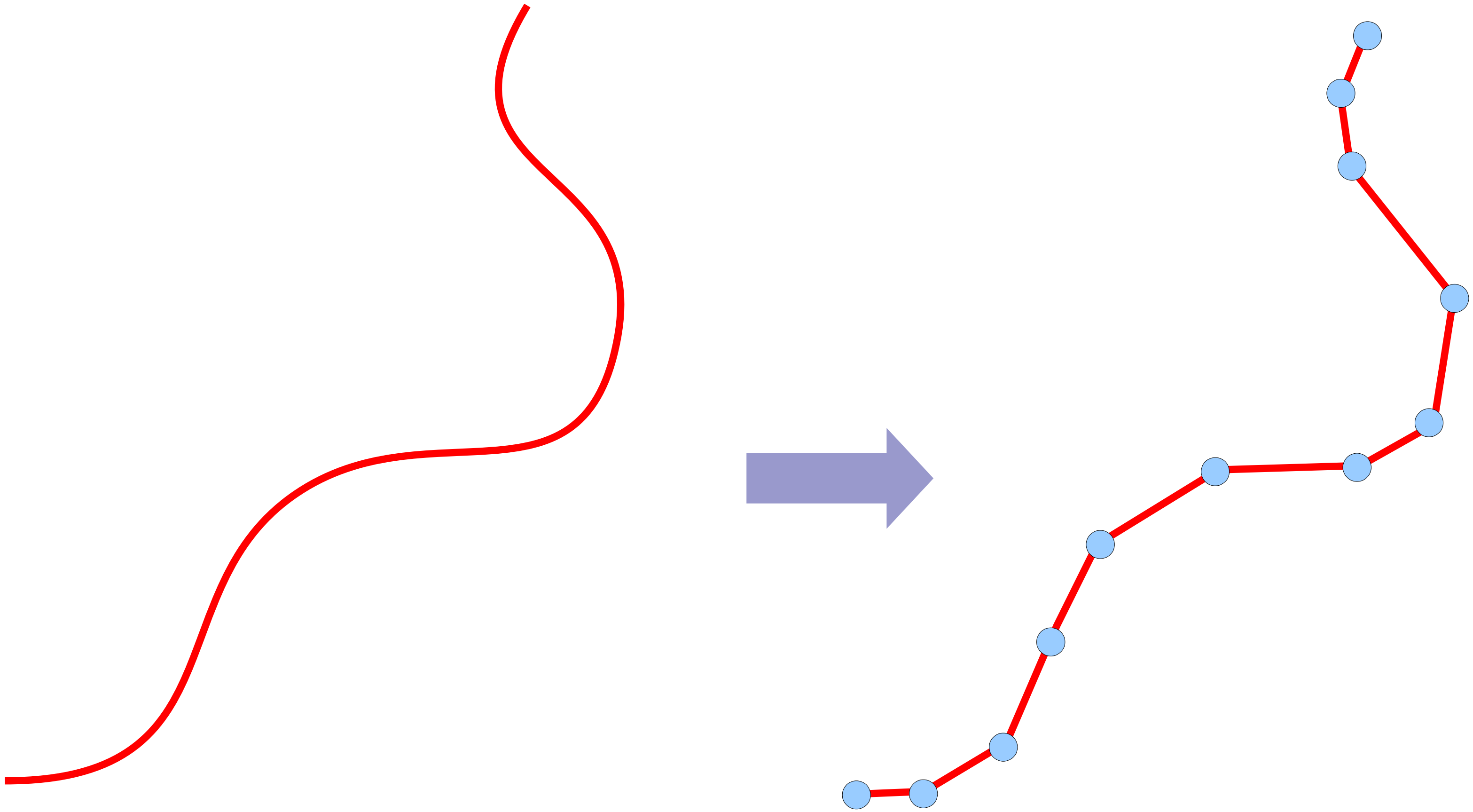
- Normals can point in **two opposing directions**
  - Choose a **consistent convention**
  - For closed surfaces we usually take the outward direction



- Many formulæ require **unit normals**
  - Divide by the length of the normal to **unitize**

