CS559: Computer Graphics

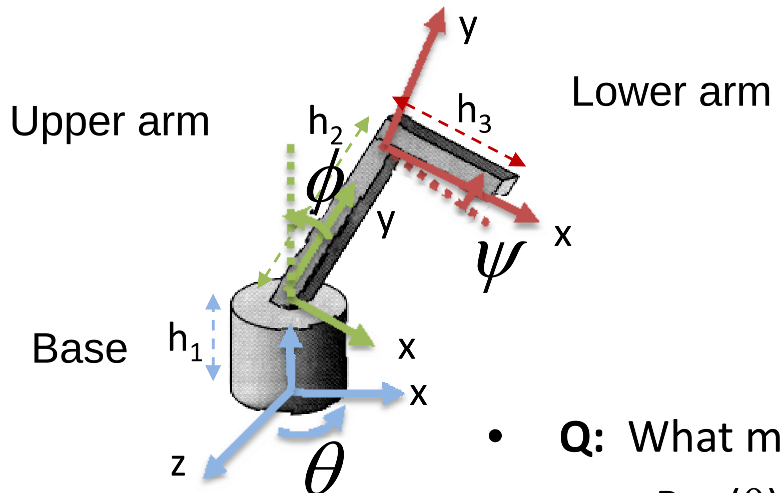
Lecture 13: Hierarchical Modeling and Curves

Li Zhang

Spring 2010

Last time: 3D Example: A robot arm

- Consider this robot arm with 3 degrees of freedom:
  - Base rotates about its vertical axis by θ
  - Upper arm rotates in its *xy*-plane by ϕ
  - Lower arm rotates in its *xy*-plane by ψ



- **Q:** What matrix do we use to transform the base to the world?
  - R_y(θ)
- **Q:** What matrix for the upper arm to the base?
  - T(0,h1,0)R_z(ϕ)
- **Q:** What matrix for the lower arm to the upper arm?
  - T(0,h2,0)R_z(ψ)

Robot arm implementation

- The robot arm can be displayed by keeping a global matrix and computing it at each step:

```
Matrix M_model;
display(){
    . . .
    robot_arm();
    . . .
}
robot_arm()
{
    M_model = R_y(theta);
    base();
    M_model = R_y(theta)*T(0,h1,0)*R_z(phi);
    upper_arm();
    M_model = R_y(theta)*T(0,h1,0)*R_z(phi)*T(0,h2,0)*R_z(psi);
    lower_arm();
}
```

- **Q:** What matrix do we use to transform the base to the world?
  - R_y(θ)
- **Q:** What matrix for the upper arm to the base?
  - T(0,h1,0)R_z(ϕ)
- **Q:** What matrix for the lower arm to the upper arm?
  - T(0,h2,0)R_z(ψ)

How to translate the whole robot?

Do the matrix computations seem wasteful?

Robot arm implementation, better

- Instead of recalculating the global matrix each time, we can just update it *in place* by concatenating matrices on the right:

```
Matrix M_model;
display(){
    . . .
    M_model = identity;
    robot_arm();
    . . .
}
robot_arm()
{
    M_model *= R_y(theta);
    base();
    M_model *= T(0,h1,0)*R_z(phi);
    upper_arm();
    M_model *= T(0,h2,0)*R_z(psi);
    lower_arm();
}
```

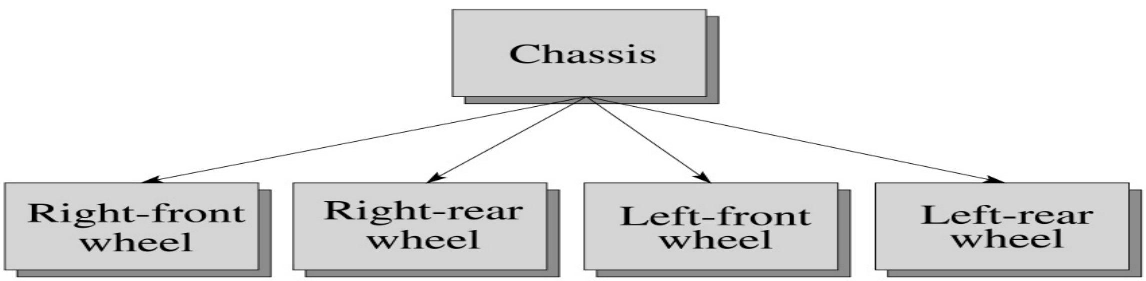
Robot arm implementation, OpenGL

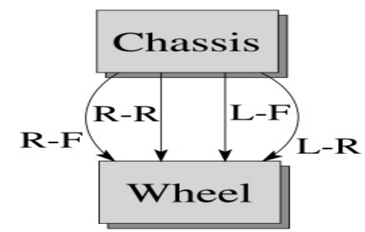
- OpenGL maintains the **model-view matrix**, as a global state variable which is updated by concatenating matrices on the ***right***.

```
display()
{
    . . .
    glMatrixMode( GL_MODELVIEW );
    glLoadIdentity();
    robot_arm();
    . . .
}
robot_arm()
{
    glRotatef( theta, 0.0, 1.0, 0.0 );
    base();
    glTranslatef( 0.0, h1, 0.0 );
    glRotatef( phi, 0.0, 0.0, 1.0 );
    lower_arm();
    glTranslatef( 0.0, h2, 0.0 );
    glRotatef( psi, 0.0, 0.0, 1.0 );
    upper_arm();
}
```

Hierarchical modeling

- Hierarchical models can be composed of instances using trees:

