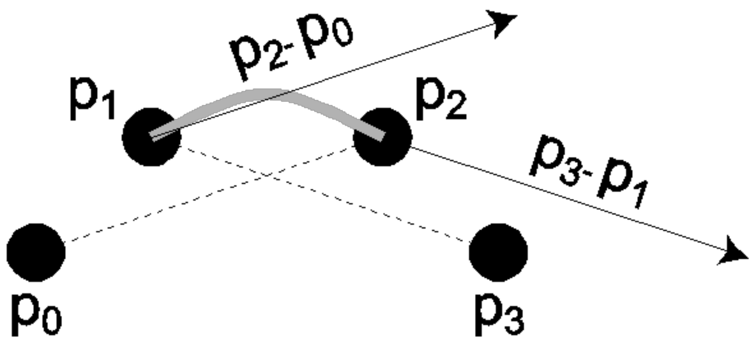
CS559: Computer Graphics

Lecture 14: More on Curves

Li Zhang

Spring 2010

Catmull-Rom Cubics



$$\mathbf{C}_{Hermite}\begin{bmatrix}\mathbf{a}_0\\ \mathbf{a}_1\\ \mathbf{a}_2\\ \mathbf{a}_3\end{bmatrix}=\begin{bmatrix}\mathbf{f}(0)\\ \mathbf{f}'(0)\\ \mathbf{f}(1)\\ \mathbf{f}'(1)\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & 0\\ -0.5 & 0 & 0.5 & 0\\ 0 & 0 & 1 & 0\\ 0 & -0.5 & 0 & 0.5\end{bmatrix}\begin{bmatrix}\mathbf{p}_0\\ \mathbf{p}_1\\ \mathbf{p}_2\\ \mathbf{p}_3\end{bmatrix}$$

$$\begin{bmatrix}\mathbf{a}_0\\ \mathbf{a}_1\\ \mathbf{a}_2\\ \mathbf{a}_3\end{bmatrix}=\mathbf{B}_{Catmull}\begin{bmatrix}\mathbf{p}_0\\ \mathbf{p}_1\\ \mathbf{p}_2\\ \mathbf{p}_3\end{bmatrix}$$

$$\mathbf{f}(0)=\mathbf{p}_1$$

$$\mathbf{f}'(0)=\frac{1}{2}\left(\mathbf{p}_2-\mathbf{p}_0\right)$$

$$\mathbf{f}(1)=\mathbf{p}_2$$

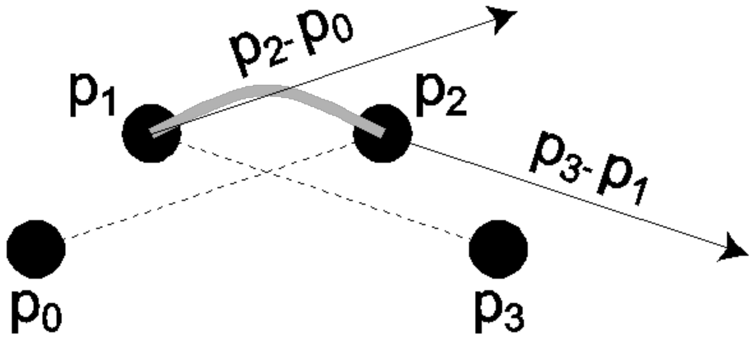
$$\mathbf{f}'(1)=\frac{1}{2}\left(\mathbf{p}_3-\mathbf{p}_1\right)$$

Catmull-Rom Cubic Curves

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

- n control points define n-2 segments
- Changing 1 point affects neighboring 4 segments

Cardinal Cubics



$$\mathbf{f}(0) = \mathbf{p}_1$$

$$\mathbf{f}'(0) = s \cdot (\mathbf{p}_2 - \mathbf{p}_0)$$

$$\mathbf{f}(1) = \mathbf{p}_2$$

$$\mathbf{f}'(1) = s \cdot (\mathbf{p}_3 - \mathbf{p}_1)$$

s: tension control

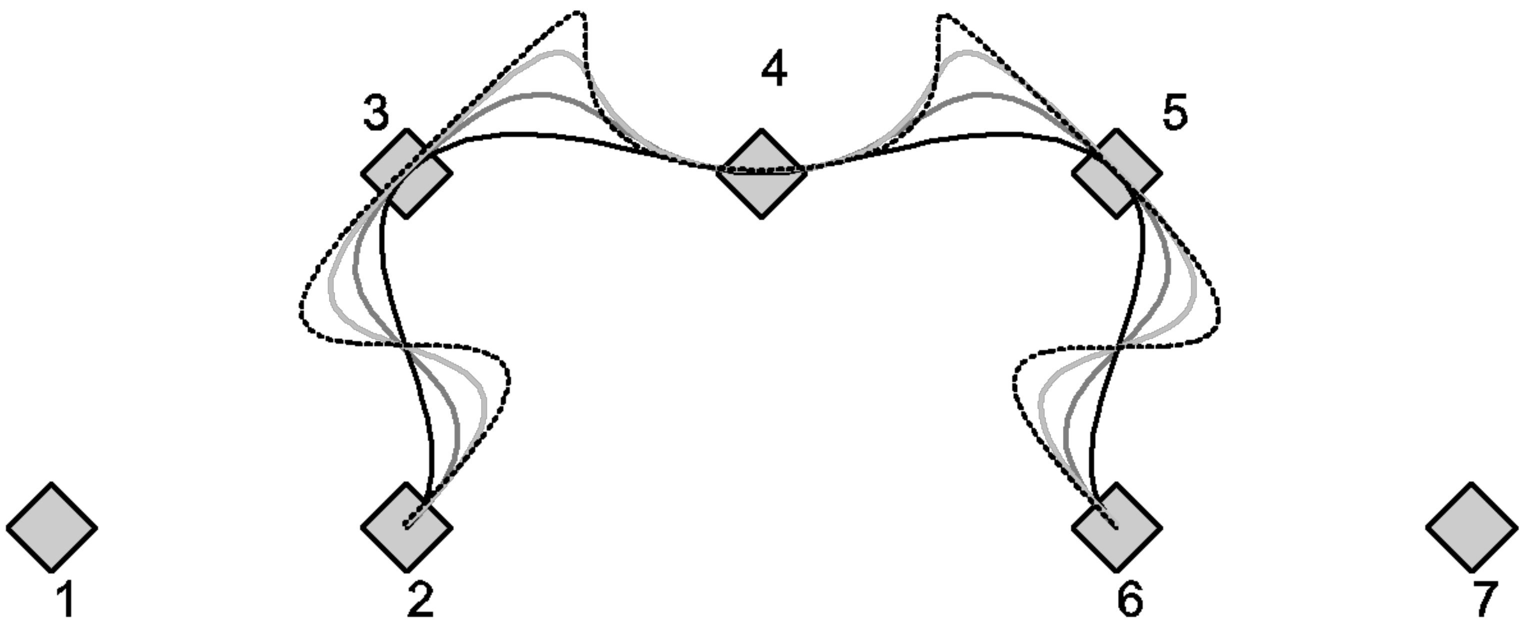
$$\mathbf{C}_{Hermite} \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{f}(0) \\ \mathbf{f}'(0) \\ \mathbf{f}(1) \\ \mathbf{f}'(1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -s & 0 & s & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -s & 0 & s \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \end{bmatrix} = \mathbf{B}_{Cardinal} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