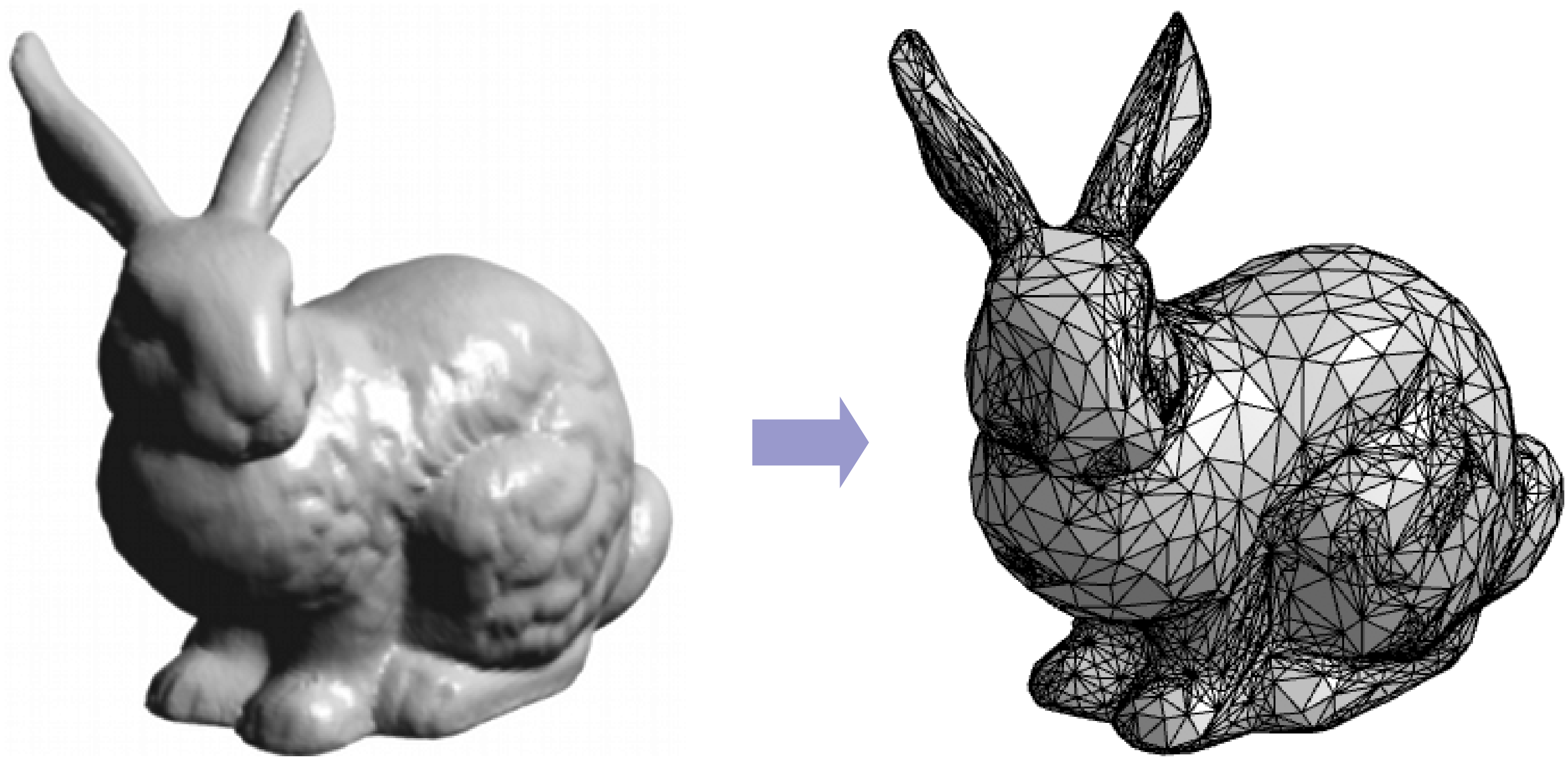
Piecewise Linear Approximation of Curve

- Straight lines are easier to process and display than curves!



Piecewise Linear Approximation of Surface

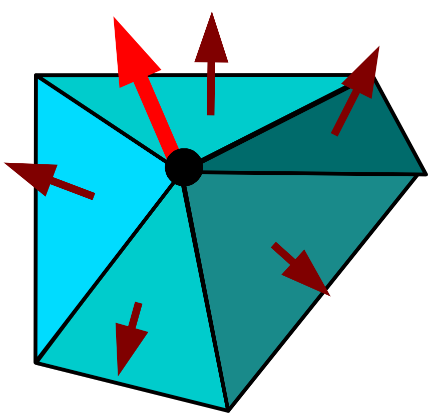
- Polygons are easier to process and display than curved surfaces!



Triangle Mesh

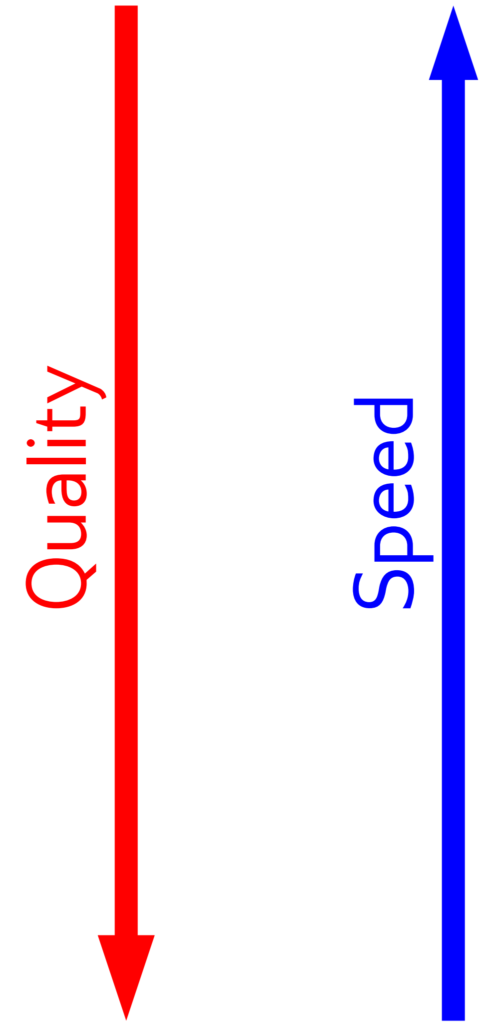
Polygon Meshes

- Set of edge-connected planar polygons (usually triangles or quads)
  - Faces share vertices and edges
  - To avoid repeating vertices, store each vertex once
  - Each face stored as set of indices into the vertex list
- Connectivity of faces also called *mesh topology*
- Normal at vertex often estimated as average of unit normals of all faces sharing that vertex
  - Useful in practice, but less precise than differentiating original surface

