Course Updates

- Assignment **3 Programming** is now in
- Assignment **3 Theory** is due today

- Assignment **4 Programming** out (due Monday)

- Assignment **2 Programming** Grading
  - Texture map is lit by either diffuse or specular but not both (**-2**/**-0**)
  - Normals are shown in only diffuse mode (**-2**/**-1**)

Interpolation, Parametric Curves and Surfaces

Computer Graphics, CSCD18

Fall 2007

Instructor: Leonid Sigal

Interpolation Basics

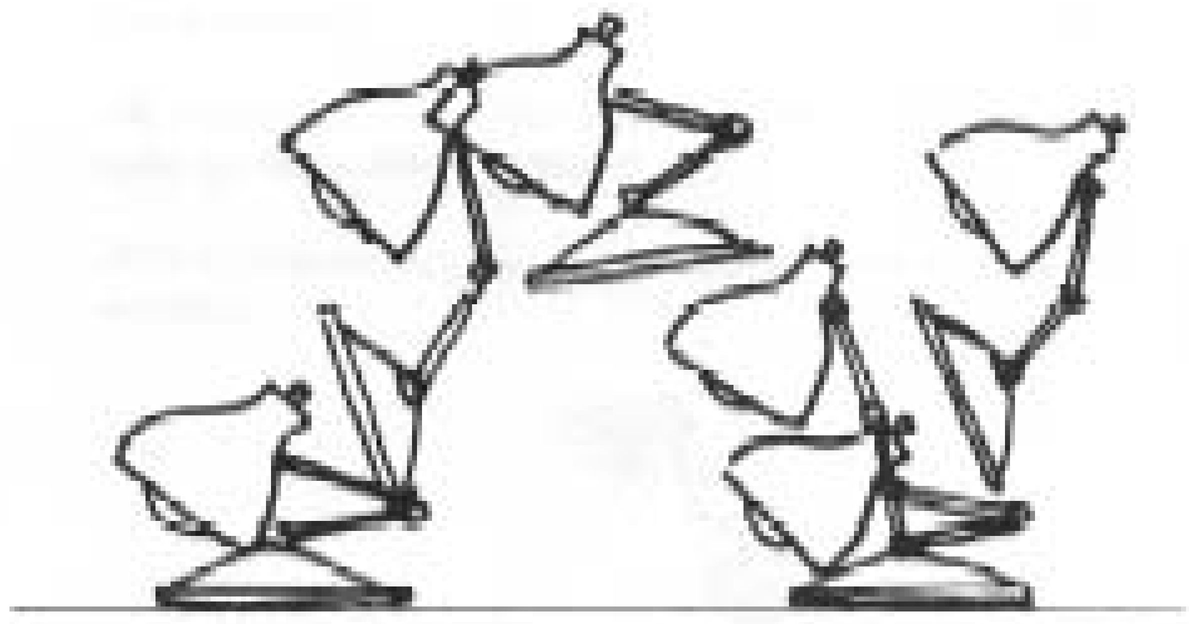
- We would like to define curves that meet the following criteria:
  - Interaction should be natural and intuitive
  - Smoothness should be controllable
  - Analytic derivatives should exist and be easy to compute
  - Adjustable resolution (easy to zoom in and out)
  - Representation should be compact

- Why do we need these curves
  - Animation
  - Curved surfaces

How can you animate something like this?

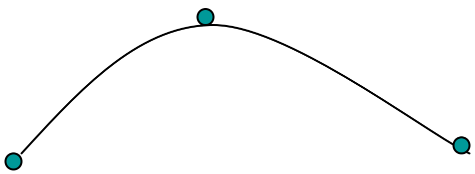
Keyframe Animation

- **Idea:** specify variables that describe **keyframes** and interpolate them over the sequence

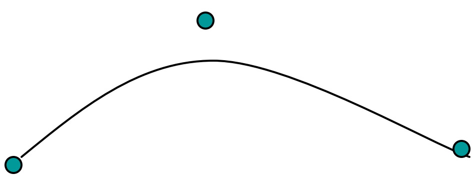


Curves Basics

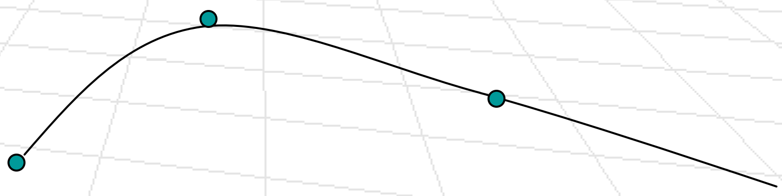
- **Interpolation**
  - Curve goes through "control points"