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

Cardinal Cubic Curves

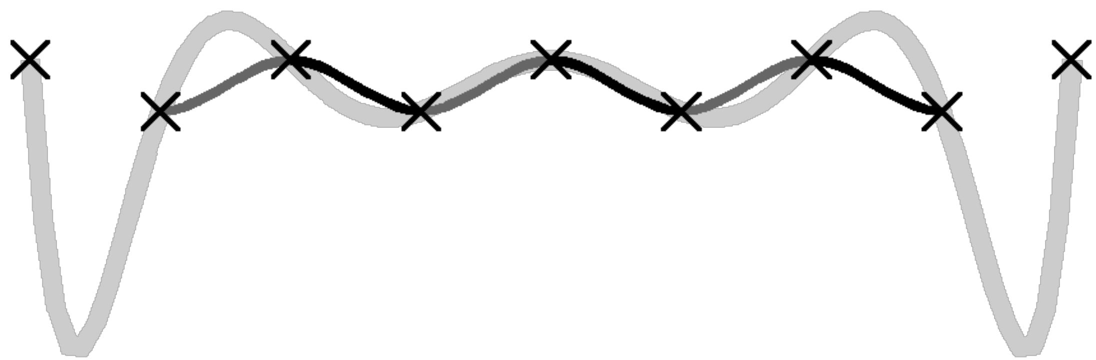
- 4 different s values



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

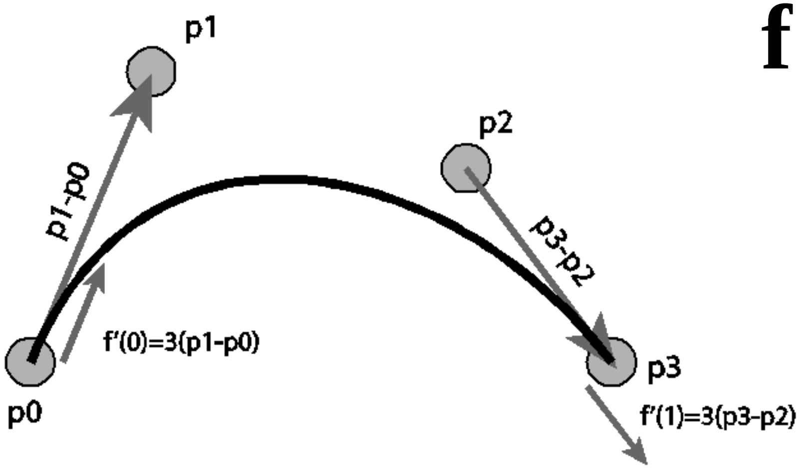
	Interpolate control points	Has local control	C2 continuity
Natural cubics	Yes	No	Yes
Hermite cubics	Yes	Yes	No
Cardinal Cubics	Yes	Yes	No

Cardinal Cubics vs Lagrange Polynomials



- Sacrifice higher order smoothness for
  - Local control
  - Avoid overshooting
- When higher order smoothness may be important?

Bezier Cubics



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

$$\mathbf{C}_{Hermite} \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{f}(0) \\ \mathbf{f}'(0) \\ \mathbf{f}(1) \\ \mathbf{f}'(1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

$$\mathbf{f}(0) = \mathbf{p}_0$$

$$\mathbf{f}'(0) = 3 \cdot (\mathbf{p}_1 - \mathbf{p}_0)$$

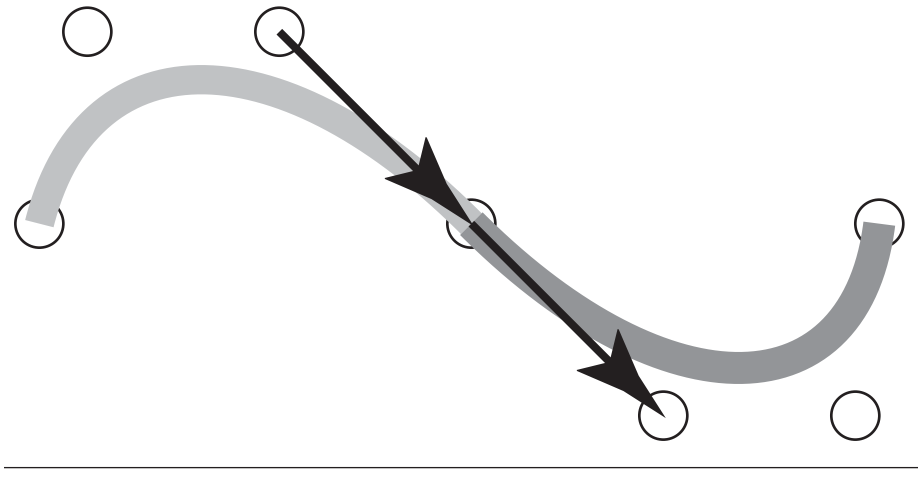
$$\mathbf{f}(1) = \mathbf{p}_3$$

$$\mathbf{f}'(1) = 3 \cdot (\mathbf{p}_3 - \mathbf{p}_2)$$

$$\begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \end{bmatrix} = \mathbf{B}_{Bezier} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

$$\mathbf{B}_{Bezier} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

Bezier Cubic Curves



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

Bezier Cubic Curves in Illustrator

- Demo

- Changing one control point will change 2 neighboring segments

Generalization of Bezier

$$\mathbf{f}(t) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \mathbf{B}_{Bezier} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} \qquad \mathbf{B}_{Bezier} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

$$\mathbf{f}(t) = b_0(t)\mathbf{p}_0 + b_1(t)\mathbf{p}_1 + b_2(t)\mathbf{p}_2 + b_3(t)\mathbf{p}_3$$