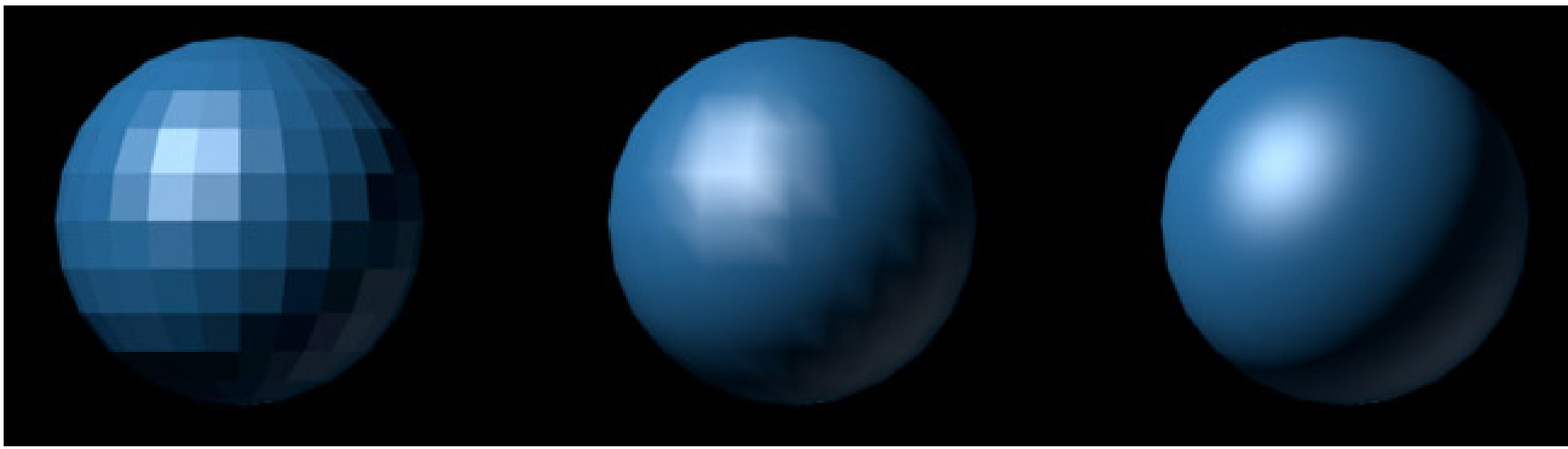


Displaying Polygon Meshes

- *Flat shading:* Compute shading at face center, use for entire face
- *Per-vertex (Gouraud) shading*: Compute shading at vertices, interpolate to face interiors
- *Per-fragment (Phong) shading:* Interpolate normals to face interiors, compute shading at each fragment
  - Don't confuse with Phong reflection model!



Displaying Polygon Meshes



Flat shading

Per-vertex (Gouraud) shading

Per-fragment (Phong) shading

(Camillo Trevisan)

Displaying Polygon Meshes

Flat shading | Per-vertex (Gouraud) shading | Per-fragment (Phong) shading

(Paul Heckbert)

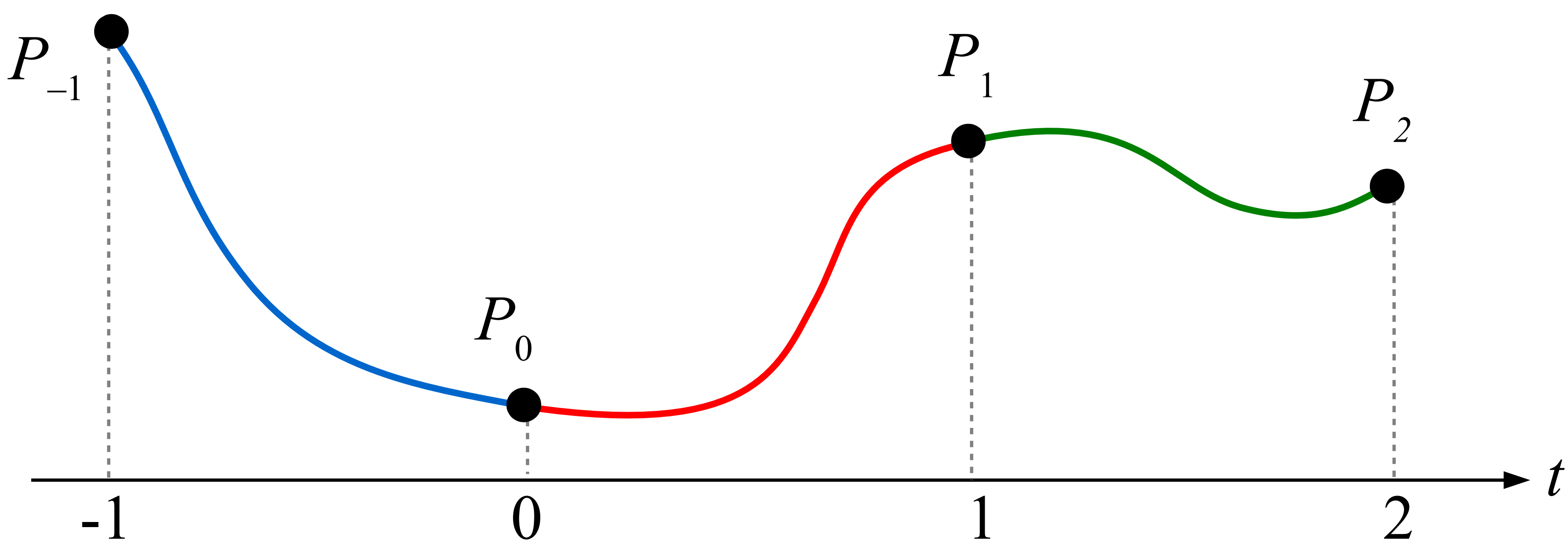
Controlling a Curve

- Specify control parameters at a few locations
  - Points
  - Tangents
  - ...
- Make the curve conform to these parameters