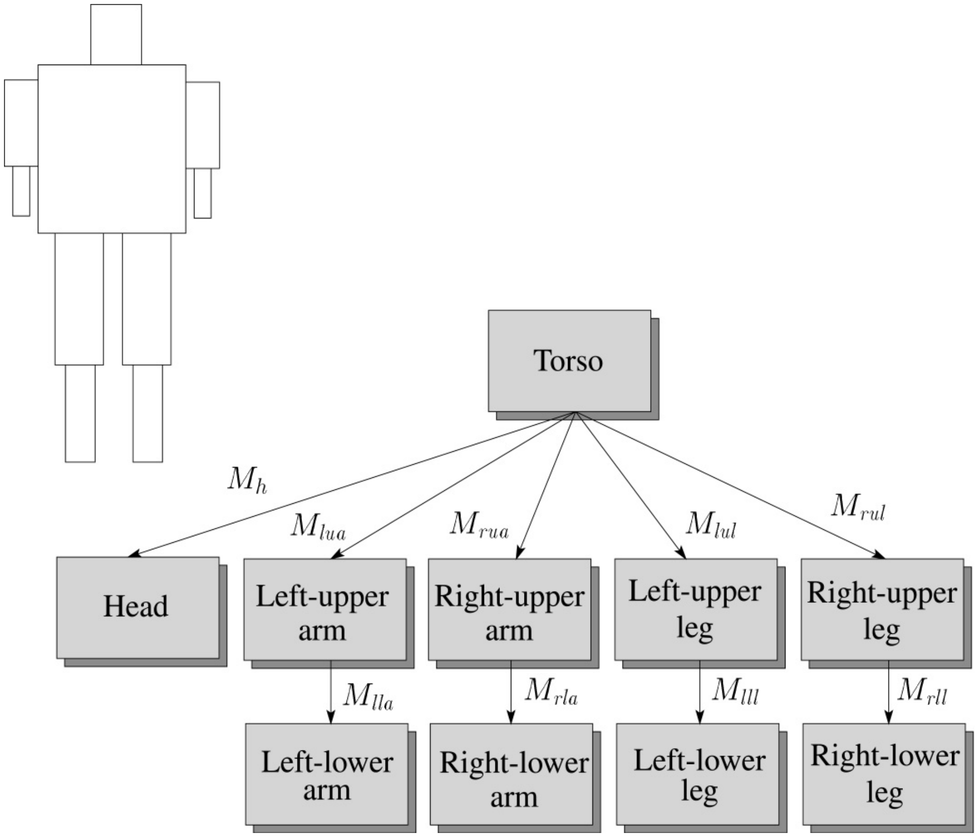




  – edges contain geometric transformations
  – nodes contain geometry (and possibly drawing attributes)

How might we draw the tree for the car?

A complex example: human figure

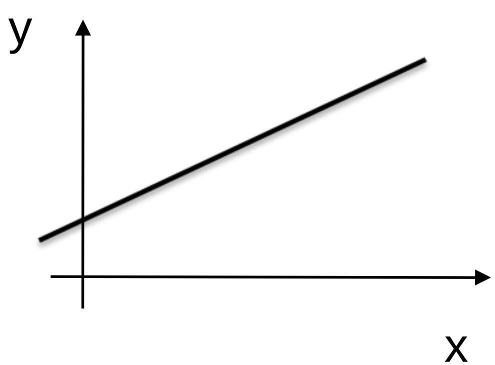


- **Q:** What’s the most sensible way to traverse this tree?

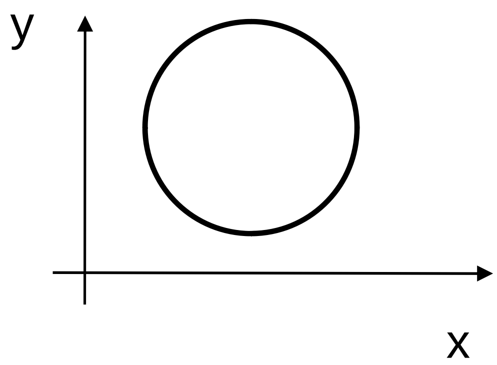
Human figure implementation, OpenGL

```
figure()
{
    torso();
    glPushMatrix();
        glTranslate( ... );
        glRotate( ... );
        head();
    glPopMatrix();
    glPushMatrix();
        glTranslate( ... );
        glRotate( ... );
        left_upper_arm();
        glPushMatrix();
            glTranslate( ... );
            glRotate( ... );
            left_lower_arm();
        glPopMatrix();
     glPopMatrix();
     . . .
}
```

