FLAME UNIVERSITY

EVERLASTING learning

FUNDAMENTALS OF COMPUTER GRAPHICS (CSIT304)

BEZIER CURVES

CHIRANJOY CHATTOPADHYAY

Associate Professor,

FLAME School of Computation and Data Science

PARAMETRIC CURVES

- The coefficients of an equations to have geometric meaning
- To predict the change of shape if one or more coefficients are modified.
- To do so, a system that supports users to design curves must be
  - Intuitive
  - Flexible
  - Unified Approach
  - Invariant
  - Efficiency and Numerically Stability

CURVE DESIGN

- Layouts a set of control
- Edit (Add, Delete, Move) the control points and some other characteristics for modifying the shape of the curve.
- There are very geometric, intuitive and numerically stable algorithms for finding points on the curve without knowing the equation of the curve.
- Once you know curves, the surface counterpart is a few steps away.

BEZIER CURVE

BEZIER CURVES

- Different choices of basis functions give different curves
  - Choice of basis determines how the control points influence the curve
  - In Hermite case, two control points define endpoints, and two more define parametric derivatives
- For Bezier curves, two control points define endpoints, and two control the tangents at the endpoints in a geometric way

BEZIER CURVES

- The user supplies $d+1$ control points, $\boldsymbol{p}_i$

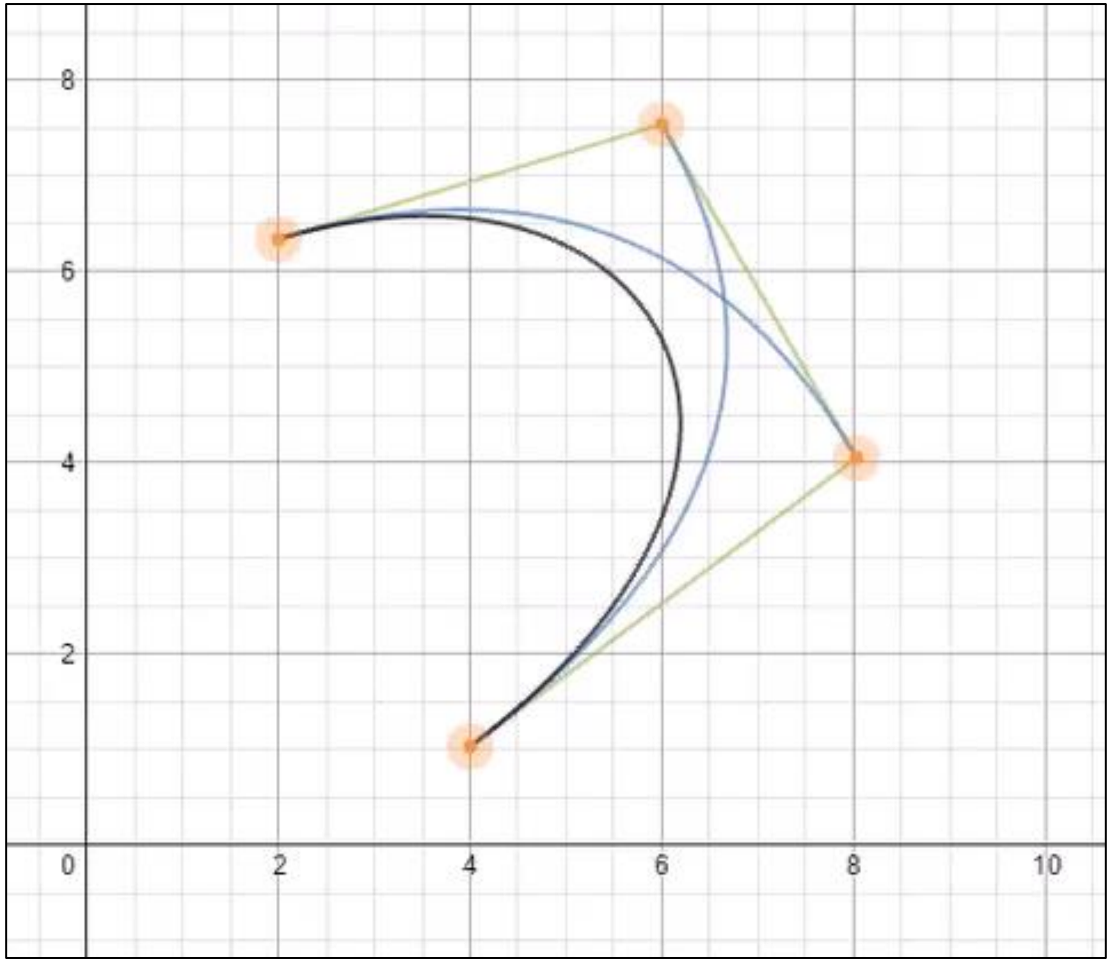
- Write the curve as:

$$\mathbf{x}(t) = \sum_{i=0}^{d} \mathbf{p}_i B_i^d(t) \qquad\qquad B_i^d(t) = \binom{d}{i} t^i (1-t)^{d-i}$$