(Getty Images)

Interpolation with Splines

- **Want:** Smooth curve through sequence of points
- **Intuition:** Generate the curve in parts, one between each pair of points
  - This is called a *spline curve*
  - Has *local control* (small change won't affect whole curve)

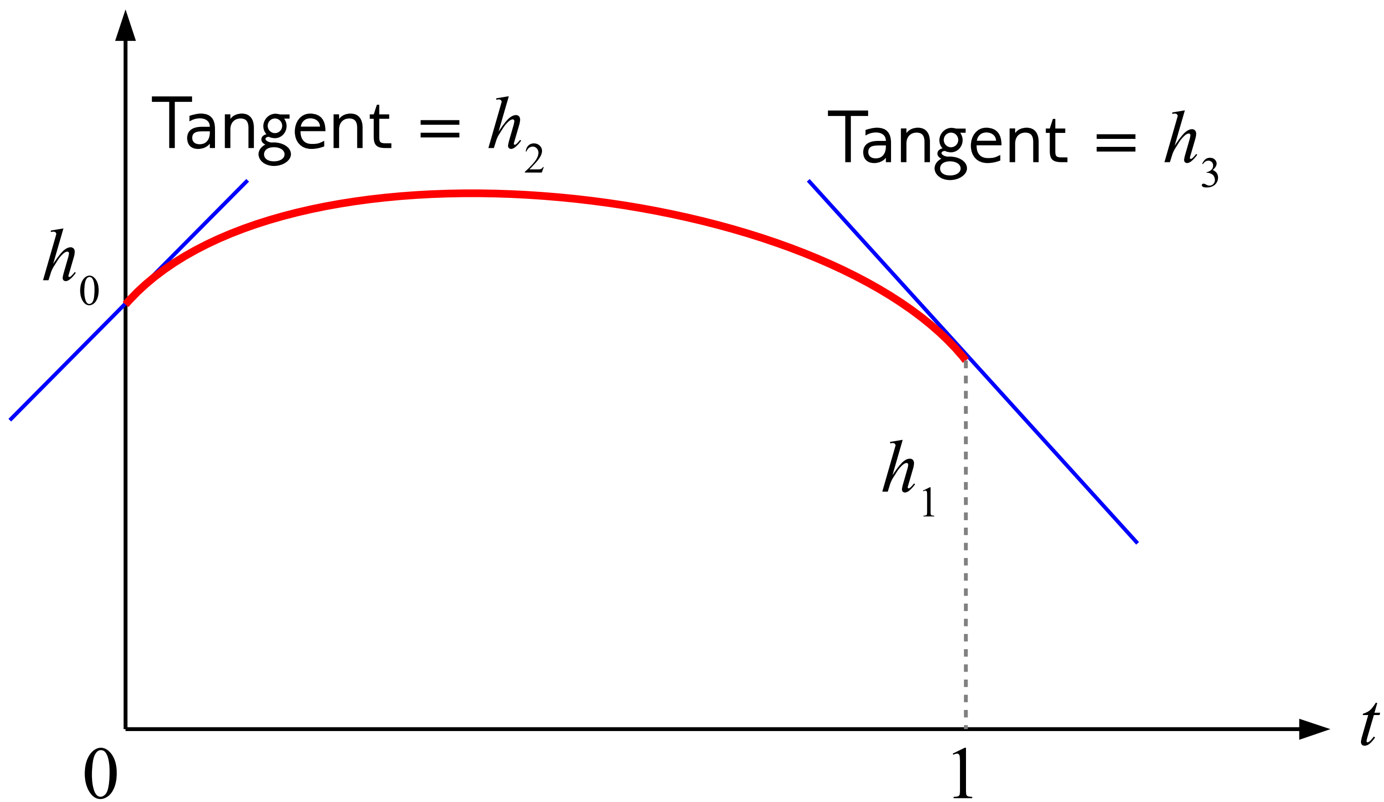


Cubic Curve

- $P(t) = at^3 + bt^2 + ct + d$
- 4 degrees of freedom
  - For instance, can be specified completely by 4 points on the curve
- Popular tradeoff between control and simplicity
- Multiple cubic segments can be linked together into a longer and more complex curve

Cubic Hermite Interpolation

- Specify positions h_0, h_1 and tangents (slopes, derivatives) h_2, h_3 at two points: $t = 0$ and $t = 1$



Cubic Hermite Interpolation

- **Q:** Why tangents and not two extra points?
- **A:** When we want two curve segments to link up smoothly, we can just require them to have a common tangent at the boundary

Cubic Hermite Interpolation

$$P(t) = at^3 + bt^2 + ct + d$$

$$P'(t) = 3at^2 + 2bt + c$$

$$\begin{aligned}
h_0 &= P(0) = d \\
h_1 &= P(1) = a + b + c + d \\
h_2 &= P'(0) = c \\
h_3 &= P'(1) = 3a + 2b + c
\end{aligned}$$

Matrix Representation

$$
\begin{aligned}
h_0 &= d \\
h_1 &= a + b + c + d \\
h_2 &= c \\
h_3 &= 3a + 2b + c
\end{aligned}
$$

Hermite constraint matrix

$$
\underbrace{\begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{bmatrix}}_{\mathbf{h}} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}}_{\mathrm{C}} \underbrace{\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}}_{\mathbf{a}}
$$