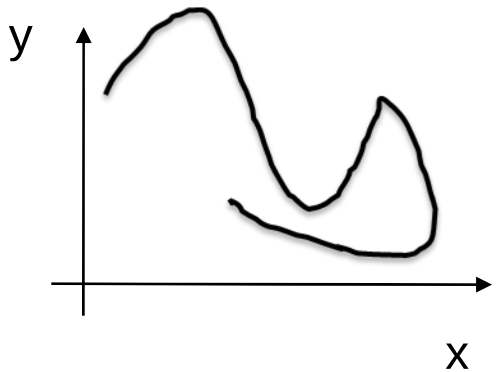
Curves



line

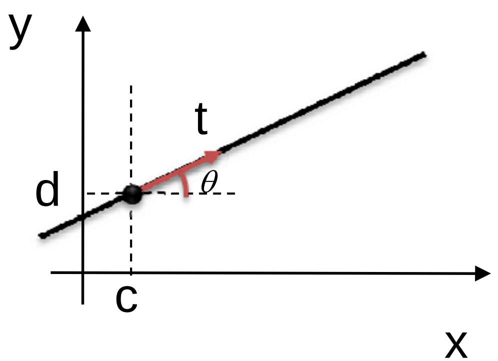


circle

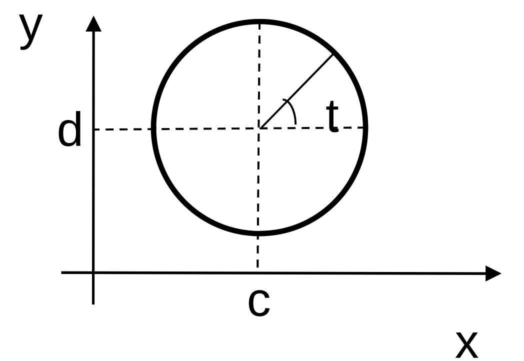


Freeform

Curve Representation



line



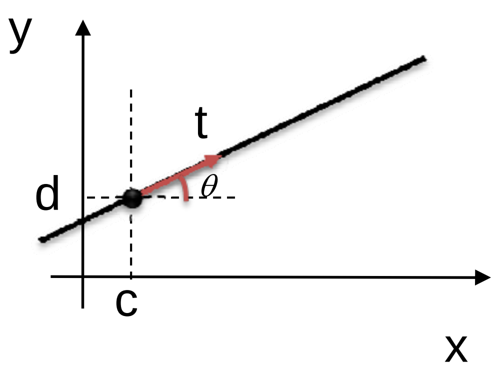
circle

Explicit: $$y = ax + b$$

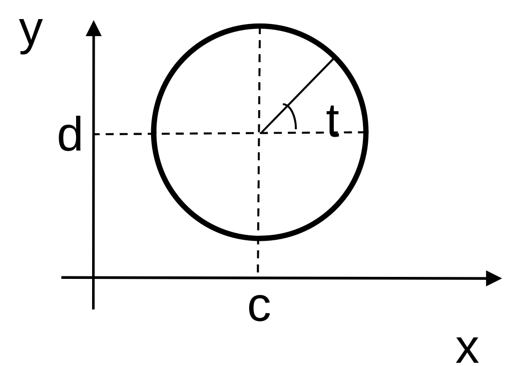
Implicit: $$f(x, y) = ax - y + b = 0$$ $$f(x, y) = (x - c)^2 + (y - d)^2 - r^2 = 0$$

Parametric: $$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} c \\ d \end{bmatrix} + t \cdot \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$$ $$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} c \\ d \end{bmatrix} + \begin{bmatrix} r \cdot \cos t \\ r \cdot \sin t \end{bmatrix}$$

Curve Representation



line

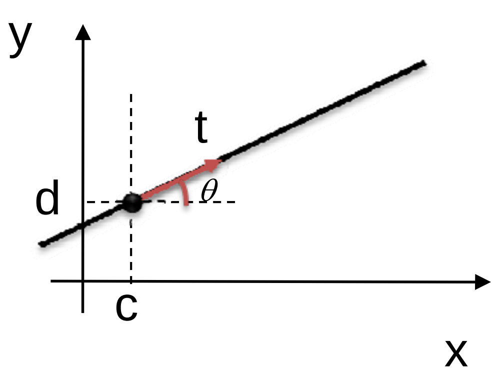


circle

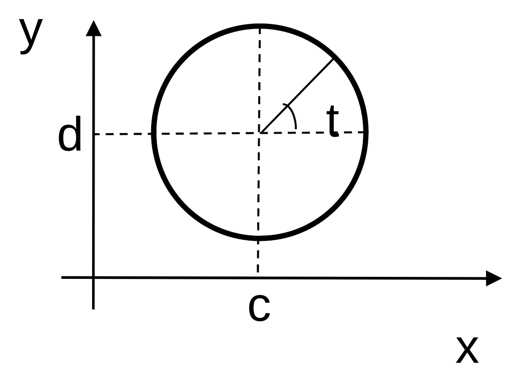
Parametric: $\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} c \\ d \end{bmatrix} + 2t \cdot \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} c \\ d \end{bmatrix} + \begin{bmatrix} r \cdot \cos 2t \\ r \cdot \sin 2t \end{bmatrix}$$

Curve Representation



line

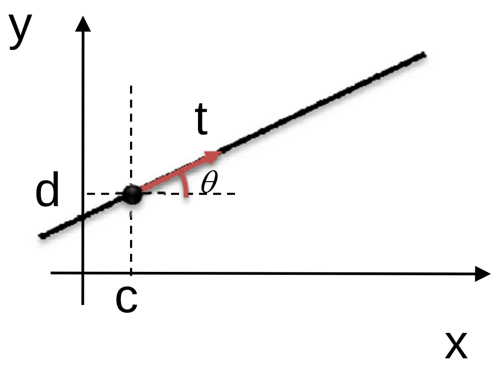


circle

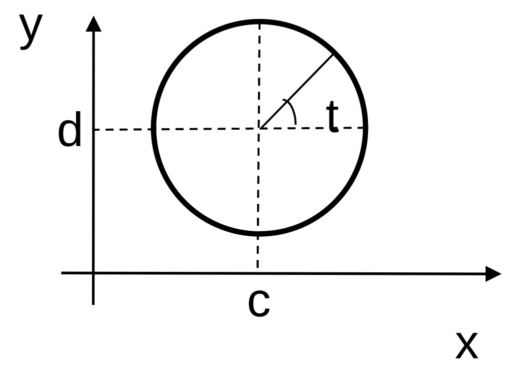
Parametric: $\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} c \\ d \end{bmatrix} + t^3 \cdot \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} c \\ d \end{bmatrix} + \begin{bmatrix} r \cdot \cos t^3 \\ r \cdot \sin t^3 \end{bmatrix}$$