$$b_0(t) = (1-t)^3$$

$$b_1(t) = 3t(1-t)^2$$

$$b_2(t) = 3t^2(1-t)$$

$$b_3(t) = t^3$$

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

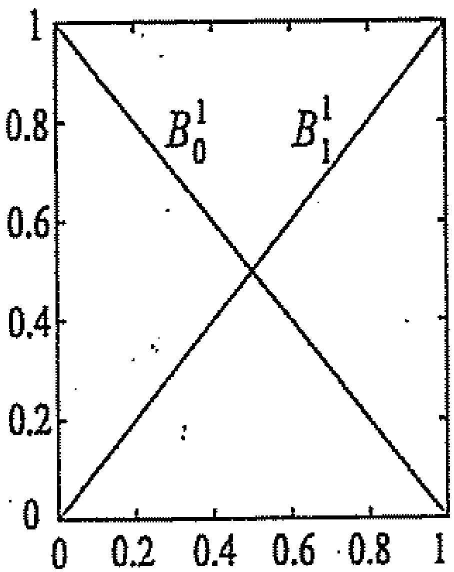
$$1 \equiv \left(\left(1-t\right)+t\right)^3$$

Generalization of Bezier

- If we have $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \cdots \mathbf{p}_n$ $\quad \mathbf{f}(t) = \sum_{i=0}^{n} \boxed{b_i(t)} \mathbf{p}_i$

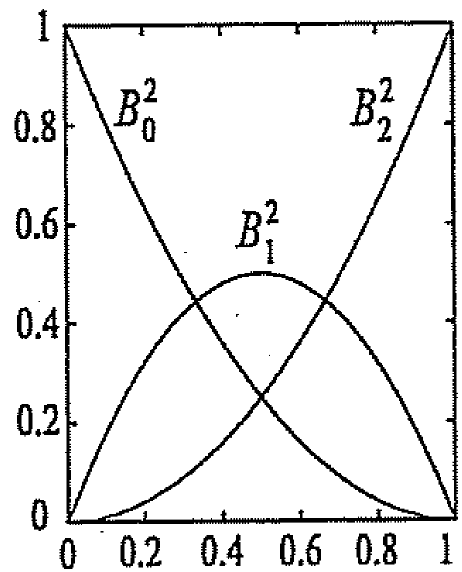
$$1 = \left((1-t)+t\right)^n = \sum_{i=0}^{n} \boxed{\binom{n}{i} t^i (1-t)^{n-i}} \qquad \binom{n}{i} = \frac{n!}{i!(n-i)!}$$

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

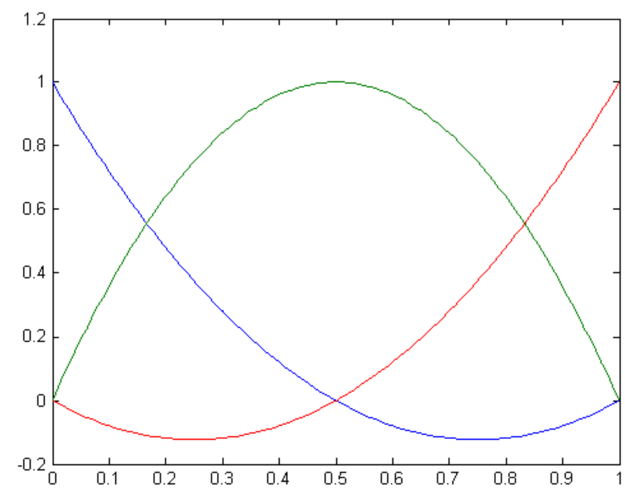


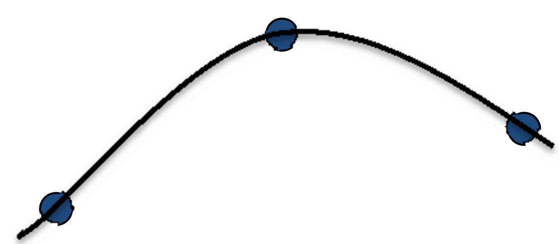
n=1

$$(1-t)^2\mathbf{p}_0 + 2t(1-t)\mathbf{p}_1 + t^2\mathbf{p}_2$$



n=2



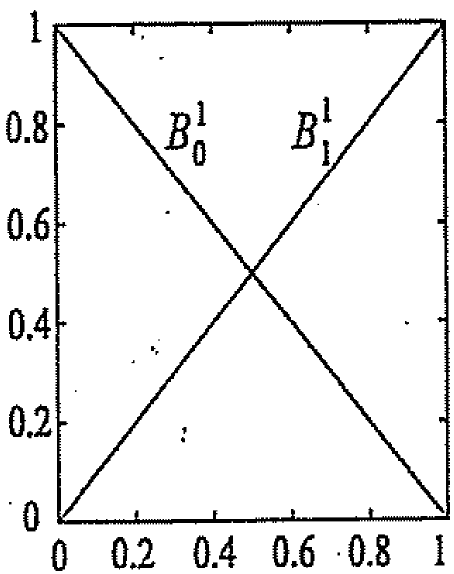


Generalization of Bezier

- If we have $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \cdots \mathbf{p}_n$ $\quad \mathbf{f}(t) = \sum_{i=0}^{n} \boxed{b_i(t)}\, \mathbf{p}_i$

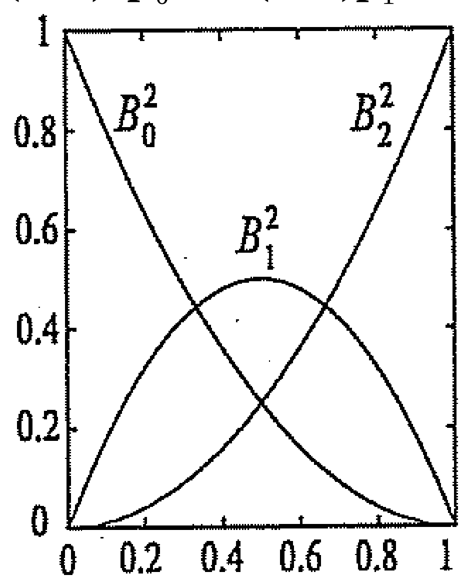
$$1 = \left((1-t)+t\right)^n = \sum_{i=0}^{n} \boxed{\binom{n}{i} t^i (1-t)^{n-i}} \qquad \binom{n}{i} = \frac{n!}{i!(n-i)!}$$