Matrix Representation

$$\mathbf{h} = \mathrm{C}\mathbf{a} \;\Rightarrow\; \mathbf{a} = \mathrm{C}^{-1}\mathbf{h}$$

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \underbrace{\begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}}_{\textit{Hermite basis matrix}} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{bmatrix}$$

Matrix Representation of Polynomials

$$P(t) = \begin{bmatrix} a & b & c & d \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

Matrix Representation of Polynomials

$$P(t) = \begin{bmatrix} h_0 & h_1 & h_2 & h_3 \end{bmatrix} \underbrace{\begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}}_{(C^{-1})^T} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}$$

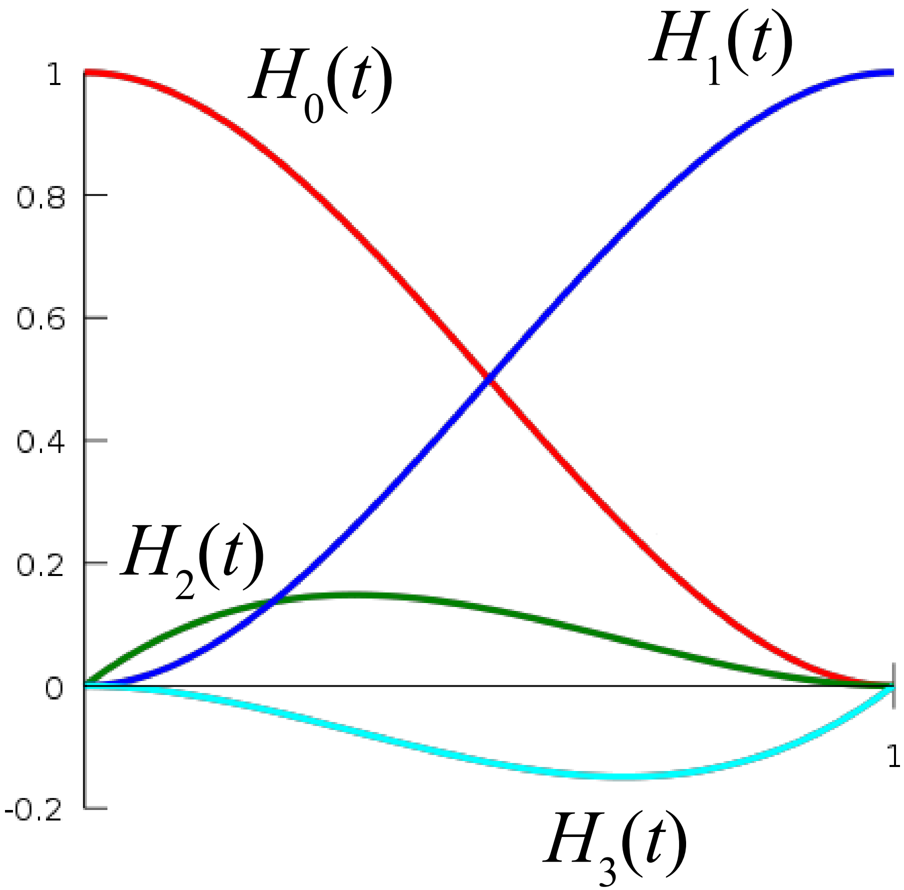
Matrix Representation of Polynomials

$$P(t) = \begin{bmatrix} h_0 & h_1 & h_2 & h_3 \end{bmatrix} \underbrace{\begin{bmatrix} H_0(t) \\ H_1(t) \\ H_2(t) \\ H_3(t) \end{bmatrix}}_{\textit{Hermite basis functions}}$$