Curve Representation



line



circle

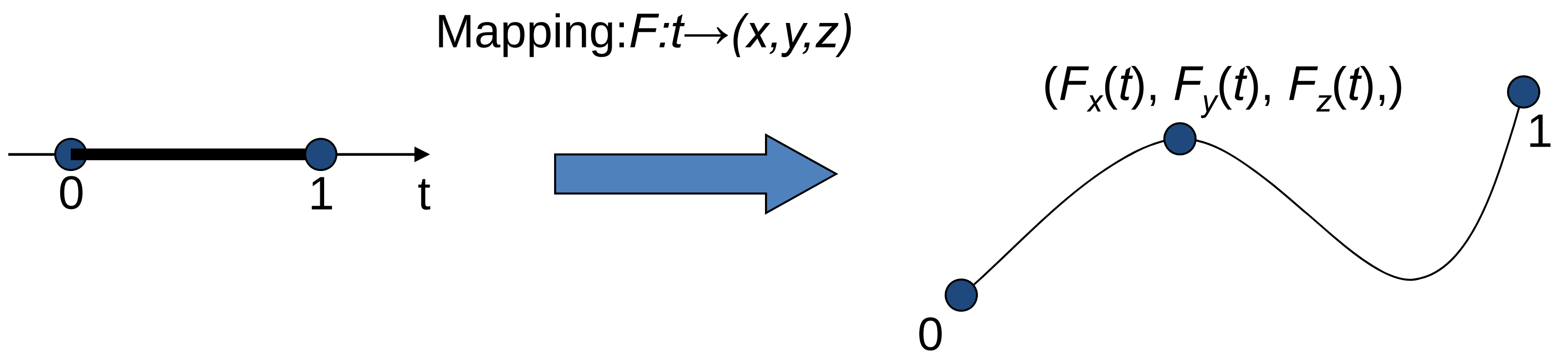
Parametric: $\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} c \\ d \end{bmatrix} + g(t) \cdot \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} c \\ d \end{bmatrix} + \begin{bmatrix} r \cdot \cos g(t) \\ r \cdot \sin g(t) \end{bmatrix}$$

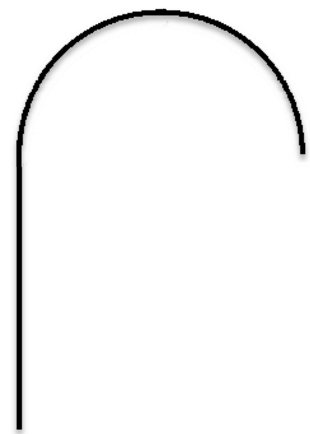
Any invertible function g will result in the same curve

What are Parametric Curves?

- Define a mapping from parameter space to 2D or 3D points
  - A function that takes parameter values and gives back 3D points
- The result is a parametric curve



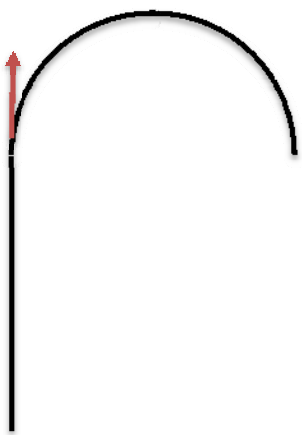
An example of a complex curve



Piecing together basic curves

$$\mathbf{f}(t) = \begin{cases} \mathbf{f}_{line}(2t) & t \le 0.5 \\ \mathbf{f}_{circle}(2t-1) & t > 0.5 \end{cases}$$

Continuities



$$C0: \mathbf{f}_{line}(1) = \mathbf{f}_{circle}(0)$$

$$C1: \mathbf{f}'_{line}(1) = \mathbf{f}'_{circle}(0) \qquad G1: \mathbf{f}'_{line}(1) = k \cdot \mathbf{f}'_{circle}(0)$$

$$C2: \mathbf{f}''_{line}(1) = \mathbf{f}''_{circle}(0) \qquad G2: \mathbf{f}''_{line}(1) = k \cdot \mathbf{f}''_{circle}(0)$$

$$\mathbf{f}(t) = \begin{cases} \mathbf{f}_{line}(2t) & t \le 0.5 \\ \mathbf{f}_{circle}(2t-1) & t > 0.5 \end{cases}$$

Polynomial Pieces

Polynomial functions: $$f(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_n t^n$$

Polynomial curves: $$\mathbf{f}(t) = \sum_{i=0}^{n} \mathbf{a}_i t^i \qquad \mathbf{a}_i \in R^3$$

Polynomial Evaluation

Polynomial functions: $$f(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_n t^n$$

```
f = a[0];
for i = 1:n
    f += a[i]*power(t,i);
end
```

$$f(t) = a_0 + t\left(a_1 + a_2 t^1 + \cdots + a_n t^{n-1}\right)$$

$$f(t) = a_0 + t\left(a_1 + t\left(a_2 + a_3 t \cdots + a_n t^{n-2}\right)\right)$$

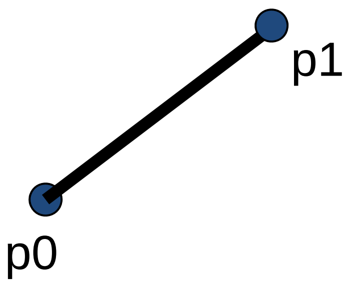
```
f = a[0];
s = 1;
for i = 1:n
    s *= t;
    f += a[i]*s;
end
```

$$f(t) = a_0 + t\left(a_1 + t\left(a_2 + \cdots a_{n-2} + t\left(a_{n-1} + t a_n\right)\right)\right)$$

```
f = a[n];
for i = n-1:-1:0
    f = a[i]+t*f;
end
```