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

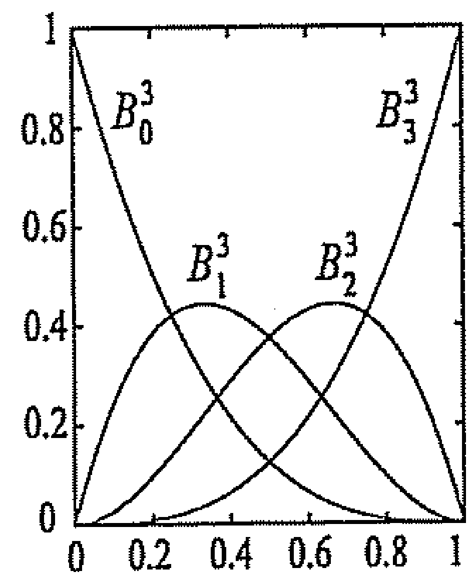


n=1

$(1-t)^2\mathbf{p}_0 + 2t(1-t)\mathbf{p}_1 + t^2\mathbf{p}_2$



n=2

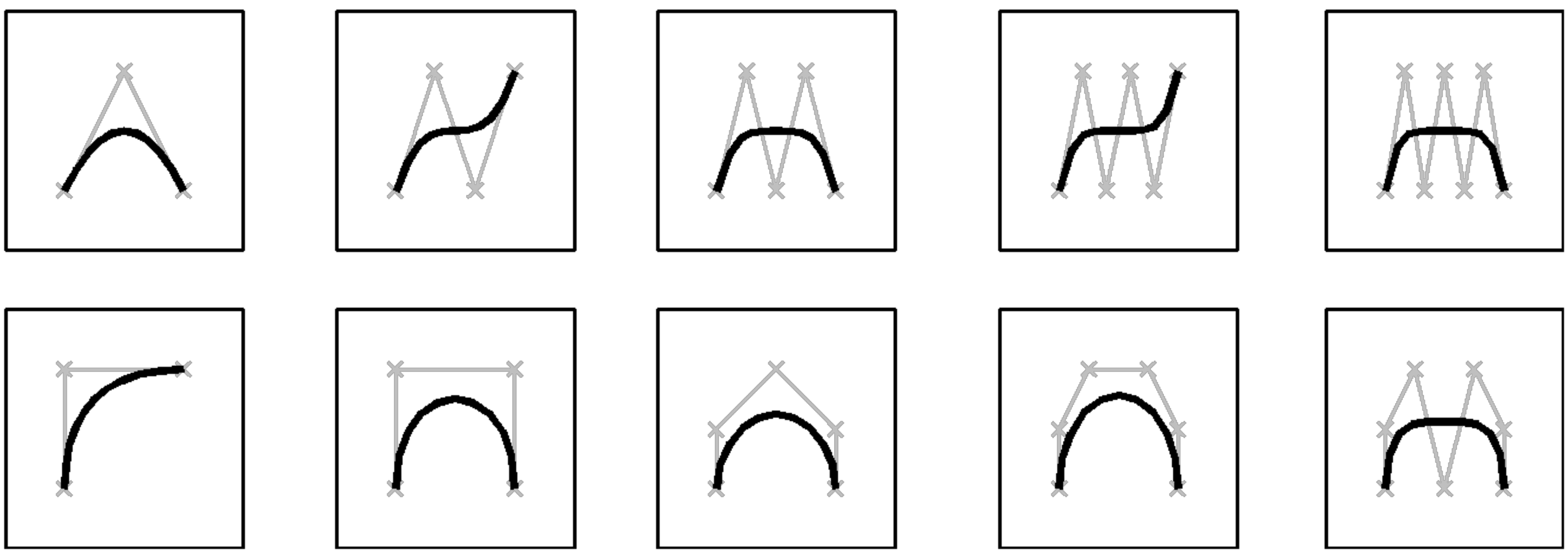


n=3

Generalization of Bezier

- If we have $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \cdots \mathbf{p}_n$ $\quad \mathbf{f}(t) = \sum_{i=0}^{n} \boxed{b_i(t)} \mathbf{p}_i$

$$1 = \left((1-u)+u\right)^n = \sum_{i=0}^{n} \boxed{\binom{n}{i} u^i (1-u)^{n-i}} \qquad \binom{n}{i} = \frac{n!}{i!(n-i)!}$$



n=2 n=3 n=4 n=5 n=6

Bezier Curve Properties

- The first and last control points are interpolated
- The tangent to the curve at the first control point is along the line joining the first and second control points

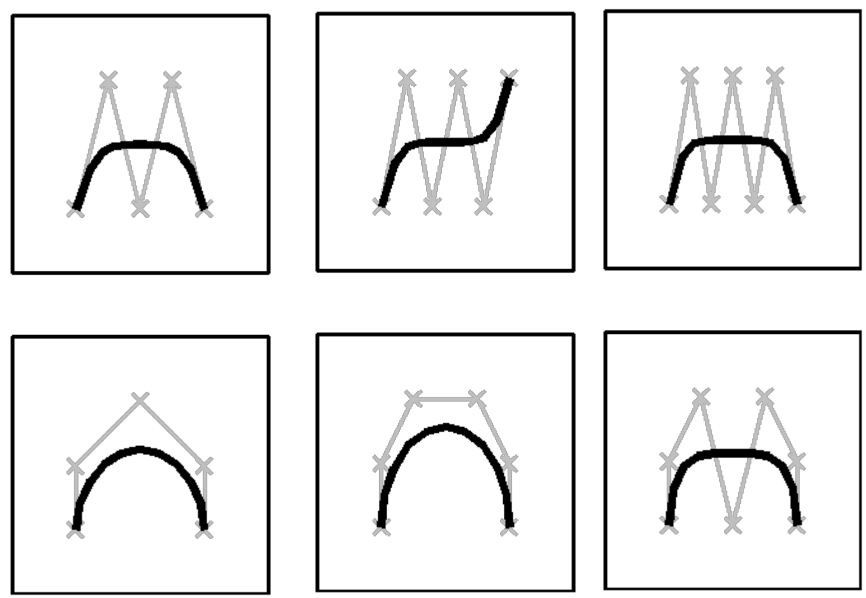
$$\mathbf{f}(t) = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} \mathbf{p}_i$$

$$\mathbf{f}'(0) = ? \;= n \cdot (\mathbf{p}_1 - \mathbf{p}_0)$$

$$\mathbf{f}'(1) = n \cdot (\mathbf{p}_n - \mathbf{p}_{n-1})$$

- The tangent at the last control point is along the line joining the second last and last control points
- K’th order derivative at 0 depends on p0 … pK
  - Why important?
    - Design curves with higher order continuity