- **Approximation**
  - Curve approximates but does not go through "control points"

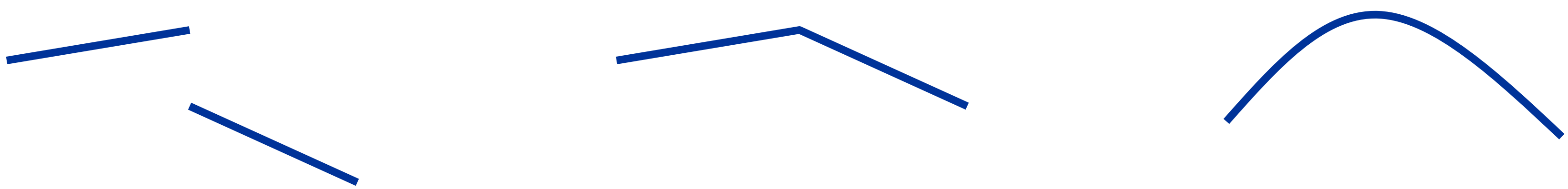


- **Extrapolation**
  - Extending curve beyond domain of control points



Continuity

- C^n continuous function implies that n-th order derivatives exist

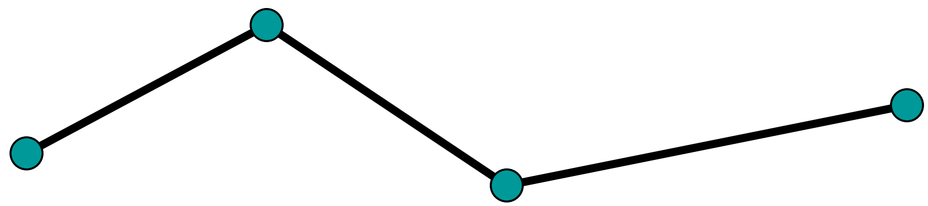


C^0 C^1 C^2

For animation purposes, C^2 continuous functions typically are sufficient

Linear Interpolation

- Simplest possible interpolation technique
  - Peace wise linear curve



- Pros:
  - Really simple to implement
  - Local (interpolation only depends on the closest two control points)
- Cons:
  - Only C^1 continuous (typically bad for animation)

Cubic Interpolation

- Consider a 2D cubic interplant (a curve in 2D)

$$\boldsymbol{c}(\boldsymbol{t}) = [\boldsymbol{x}(\boldsymbol{t})\ \boldsymbol{y}(\boldsymbol{t})]$$

where

$$\boldsymbol{x}(\boldsymbol{t}) = \boldsymbol{a}_0 + \boldsymbol{a}_1\boldsymbol{t} + \boldsymbol{a}_2\boldsymbol{t}^2 + \boldsymbol{a}_3\boldsymbol{t}^3$$

$$\boldsymbol{y}(\boldsymbol{t}) = \boldsymbol{b}_0 + \boldsymbol{b}_1\boldsymbol{t} + \boldsymbol{b}_2\boldsymbol{t}^2 + \boldsymbol{b}_3\boldsymbol{t}^3$$

Cubic Interpolation

- Consider a 2D cubic interplant (a curve in 2D)

$$c(t) = [x(t)\ y(t)]$$

where
$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$
$$y(t) = b_0 + b_1 t + b_2 t^2 + b_3 t^3$$

Alternatively,

$$c(t) = \underbrace{\begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix}}_{\text{bases}} \underbrace{\begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}}_{\text{coefficients}}$$

Cubic Interpolation

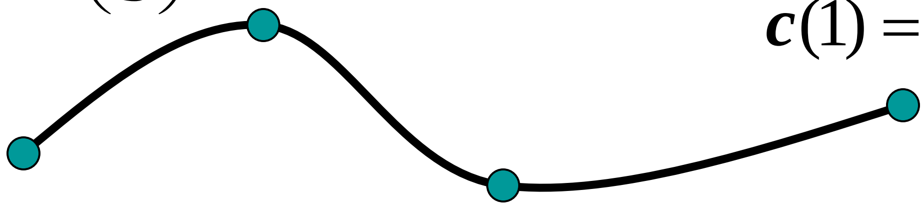
- We have 8 unknowns (coefficients) how many 2D points do we need to constrain the curve?