A line Segment

- We have seen the parametric form for a line:



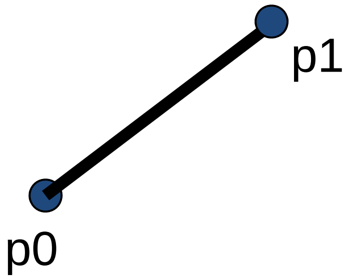
$$x = (1-t)x_0 + tx_1$$
$$y = (1-t)y_0 + ty_1$$
$$z = (1-t)z_0 + tz_1$$

$$\mathbf{p} = \mathbf{f}(t) = (1-t)\cdot\mathbf{p}_0 + t\cdot\mathbf{p}_1$$

- Note that x, y and z are each given by an equation that involves:
  - The parameter t
  - Some user specified control points, x_0 and x_1
- This is an example of a parametric curve

A line Segment

- We have seen the parametric form for a line:



$$
\begin{aligned}
x &= (1-t)x_0 + tx_1 \\
y &= (1-t)y_0 + ty_1 \\
z &= (1-t)z_0 + tz_1
\end{aligned}
$$

$$\mathbf{p} = \mathbf{f}(t) = (1-t)\cdot\mathbf{p}_0 + t\cdot\mathbf{p}_1$$

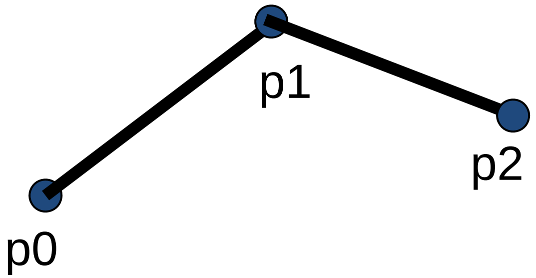
$$\mathbf{p} = \mathbf{f}(t) = \mathbf{p}_0 + t\cdot(\mathbf{p}_1 - \mathbf{p}_0)$$

$$\mathbf{f}(t) = \sum_{i=0}^{n} \mathbf{a}_i t^i \qquad \mathbf{a}_i \in R^3$$

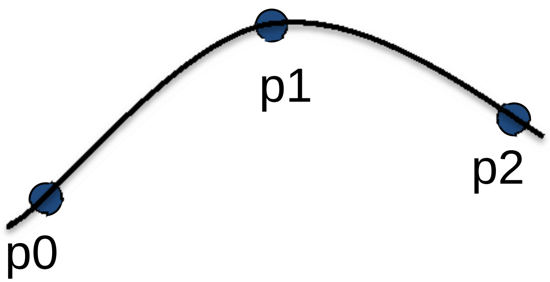
$$
\begin{aligned}
\mathbf{a}_0 &= \mathbf{p}_0 \\
\mathbf{a}_1 &= \mathbf{p}_1 - \mathbf{p}_0
\end{aligned}
\quad \Longleftrightarrow \quad
\begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \end{bmatrix} = \mathbf{B} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \end{bmatrix}
$$

From control points **p**, we can solve coefficients **a**

More control points



More control points



$$\mathbf{f}(t) = \mathbf{a}_0 + \mathbf{a}_1 t + \mathbf{a}_2 t^2$$

$$\mathbf{p}_0 = \mathbf{f}(0) = \mathbf{a}_0 + 0\mathbf{a}_1 + 0^2\mathbf{a}_2$$
$$\mathbf{p}_1 = \mathbf{f}(0.5) = \mathbf{a}_0 + 0.5\mathbf{a}_1 + 0.5^2\mathbf{a}_2$$
$$\mathbf{p}_2 = \mathbf{f}(1.0) = \mathbf{a}_0 + 1\mathbf{a}_1 + 1^2\mathbf{a}_2$$

$$\begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0.5 & 0.25 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \end{bmatrix} = \mathbf{C} \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \end{bmatrix} = \mathbf{C}^{-1} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -3 & 4 & -1 \\ 2 & -4 & 2 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix}$$

$$\mathbf{f}(t) = \begin{bmatrix} 1 & t & t^2 \end{bmatrix} \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \end{bmatrix}$$
$$= \mathbf{t}\mathbf{C}^{-1} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix} = \mathbf{t}\mathbf{B} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix}$$

By solving a matrix equation that satisfies the constraints, we can get polynomial coefficients

Two views on polynomial curves

$$\mathbf{f}(t) = \begin{bmatrix} 1 & t & t^2 \end{bmatrix} \mathbf{B} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix}$$

$$\mathbf{f}(t) = \begin{bmatrix} 1 & t & t^2 \end{bmatrix} \left(\mathbf{B} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix} \right)$$

$$\mathbf{f}(t) = \mathbf{a}_0 + \mathbf{a}_1 t + \mathbf{a}_2 t^2$$

From control points **p**,
we can compute coefficients **a**

$$\mathbf{f}(t) = \left(\begin{bmatrix} 1 & t & t^2 \end{bmatrix} \mathbf{B} \right) \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix}$$

$$= \begin{bmatrix} b_0(t) & b_1(t) & b_2(t) \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \end{bmatrix}$$

$$\mathbf{f}(t) = b_0(t)\mathbf{p}_0 + b_1(t)\mathbf{p}_1 + b_2(t)\mathbf{p}_2$$

Each point on the curve is a linear
blending of the control points

What are b0(t), b1(t), …?