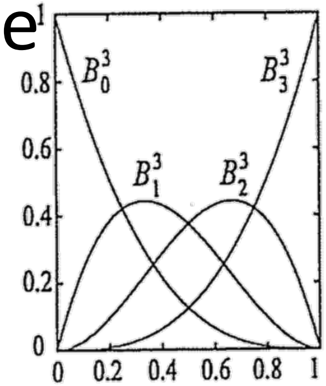
- The curve lies entirely within the convex hull of its control points
  - The basis functions sum to 1 and are everywhere positive → convex combination
  - Terminologies:
    - Convex shape
    - convex hull
    - convex combination
  - Why important?
    - Collision detection



Bezier Curve Properties

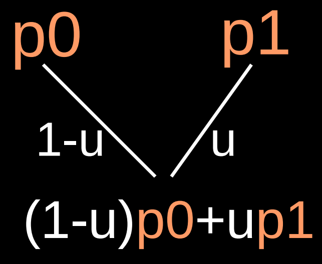
- Reversing the control points yields the same curve $\mathbf{f}_{bezier}(t;\mathbf{p}_0,\mathbf{p}_1,\cdots\mathbf{p}_n)=\mathbf{f}_{bezier}(1-t;\mathbf{p}_n,\mathbf{p}_{n-1},\cdots\mathbf{p}_0)$
  - why?

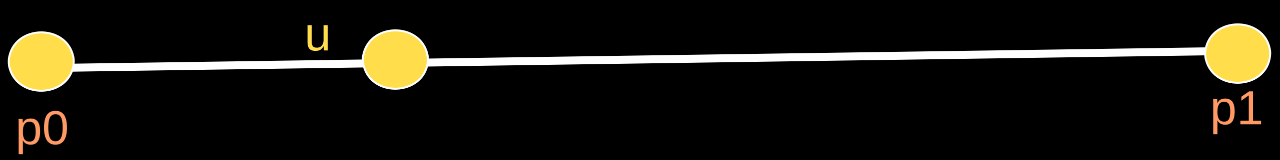
$$\binom{n}{i}=\binom{n}{n-i}$$



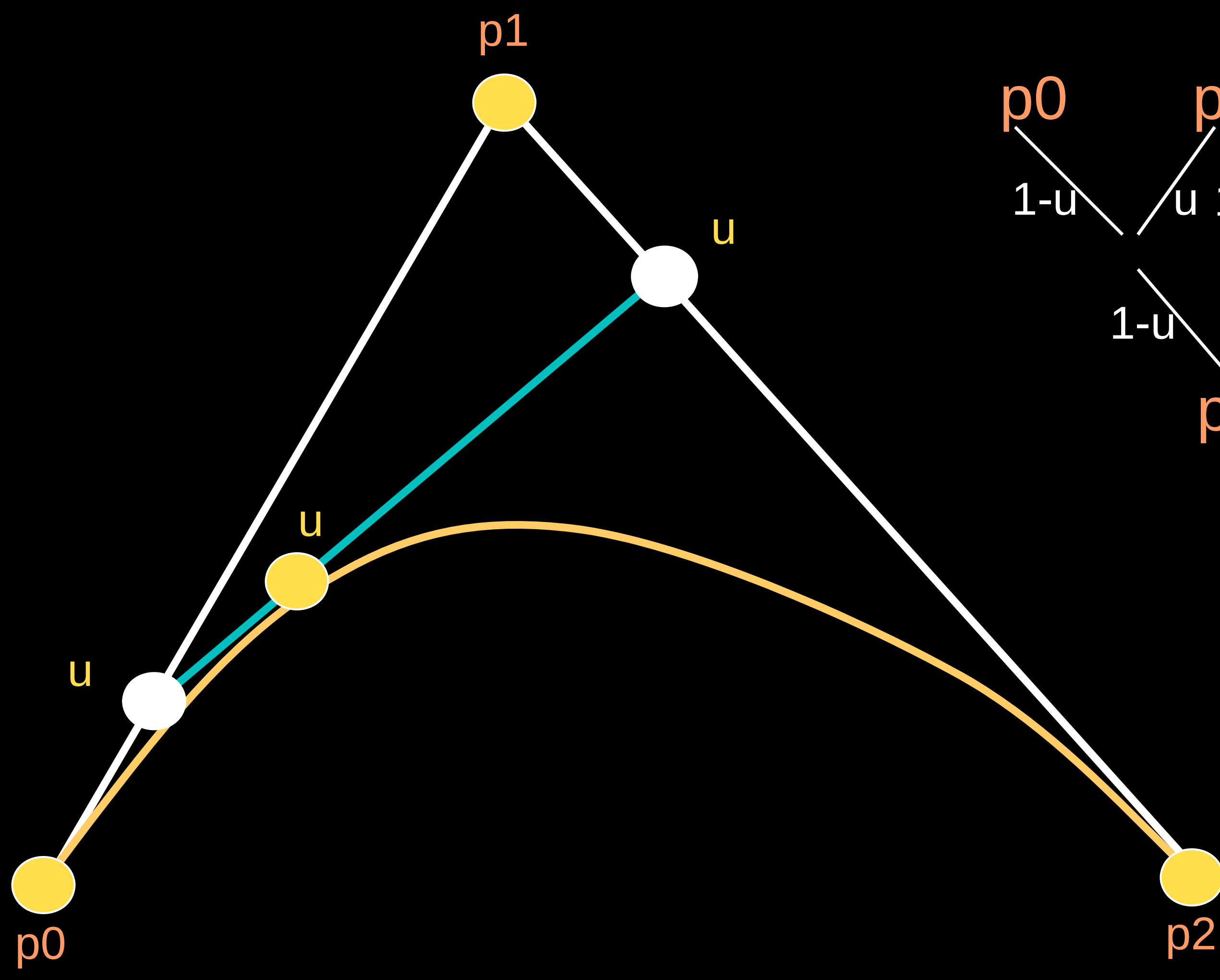
- The curves are affine invariant – Translating, scaling, rotating, or skewing the control points is the same as performing those operations on the curve itself.
  - Why important?
    - Efficient rendering
- Subdivision for efficient rendering, editing, approximating.