$$\boldsymbol{c}(\boldsymbol{t}) = \underbrace{\begin{bmatrix} 1 & \boldsymbol{t} & \boldsymbol{t}^2 & \boldsymbol{t}^3 \end{bmatrix}}_{\text{bases}} \left.\begin{bmatrix} \boldsymbol{a}_0 & \boldsymbol{b}_0 \\ \boldsymbol{a}_1 & \boldsymbol{b}_1 \\ \boldsymbol{a}_2 & \boldsymbol{b}_2 \\ \boldsymbol{a}_3 & \boldsymbol{b}_3 \end{bmatrix}\right\} \text{coefficients}$$

Cubic Interpolation

$$c\left(\frac{1}{3}\right) = [x_2, y_2]$$

$$c(1) = [x_4, y_4]$$



$$c(0) = [x_1, y_1]$$

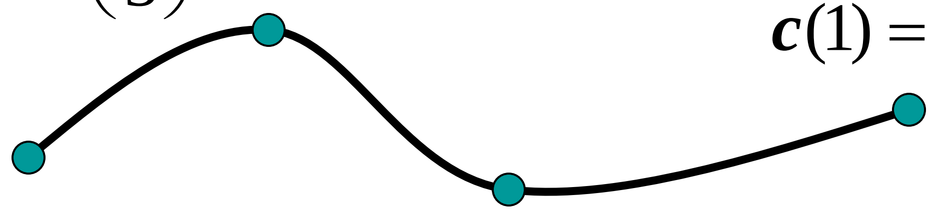
$$c\left(\frac{2}{3}\right) = [x_3, y_3]$$

$$c(t) = \underbrace{[1 \quad t \quad t^2 \quad t^3]}_{\textbf{bases}} \left.\begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}\right\} \textbf{coefficients}$$

Cubic Interpolation

$$c\left(\frac{1}{3}\right) = [x_2, y_2]$$

$$c(1) = [x_4, y_4]$$



$$c(0) = [x_1, y_1]$$

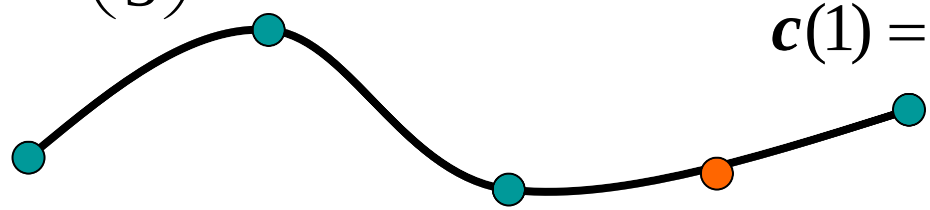
$$c\left(\frac{2}{3}\right) = [x_3, y_3]$$

$$\begin{bmatrix} x_0 & y_0 \\ x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & \left(\frac{1}{3}\right) & \left(\frac{1}{3}\right)^2 & \left(\frac{1}{3}\right)^3 \\ 1 & \left(\frac{2}{3}\right) & \left(\frac{2}{3}\right)^2 & \left(\frac{2}{3}\right)^3 \\ 1 & 1 & 1 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}}_{\text{coefficients}}$$

Cubic Interpolation

$$c\left(\frac{1}{3}\right) = [x_2, y_2]$$

$$c(1) = [x_4, y_4]$$



$$c(0) = [x_1, y_1]$$

$$c\left(\frac{2}{3}\right) = [x_3, y_3]$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & \left(\frac{1}{3}\right) & \left(\frac{1}{3}\right)^2 & \left(\frac{1}{3}\right)^3 \\ 1 & \left(\frac{2}{3}\right) & \left(\frac{2}{3}\right)^2 & \left(\frac{2}{3}\right)^3 \\ 1 & 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_0 & y_0 \\ x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \end{bmatrix} = \left.\boxed{\begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix}}\right\} \text{coefficients}$$