$$P(t) = \sum_{i=0}^{3} h_i H_i(t)$$

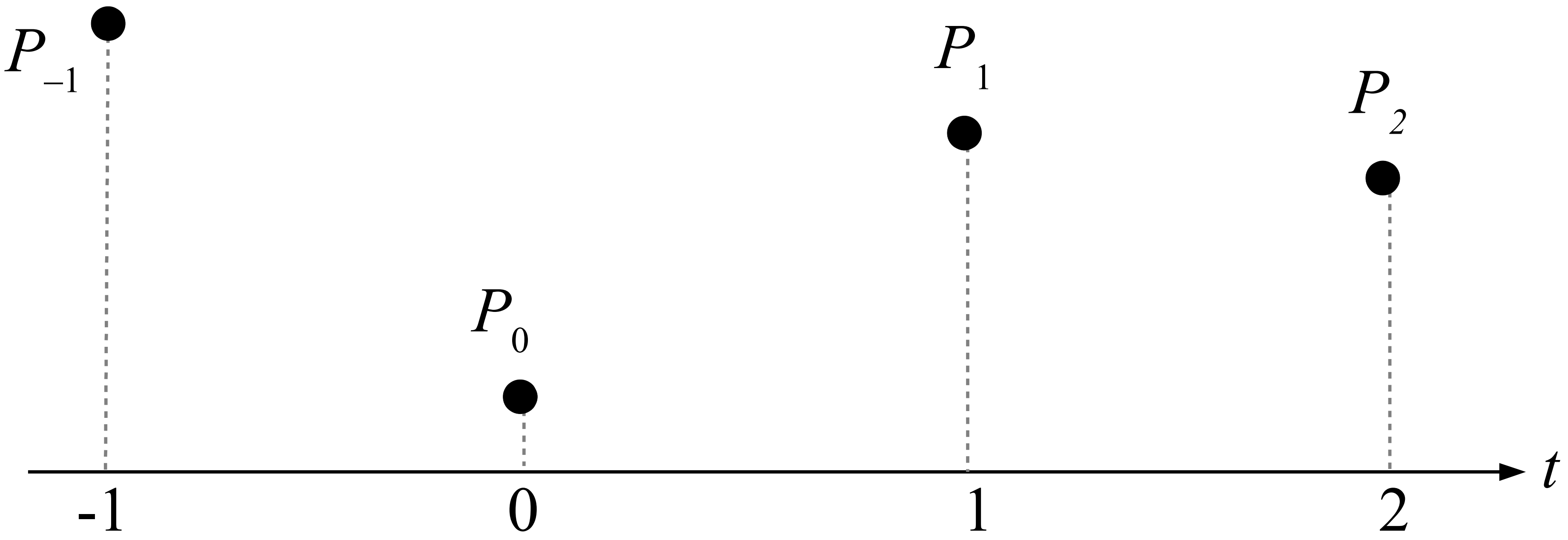
Hermite Basis Functions

$$H_0(t) = 2t^3 - 3t^2 + 1$$
$$H_1(t) = -2t^3 + 3t^2$$
$$H_2(t) = t^3 - 2t^2 + t$$
$$H_3(t) = t^3 - t^2$$



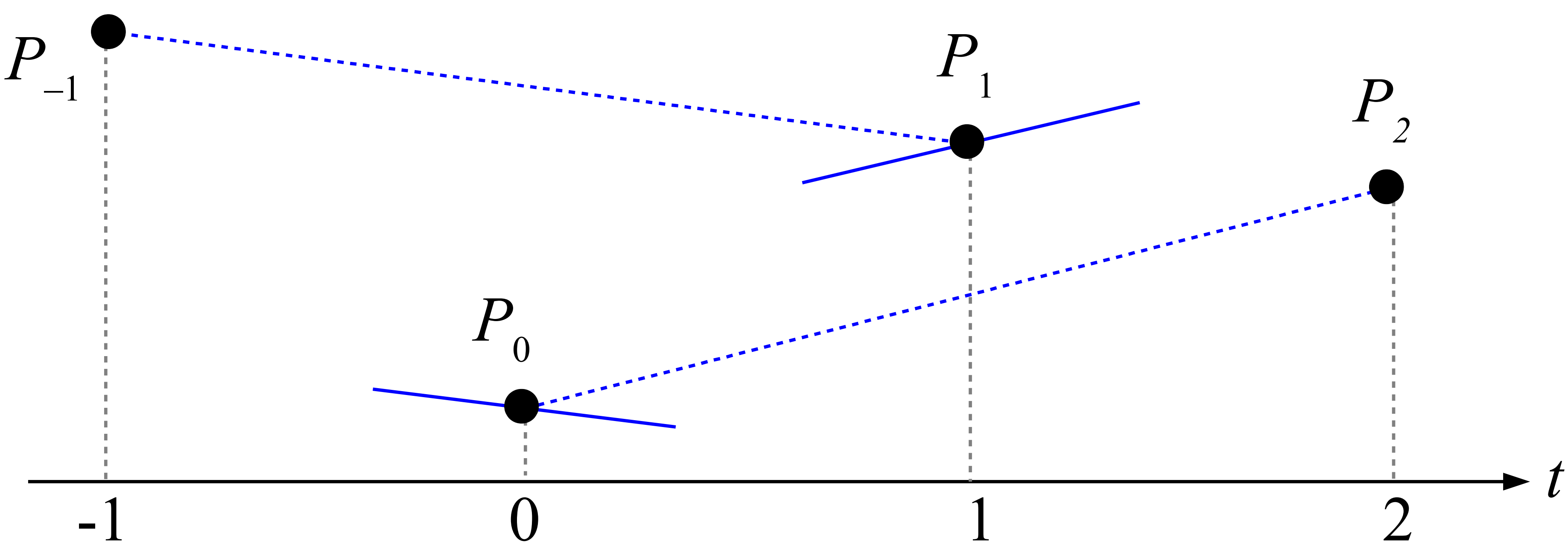
Catmull-Rom Interpolation

- **Want:** Smooth curve through sequence of points
- **Intuition:** A plausible tangent at each point can be inferred directly from the data
  - Now use Hermite interpolation