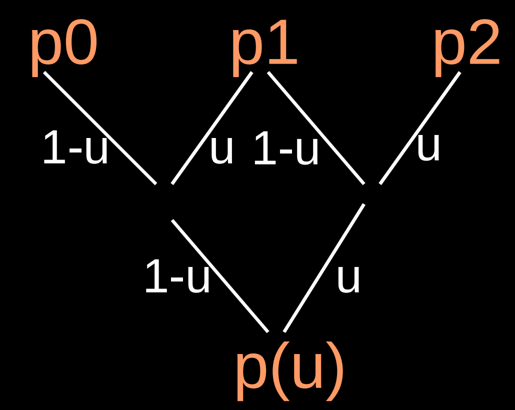
Geometric Interpretation: Line





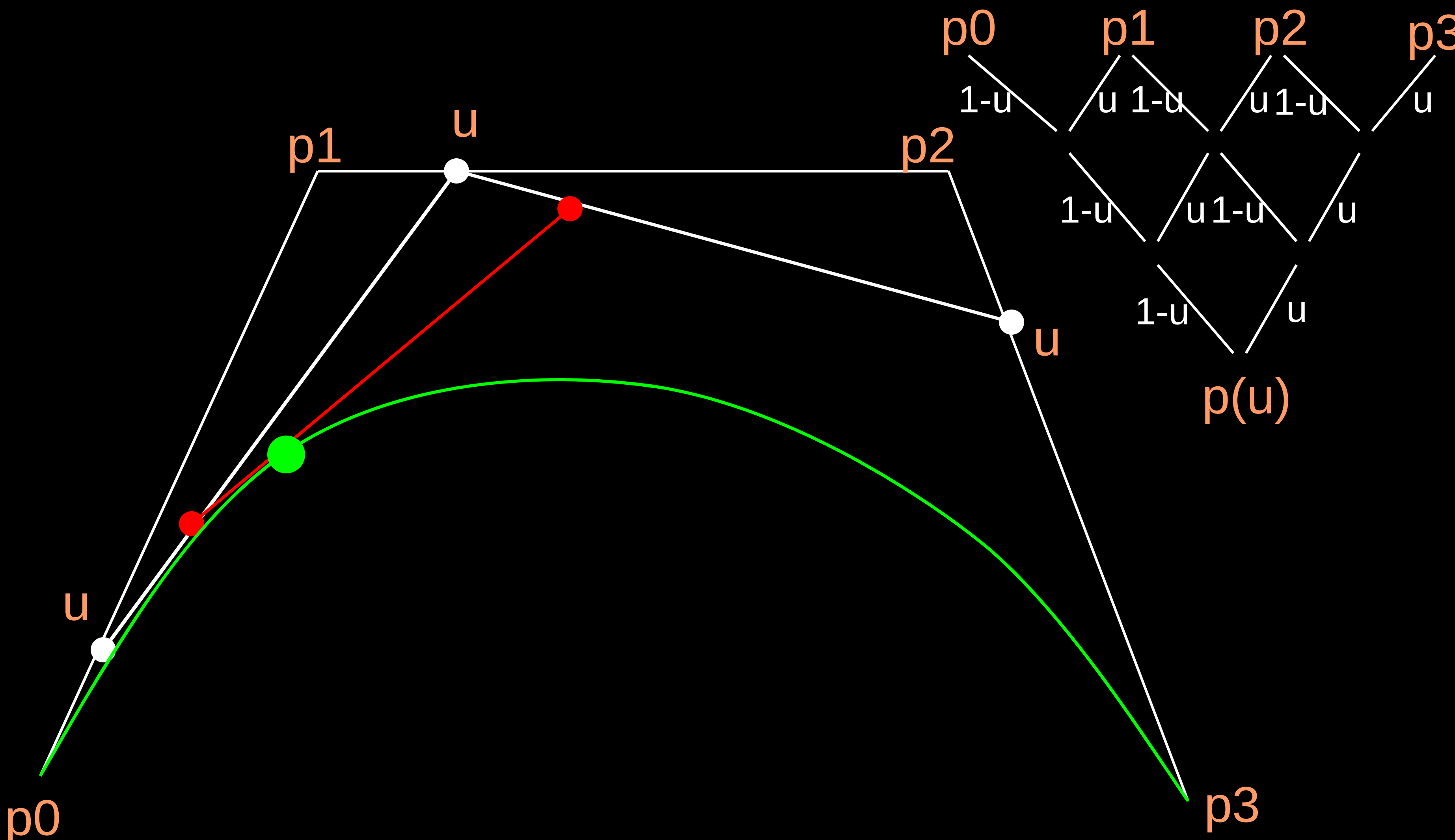
Geometric Interpretation: Quadratic





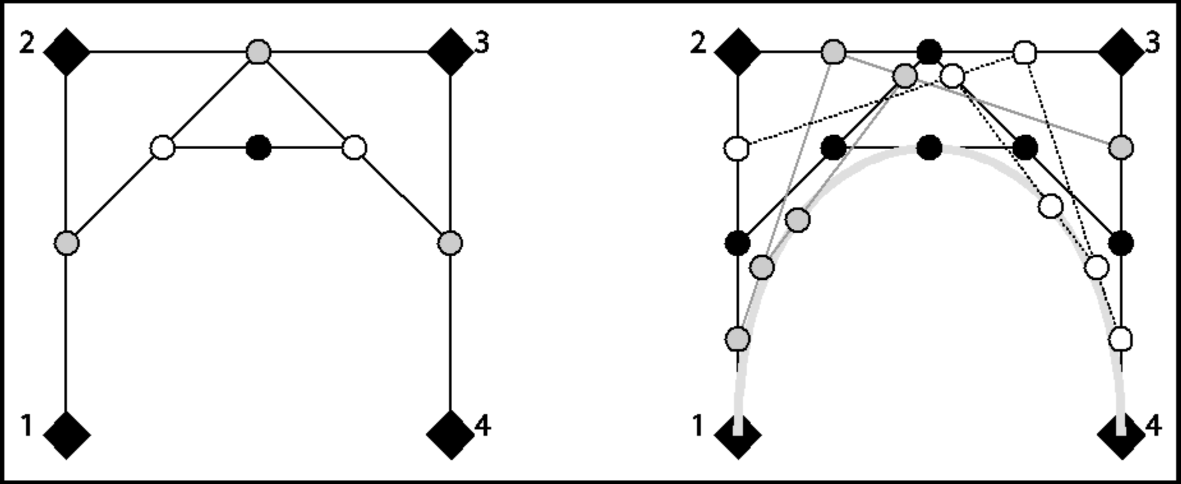
http://www1.cs.columbia.edu/~cs4160/html06f/