Two control point case:

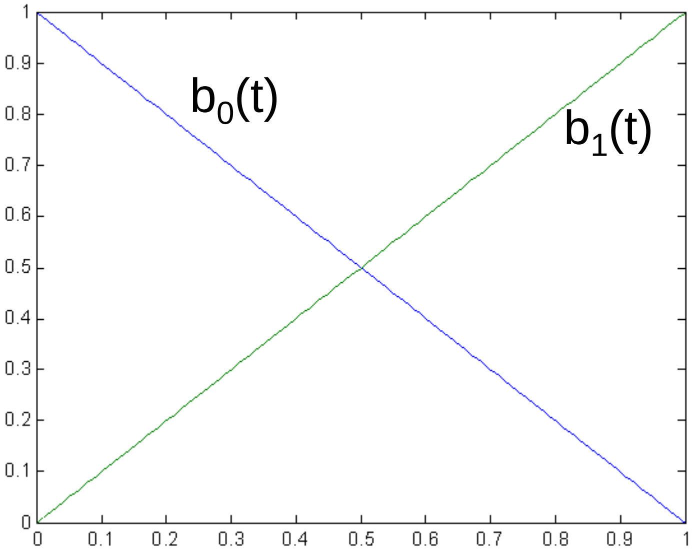
$$\mathbf{f}(t) = \left(\begin{bmatrix} 1 & t \end{bmatrix} \mathbf{B} \right) \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \end{bmatrix}$$

$$\mathbf{f}(t) = b_0(t)\mathbf{p}_0 + b_1(t)\mathbf{p}_1$$

$$\begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \end{bmatrix}$$

$$\begin{bmatrix} b_0(t) & b_1(t) \end{bmatrix} = \begin{bmatrix} 1 & t \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1-t & t \end{bmatrix}$$



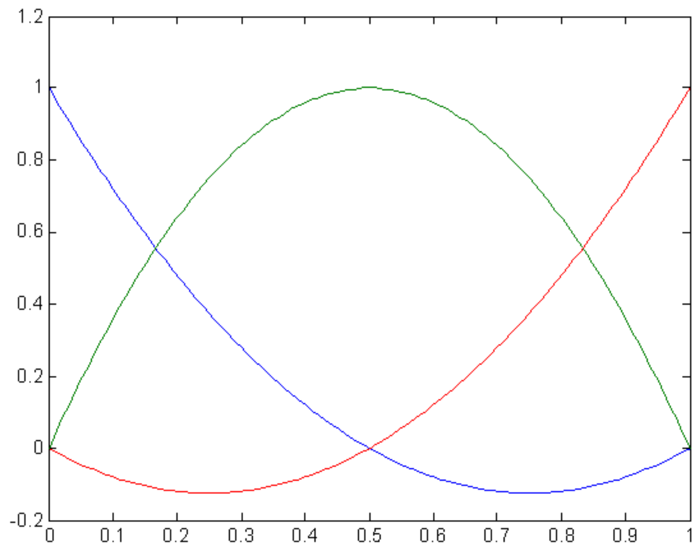
What are b0, b1, …?

Three control point case:

$$\mathbf{f}(t) = \left(\begin{bmatrix}1 & t & t^2\end{bmatrix}\mathbf{B}\right)\begin{bmatrix}\mathbf{p}_0\\ \mathbf{p}_1\\ \mathbf{p}_2\end{bmatrix}$$

$$\mathbf{f}(t) = b_0(t)\mathbf{p}_0 + b_1(t)\mathbf{p}_1 + b_2(t)\mathbf{p}_2$$

$$\begin{bmatrix}\mathbf{a}_0\\ \mathbf{a}_1\\ \mathbf{a}_2\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\ -3 & 4 & -1\\ 2 & -4 & 2\end{bmatrix}\begin{bmatrix}\mathbf{p}_0\\ \mathbf{p}_1\\ \mathbf{p}_2\end{bmatrix}$$



$$\begin{bmatrix}b_0(t) & b_1(t) & b_2(t)\end{bmatrix} = \begin{bmatrix}1 & t & t^2\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ -3 & 4 & -1\\ 2 & -4 & 2\end{bmatrix}$$

$$= \begin{bmatrix}1 - 3t + 2t^2 & 4t - 4t^2 & -t + 2t^2\end{bmatrix}$$

Which is $b_0(t)$?

$$b_0(t) + b_1(t) + b_2(t) = ? \quad \equiv 1$$

What are b0, b1, …?

Three control point case:

$$\mathbf{f}(t) = \left(\begin{bmatrix}1 & t & t^2\end{bmatrix}\mathbf{B}\right)\begin{bmatrix}\mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2\end{bmatrix}$$

$$\mathbf{f}(t) = b_0(t)\mathbf{p}_0 + b_1(t)\mathbf{p}_1 + b_2(t)\mathbf{p}_2$$

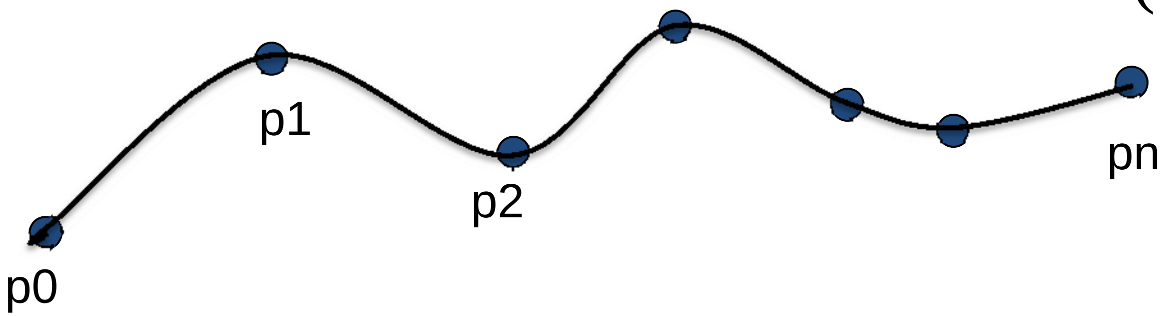
- Why $b_0(t) + b_1(t) + b_2(t) \equiv 1$ is important?
  - Translation-invariant interpolation