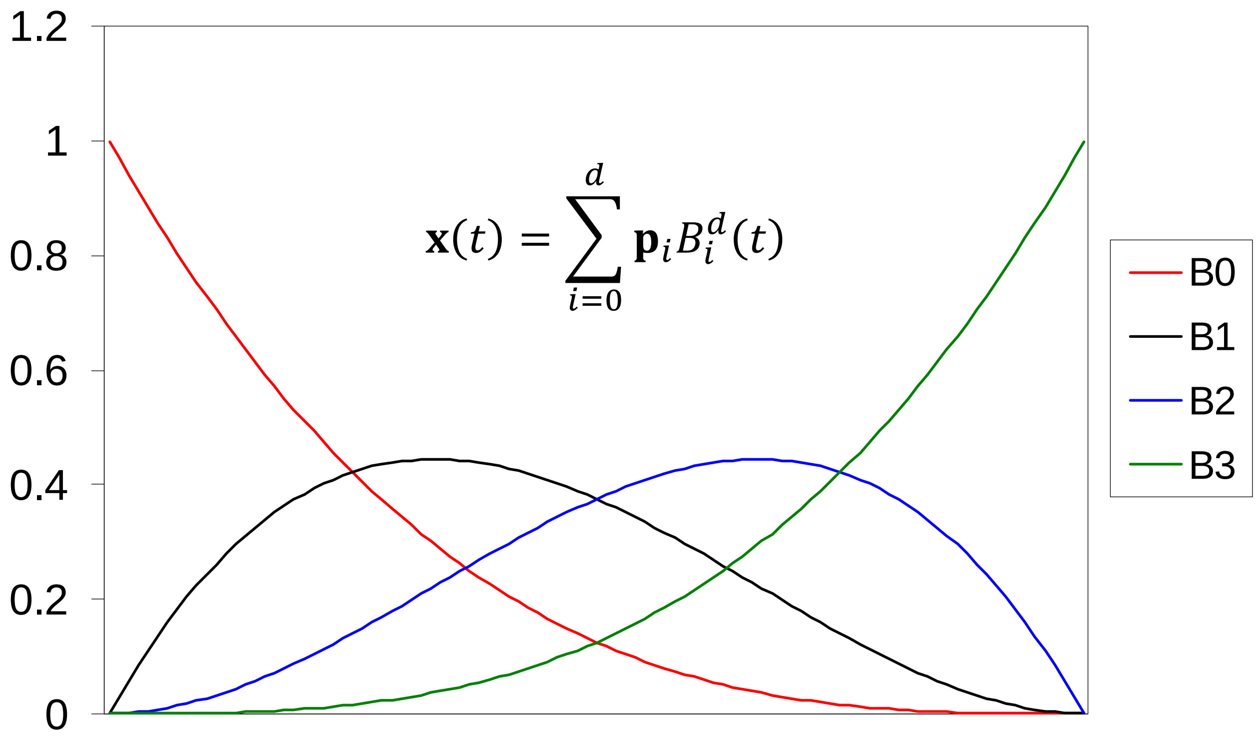
- The functions $\boldsymbol{B_i^d}$ are the *Bernstein polynomials* of degree $\boldsymbol{d}$