Geometric Interpretation: Cubic



http://www1.cs.columbia.edu/~cs4160/html06f/

Changing u



u=0.5

u=0.25, u=0.5, u=0.75

De Casteljau algorithm, Recursive Rendering

Subdivision

```
DeCasteljau(float p[N][3], float u) {

    for (int n = N-1; n >= 1; --n) {
        for (int j = 0; j < n; ++j) {
            p[j][0] = (1-u) * p[j][0] + u * p[j+1][0];
            p[j][1] = (1-u) * p[j][1] + u * p[j+1][1];
            p[j][2] = (1-u) * p[j][2] + u * p[j+1][2];
        }
    }
    //(p[0][0], p[0][1], p[0][2]) saves the result
}
```

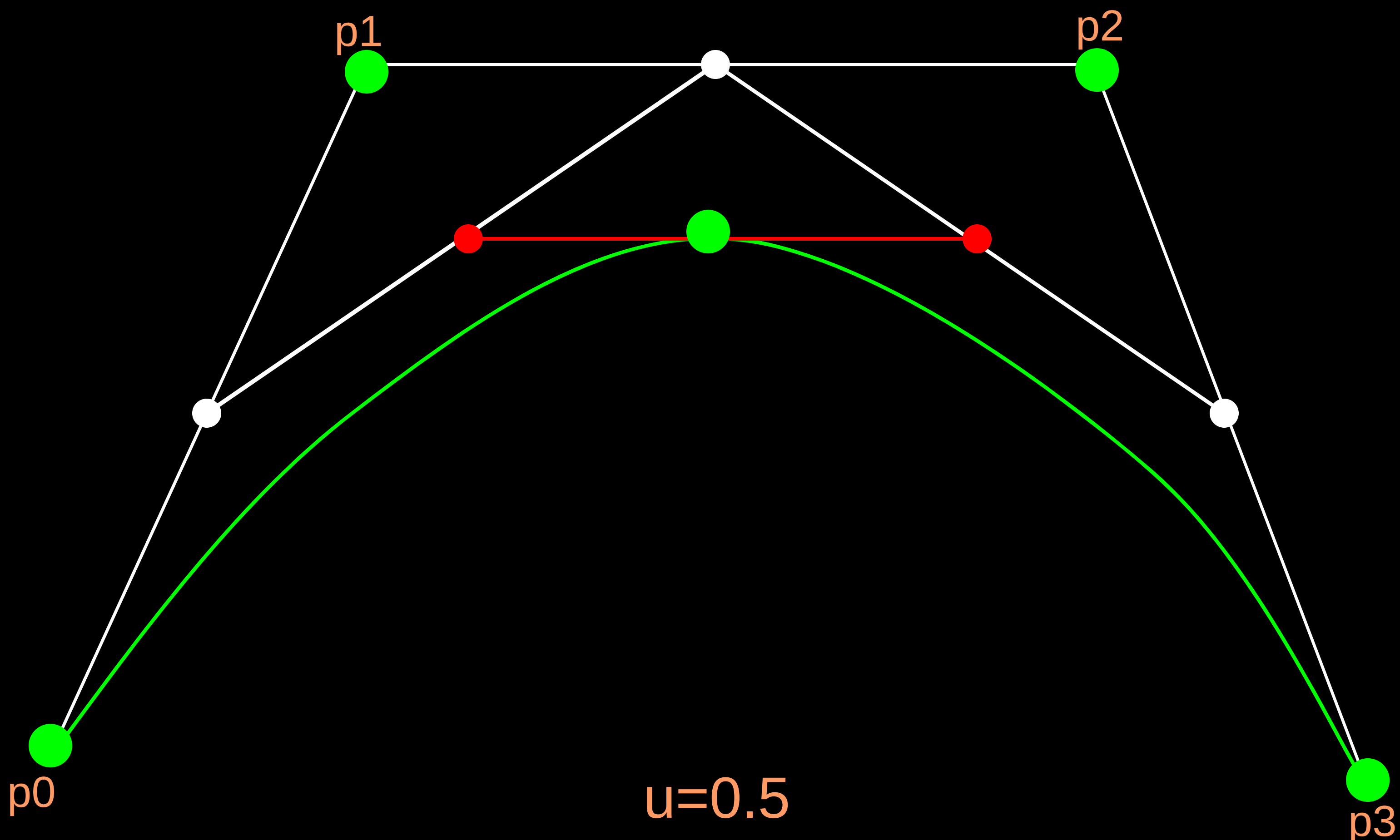
```
DeCasteljau(float p[N][3], float u) {

    for (int n = N-1; n >= 1; --n) {
        for (int j = 0; j < n; ++j) {
            p[j][0] += u*(p[j+1][0]-p[j][0]);
            p[j][1] += u*(p[j+1][1]-p[j][1]);
            p[j][2] += u*(p[j+1][2]-p[j][2]);
        }
    }
    //(p[0][0], p[0][1], p[0][2]) saves the result
}
```