$$\begin{bmatrix}\mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2\end{bmatrix} \Rightarrow \begin{bmatrix}\mathbf{p}_0 + \mathbf{d} \\ \mathbf{p}_1 + \mathbf{d} \\ \mathbf{p}_2 + \mathbf{d}\end{bmatrix}$$

$$\begin{aligned}\mathbf{f}_{new}(t) &= b_0(t)(\mathbf{p}_0 + \mathbf{d}) + b_1(t)(\mathbf{p}_1 + \mathbf{d}) + b_2(t)(\mathbf{p}_2 + \mathbf{d}) \\ &= b_0(t)\mathbf{p}_0 + b_1(t)\mathbf{p}_1 + b_2(t)\mathbf{p}_2 \\ &\quad + \left(b_0(t) + b_1(t) + b_2(t)\right)\mathbf{d} \\ &= \mathbf{f}(t) + \mathbf{d} \qquad \text{for any t}\end{aligned}$$

Many control points

$$\mathbf{f}(t) = \sum_{i=0}^{n} \mathbf{a}_i t^i \quad \mathbf{a}_i \in R^3$$



$$\mathbf{f}(t_0) = \mathbf{a}_0 + \mathbf{a}_1 t_0 + \mathbf{a}_2 t_0{}^2 + \cdots + \mathbf{a}_n t_0{}^n = \mathbf{p}_0$$

$$\mathbf{f}(t_1) = \mathbf{a}_0 + \mathbf{a}_1 t_1 + \mathbf{a}_2 t_1{}^2 + \cdots + \mathbf{a}_n t_1{}^n = \mathbf{p}_1$$

$$\cdots$$

$$\mathbf{f}(t_n) = \mathbf{a}_0 + \mathbf{a}_1 t_n + \mathbf{a}_2 t_n{}^2 + \cdots + \mathbf{a}_n t_n{}^n = \mathbf{p}_n$$

$$\begin{bmatrix} 1 & t_0 & \cdots & t_0^n \\ 1 & t_1 & \cdots & t_1^n \\ & \cdots & & \\ 1 & t_n & \cdots & t_n^n \end{bmatrix} \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_n \end{bmatrix} = \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_n \end{bmatrix}$$

Straightforward, but not intuitive

Many control points

- A shortcut $\mathbf{f}(t)=\sum_{i=0}^{n} b_i(t)\mathbf{p}_i$

Getting $b_i(t)$ is easier!

- Goal: $\mathbf{f}(t_i)=\mathbf{p}_i$

$$\sum_{i=0}^{n} b_i(t) = ? \equiv 1$$ Why?

- Idea:

$$b_i(t_i)=1$$
$$b_i(t_j)=0 \quad j\neq i$$

$$\sum_{i=0}^{n} b_i(t_j) = ? = 1, \forall j = 0,1,\cdots n$$

$$B(t)=\sum_{i=0}^{n} b_i(t)$$ is a polynomial of degree n

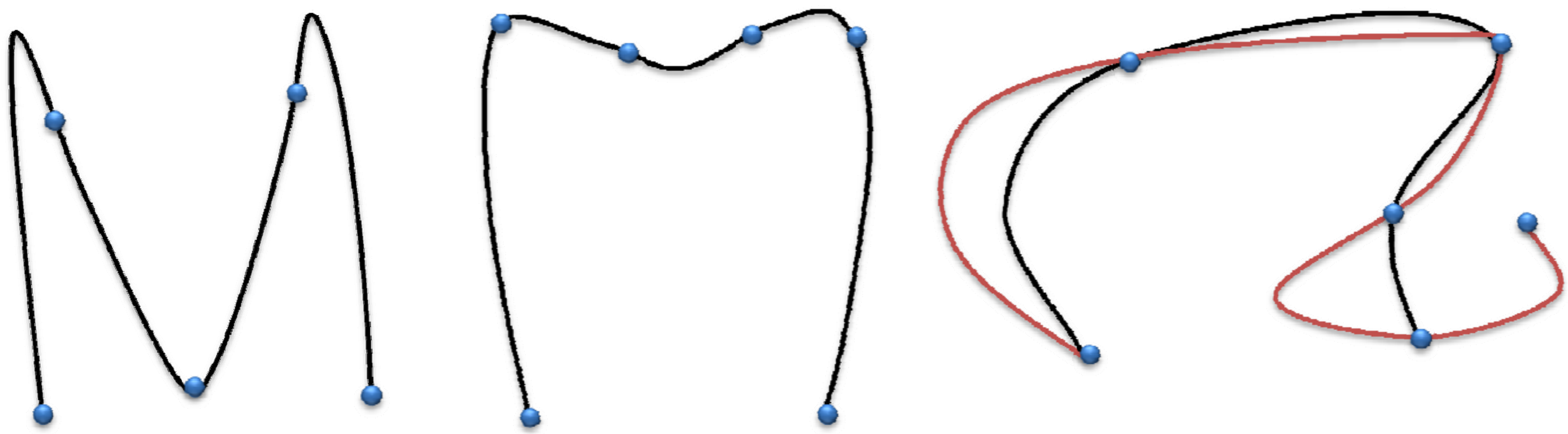
- Magic: $$b_i(t)=\prod_{j=0,\, j\neq i}^{n} \frac{t-t_j}{t_i-t_j}$$