Cubic Interpolation

- Consider a 2D cubic interplant (a curve in 2D)

$$\boldsymbol{c}(\boldsymbol{t}) = [\boldsymbol{x}(\boldsymbol{t})\ \boldsymbol{y}(\boldsymbol{t})]$$

where
$$\boldsymbol{x}(\boldsymbol{t}) = \boldsymbol{a}_0 + \boldsymbol{a}_1\boldsymbol{t} + \boldsymbol{a}_2\boldsymbol{t}^2 + \boldsymbol{a}_3\boldsymbol{t}^3$$
$$\boldsymbol{y}(\boldsymbol{t}) = \boldsymbol{b}_0 + \boldsymbol{b}_1\boldsymbol{t} + \boldsymbol{b}_2\boldsymbol{t}^2 + \boldsymbol{b}_3\boldsymbol{t}^3$$

Alternatively we can place derivative constrains

$$\tau(\boldsymbol{t}) = \frac{\boldsymbol{d}\ \boldsymbol{c}(\boldsymbol{t})}{\boldsymbol{dt}} = \underbrace{\frac{\boldsymbol{d}}{\boldsymbol{dt}}[1 \quad \boldsymbol{t} \quad \boldsymbol{t}^2 \quad \boldsymbol{t}^3]}_{\text{bases}} \underbrace{\begin{bmatrix} \boldsymbol{a}_0 & \boldsymbol{b}_0 \\ \boldsymbol{a}_1 & \boldsymbol{b}_1 \\ \boldsymbol{a}_2 & \boldsymbol{b}_2 \\ \boldsymbol{a}_3 & \boldsymbol{b}_3 \end{bmatrix}}_{\text{coefficients}}$$

Cubic Interpolation

- Consider a 2D cubic interplant (a curve in 2D)

$$\boldsymbol{c}(\boldsymbol{t}) = [\boldsymbol{x}(\boldsymbol{t})\ \boldsymbol{y}(\boldsymbol{t})]$$

where

$$\boldsymbol{x}(\boldsymbol{t}) = \boldsymbol{a}_0 + \boldsymbol{a}_1\boldsymbol{t} + \boldsymbol{a}_2\boldsymbol{t}^2 + \boldsymbol{a}_3\boldsymbol{t}^3$$

$$\boldsymbol{y}(\boldsymbol{t}) = \boldsymbol{b}_0 + \boldsymbol{b}_1\boldsymbol{t} + \boldsymbol{b}_2\boldsymbol{t}^2 + \boldsymbol{b}_3\boldsymbol{t}^3$$

Alternatively we can place derivative constrains

$$\tau(\boldsymbol{t}) = \frac{\boldsymbol{d}\ \boldsymbol{c}(\boldsymbol{t})}{\boldsymbol{dt}} = \underbrace{\begin{bmatrix} 0 & 1 & 2\boldsymbol{t} & 3\boldsymbol{t}^2 \end{bmatrix}}_{\text{different bases}} \left.\begin{bmatrix} \boldsymbol{a}_0 & \boldsymbol{b}_0 \\ \boldsymbol{a}_1 & \boldsymbol{b}_1 \\ \boldsymbol{a}_2 & \boldsymbol{b}_2 \\ \boldsymbol{a}_3 & \boldsymbol{b}_3 \end{bmatrix}\right\}\text{same coefficients}$$

Cubic Interpolation

- What happens if there are more then 4 points?
  - There may not be a solution that goes through all the control points (or any of the control points)
  - Interpolation may not result in intuitive results

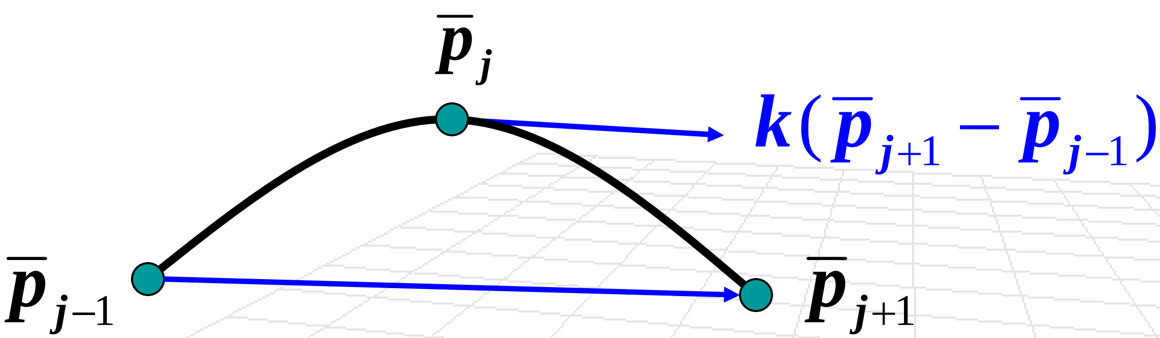
- Cubic interpolation is global
  - Changing one control point changes the interpolation for all points