Subdivision

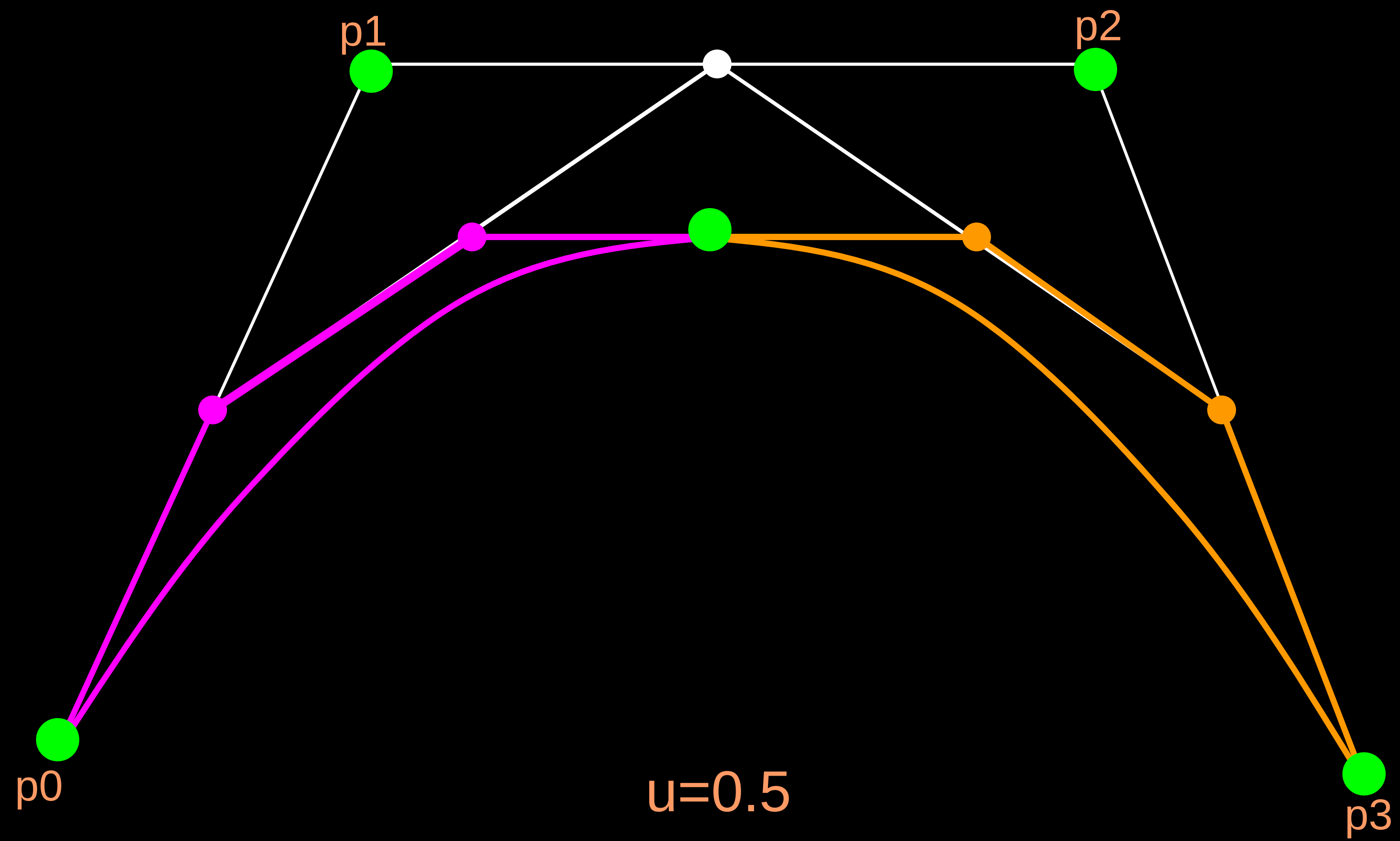
- Given a Bezier curve defined by $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$, ..., $\mathbf{P}_n$
- we want to find *two* sets of n+1 control points $\mathbf{Q}_0$, $\mathbf{Q}_1$, $\mathbf{Q}_2$, ..., $\mathbf{Q}_n$ and $\mathbf{R}_0$, $\mathbf{R}_1$, $\mathbf{R}_2$, ..., $\mathbf{R}_n$ such that
  - the Bézier curve defined by $\mathbf{Q}_i$'s is the piece of the original Bézier curve on $[0,u]$
  - the Bézier curve defined by $\mathbf{R}_i$'s is the piece of the original Bézier curve on $[u,1]$

Bezier Curve Subdivision



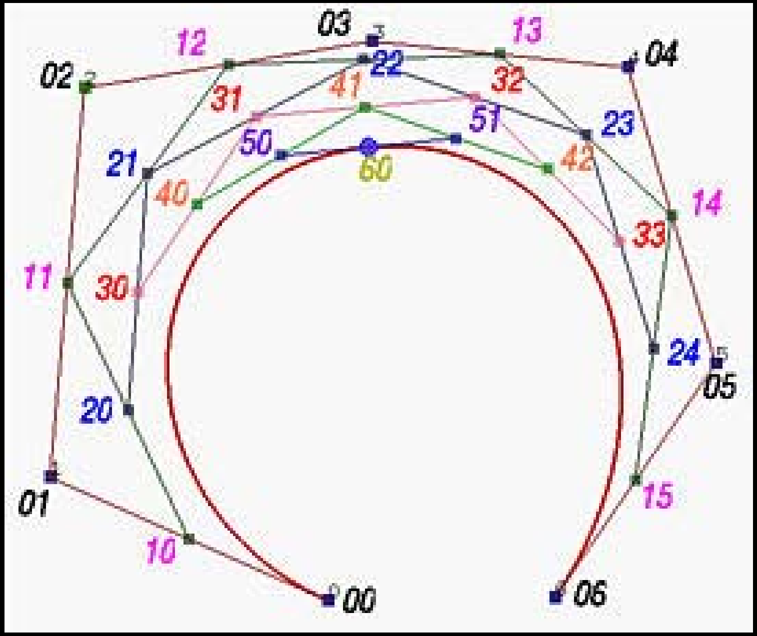
http://www1.cs.columbia.edu/~cs4160/html06f/

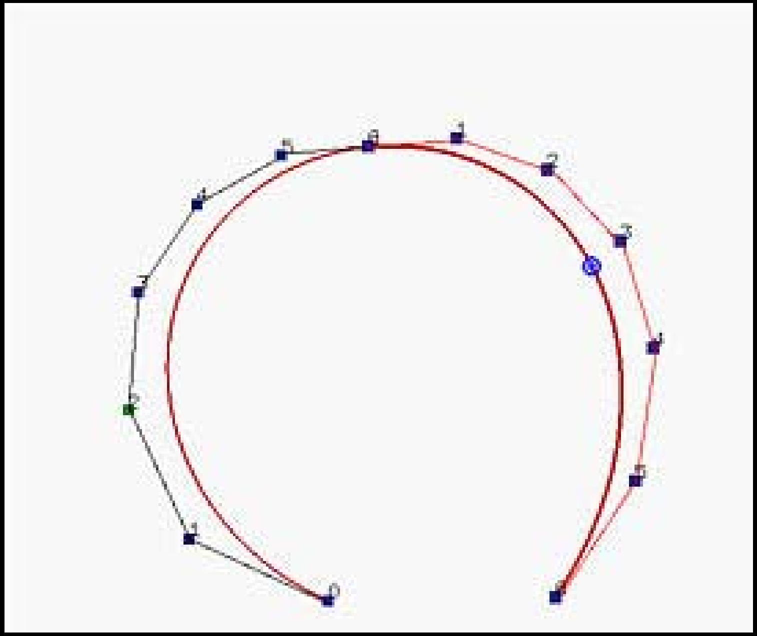
Bezier Curve Subdivision



http://www1.cs.columbia.edu/~cs4160/html06f/

A 6th degree subdivision example





http://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/Bezier/bezier-sub.html

Bezier Curve Subdivision

- Why is subdivision useful?
  - Collision/intersection detection
    - Recursive search
  - Good for curve editing and approximation