If an n-degree polynomial has the same value at n+1 locations, it must be a constant polynomial

$$=\frac{t-t_0}{t_i-t_0}\frac{t-t_1}{t_i-t_1}\cdots\frac{t-t_{i-1}}{t_i-t_{i-1}}\cancel{\frac{t-t_i}{t_i-t_i}}\frac{t-t_{i+1}}{t_i-t_{i+1}}\cdots\frac{t-t_n}{t_i-t_n}$$

Lagrange Interpolation

Lagrange Polynomial Interpolation

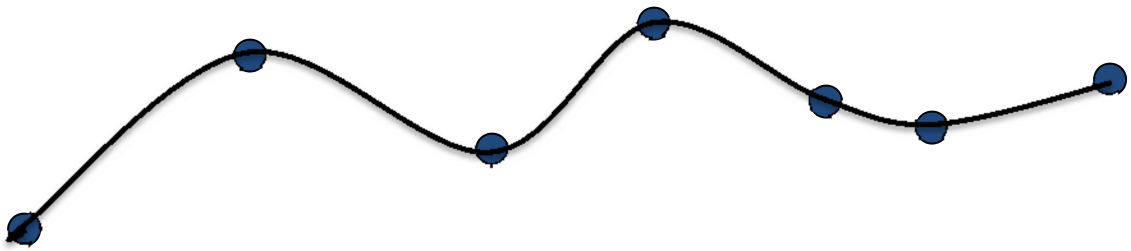


Lagrange Interpolation Demo

- http://www.math.ucla.edu/~baker/java/hoefer/Lagrange.htm

- Properties:
  - The curve passes through all the control points
  - Very smooth: C^n for n control points
  - Do not have local control
  - Overshooting

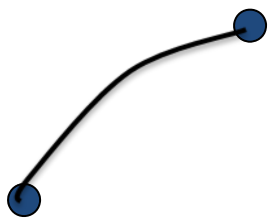
Piecewise Cubic Polynomials



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

- Desired Features:
  - Interpolation
  - Local control
  - C1 or C2

Natural Cubics



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

$$\mathbf{p}_0 = \mathbf{f}(0) = \mathbf{a}_0 \quad + 0\mathbf{a}_1 \quad + 0^2\mathbf{a}_2 \quad + 0^3\mathbf{a}_3$$

$$\mathbf{p}_1 = \mathbf{f}'(0) = \quad \mathbf{a}_1 \quad + 2\cdot 0\mathbf{a}_2 \quad + 3\cdot 0^2\mathbf{a}_3$$

$$\mathbf{p}_2 = \mathbf{f}''(0) = \quad + 2\mathbf{a}_2 \quad + 6\cdot 0\mathbf{a}_3$$

$$\mathbf{p}_3 = \mathbf{f}(1) = \mathbf{a}_0 \quad + \mathbf{a}_1 \quad + \mathbf{a}_2 \quad + \mathbf{a}_3$$

$$\begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ -1 & -1 & -0.5 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

Natural Cubics

- If we have n points, how to use natural cubic to interpolate them?
  - Define the first and second derivatives for the starting point of the first segment.
  - Compute the cubic for the first segment
  - Copy the first and second derivatives for the end point of the first segment to the starting point for the second segment
- How many segments do we have for n control points?
  - n-1

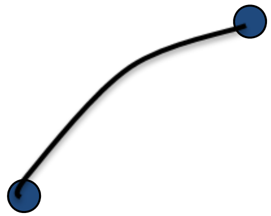
Natutral Cubic Curves

- Demo: http://www.cse.unsw.edu.au/~lambert/splines/

Natural Cubics

	Interpolate control points	Has local control	C2 continuity
Natural cubics			

Hermit Cubics



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

$$\mathbf{p}_0 = \mathbf{f}(0) = \mathbf{a}_0 \quad + 0\mathbf{a}_1 \quad + 0^2\mathbf{a}_2 \quad + 0^3\mathbf{a}_3$$

$$\mathbf{p}_1 = \mathbf{f}'(0) = \quad \mathbf{a}_1 \quad + 2\cdot 0\mathbf{a}_2 \quad + 3\cdot 0^2\mathbf{a}_3$$

$$\mathbf{p}_2 = \mathbf{f}(1) = \mathbf{a}_0 \quad + \mathbf{a}_1 \quad + \mathbf{a}_2 \quad + \mathbf{a}_3$$

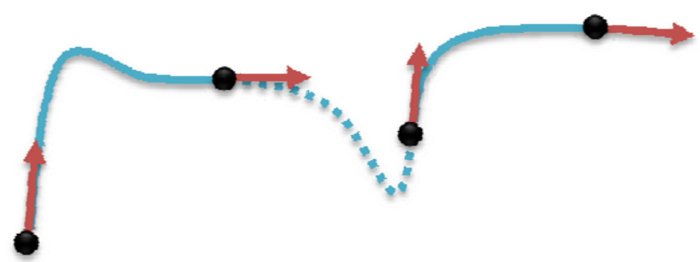
$$\mathbf{p}_3 = \mathbf{f}'(1) = \quad \mathbf{a}_1 \quad + 2\mathbf{a}_2 \quad + 3\mathbf{a}_3$$

$$\begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -2 & 3 & -1 \\ 2 & 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

Hermite Cubic Curves

- If we have n points, how to use Hermite cubic to interpolate them?
  - For each pair, using the first derivatives at starting and ending points to define the inbetween



- How many segments do we have for n controls?
  - n/2-1

Hermite Cubies

	Interpolate control points	Has local control	C2 continuity
Natural cubics	Yes	No	Yes
Hermite cubics			