- In general (at least for animation) local control is better

Catmull-Rom Splines

- **Idea:** piecewise cubic curves of degree-3 with C^1 continuity

- A user specifies points and the tangent at each point is set to be parallel to the vector between adjacent points



- ***k*** is the st by the user parameter, that determines the “tension” of the curve

Catmull-Rom Splines

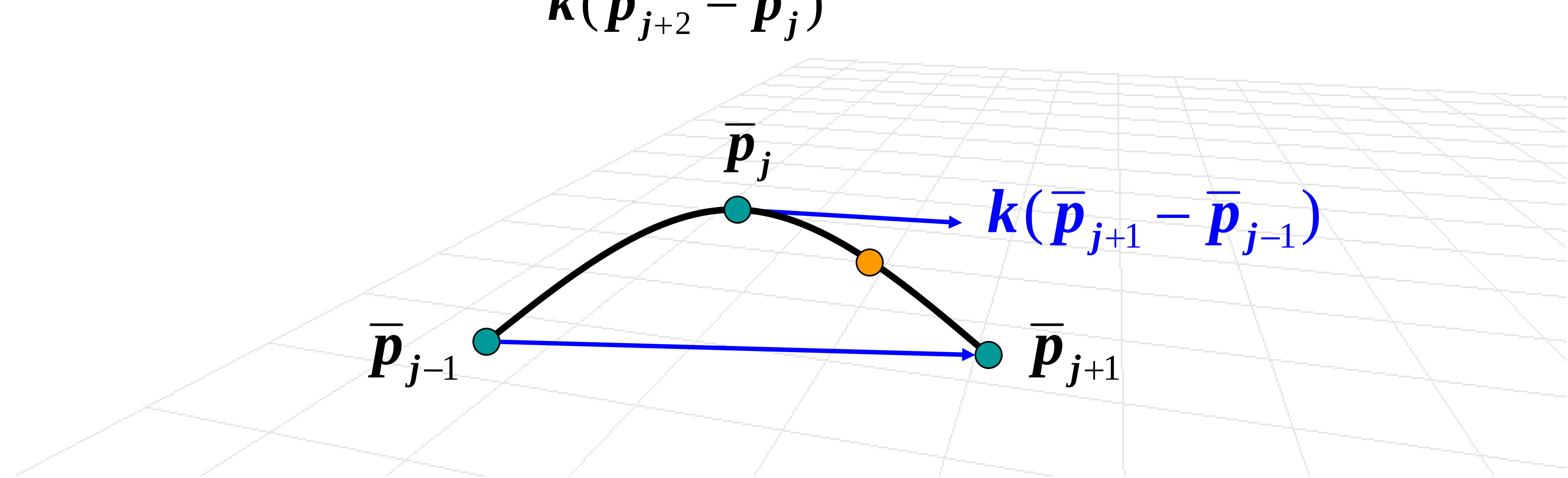
- To interpolate a value for the point between $\boldsymbol{p_j}$ and $\boldsymbol{p_{j+1}}$ one needs to consider 4 bits of information

$$\bar{\boldsymbol{p}}_j$$

$$\bar{\boldsymbol{p}}_{j+1}$$

$$\boldsymbol{k}(\bar{\boldsymbol{p}}_{j+1} - \bar{\boldsymbol{p}}_{j-1})$$

$$\boldsymbol{k}(\bar{\boldsymbol{p}}_{j+2} - \bar{\boldsymbol{p}}_{j})$$



Catmull-Rom Splines

- To interpolate a value for the point between $\boldsymbol{p}_j$ and $\boldsymbol{p}_{j+1}$ one needs to consider 4 bits of information

$$\bar{p}_j$$

$$\bar{p}_{j+1}$$

$$k(\bar{p}_{j+1} - \bar{p}_{j-1})$$

$$k(\bar{p}_{j+2} - \bar{p}_j)$$

4 points lead to cubic interplant (see lecture notes for details)