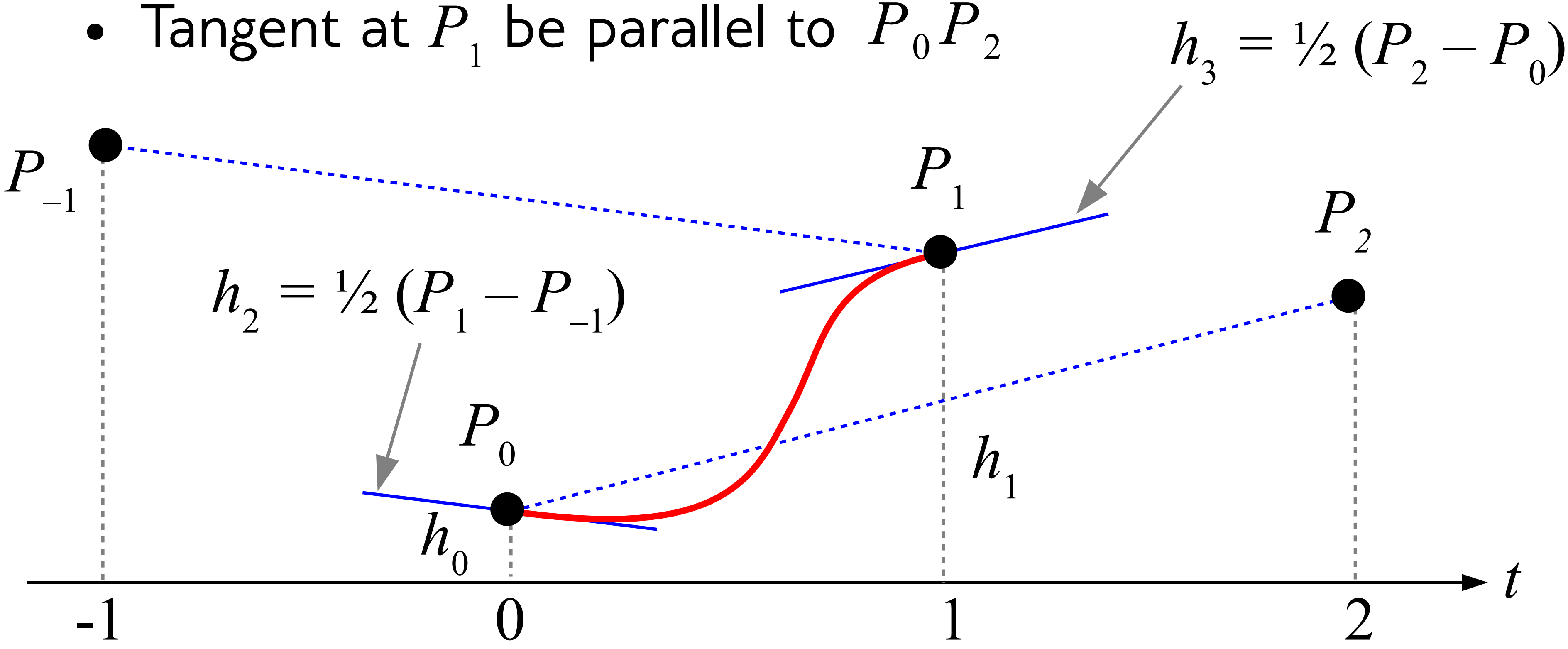
Catmull-Rom Interpolation

- For each segment (P_0, P_1), use neighboring control points P_{-1}, P_2 and require that:
  - Tangent at P_0 be parallel to $\overline{P_{-1}P_1}$
  - Tangent at P_1 be parallel to $\overline{P_0 P_2}$



Catmull-Rom Interpolation

- For each segment (P_0, P_1), use neighboring control points P_{-1}, P_2 and require that:
  - Tangent at P_0 be parallel to $\overline{P_{-1}P_1}$
  - Tangent at P_1 be parallel to $\overline{P_0P_2}$



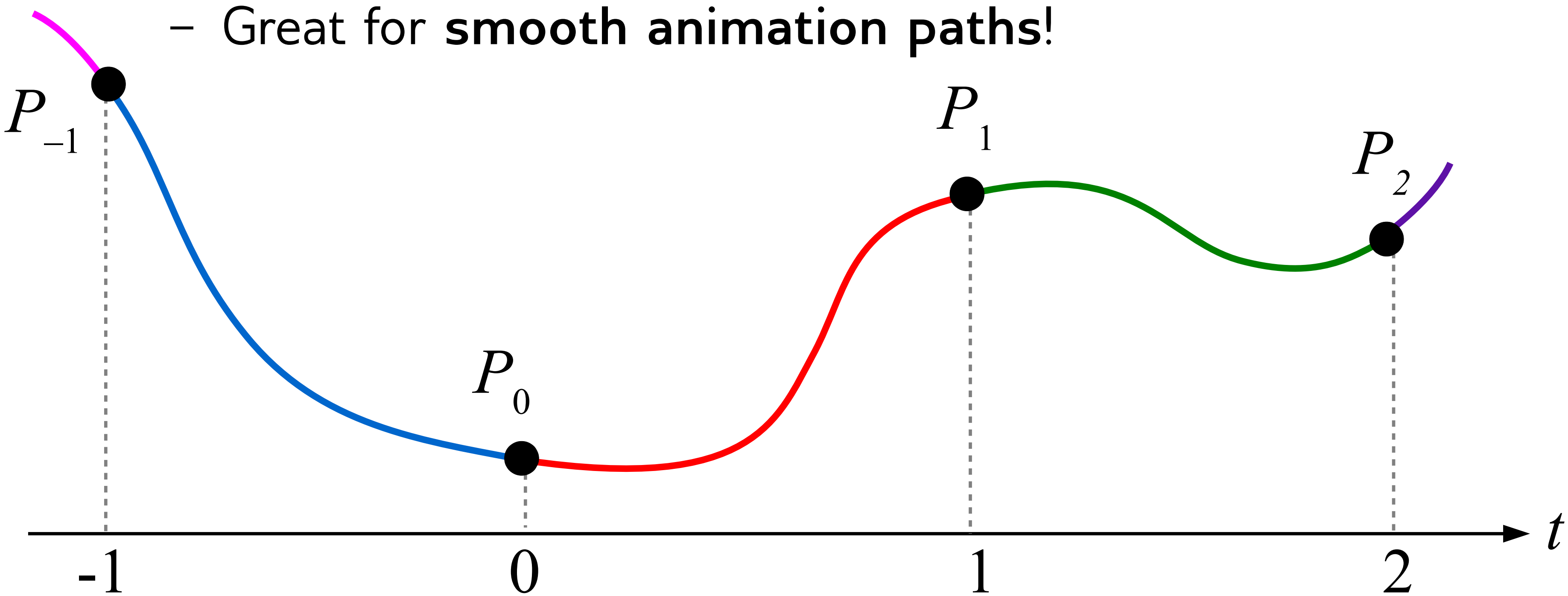
Catmull-Rom Interpolation

- In terms of Hermite constraints:

$$h_0 = P_0$$
$$h_1 = P_1$$
$$h_2 = ½\,(P_1 - P_{-1})$$
$$h_3 = ½\,(P_2 - P_0)$$

Catmull-Rom Interpolation

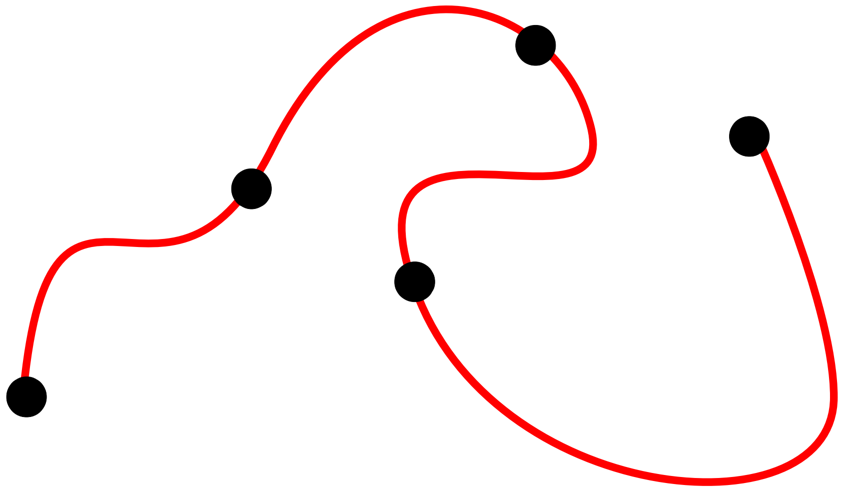
- Repeat for every such interval
- Resulting curve is:
  - *C_0-continuous* (segments meet end-to-end)
  - *C_1-continuous* (C_0 + derivative is continuous)
    - Great for **smooth animation paths**!



Curves in 2D/3D/...

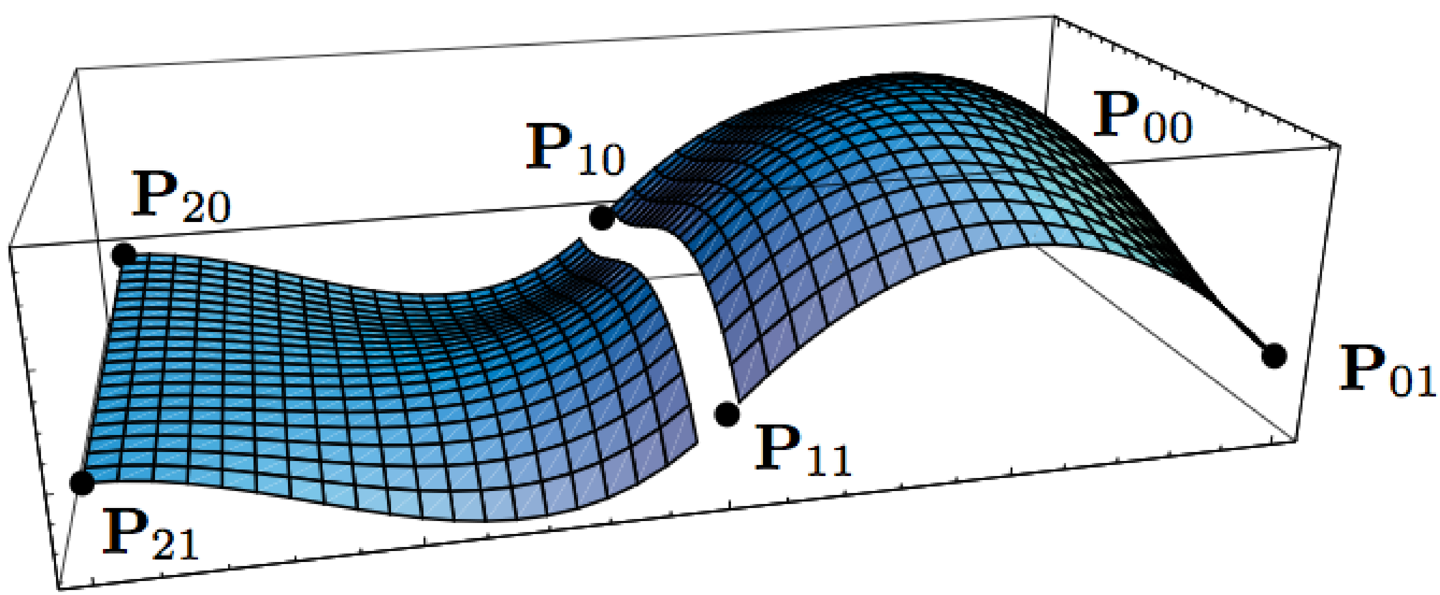
- Control points/tangents can be any-dimensional
  - One way to look at it: treat each coordinate separately, so we have different $[a, b, c, d]$ for each dimension
  - Another way: the constraints and cœfficients are now vectors, not scalars
  - t is "distance" along curve from one point to the next

$$\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} = C^{-1} \begin{bmatrix} \mathbf{h}_0 \\ \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{bmatrix}$$



Curved Surfaces as Spline Patches

- Grid of control points (control polyhedron)
- Surface indexed by $(s, t) \in \mathbb{R}^2$
- Basis functions are pairwise products of 1D (curve) basis functions



Two bicubic patches joined smoothly

(Salomon 2006)