- This equation can be written as a matrix equation also

EXAMPLE

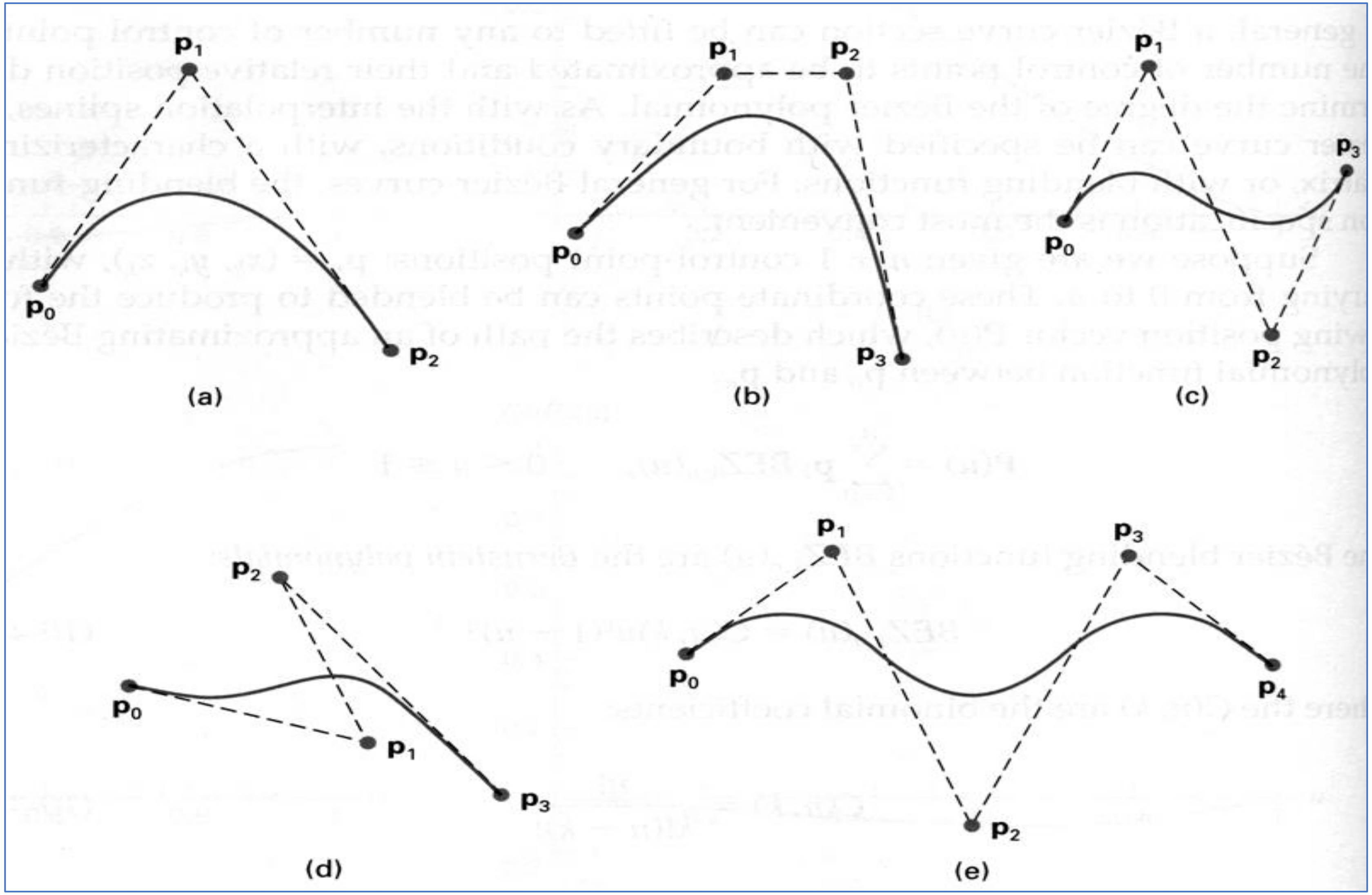


BEZIER BASIS FUNCTIONS FOR D=3



$$B_0^3(t) = \binom{3}{0} t^0 (1-t)^{3-0} = \frac{3!}{0!\,(3-0)!} \times 1 \times (1-t)^3 = (1-t)^3$$

SOME BEZIER CURVES

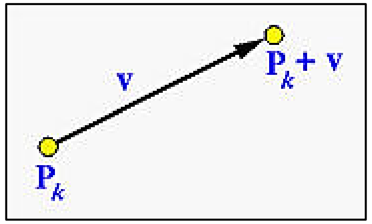


BEZIER CURVE PROPERTIES

- The first and last control points are interpolated
- Affine invariant
- The curve lies entirely within the convex hull of its control points
- The tangent to the curve at the first control point is along the line joining the first and second control points
- The tangent at the last control point is along the line joining the second last and last control points

MOVEMENT OF THE CONTROL POINT

- $\boldsymbol{P}_k$ is moved to a new position $\boldsymbol{P}_{k\ +\ v}$



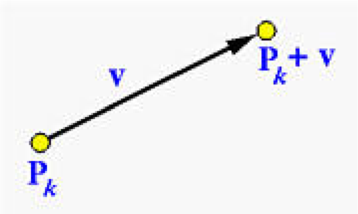
$$P(t) = \sum_{i=0}^{n} P_i B_i^n(t) \quad P_0, P_1, \dots, P_k, \dots, P_n \quad \Rightarrow \quad P_0, P_1, \dots, P_{k+v}, \dots, P_n$$

What can we say for the transformed curve $\boldsymbol{P'(t)}$?

*The corresponding point of t on the new curve is obtained by **translating** the corresponding point of t on the original curve in the **direction of v with a distance $B_i^n(t)$***

Show Animation

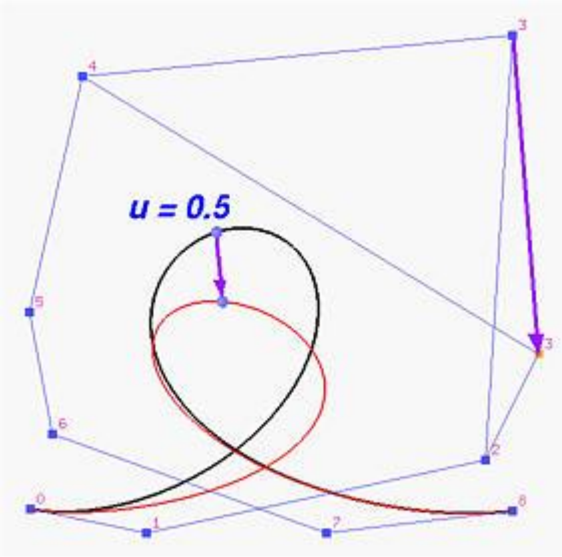
MOVING CONTROL POINTS



Since the new Bézier curve is defined by $\mathbf{P}_0$, $\mathbf{P}_1$. ..., $\mathbf{P}_k+\mathbf{v}$, ..., $\mathbf{P}_n$, its equation $\mathbf{D}(u)$ is

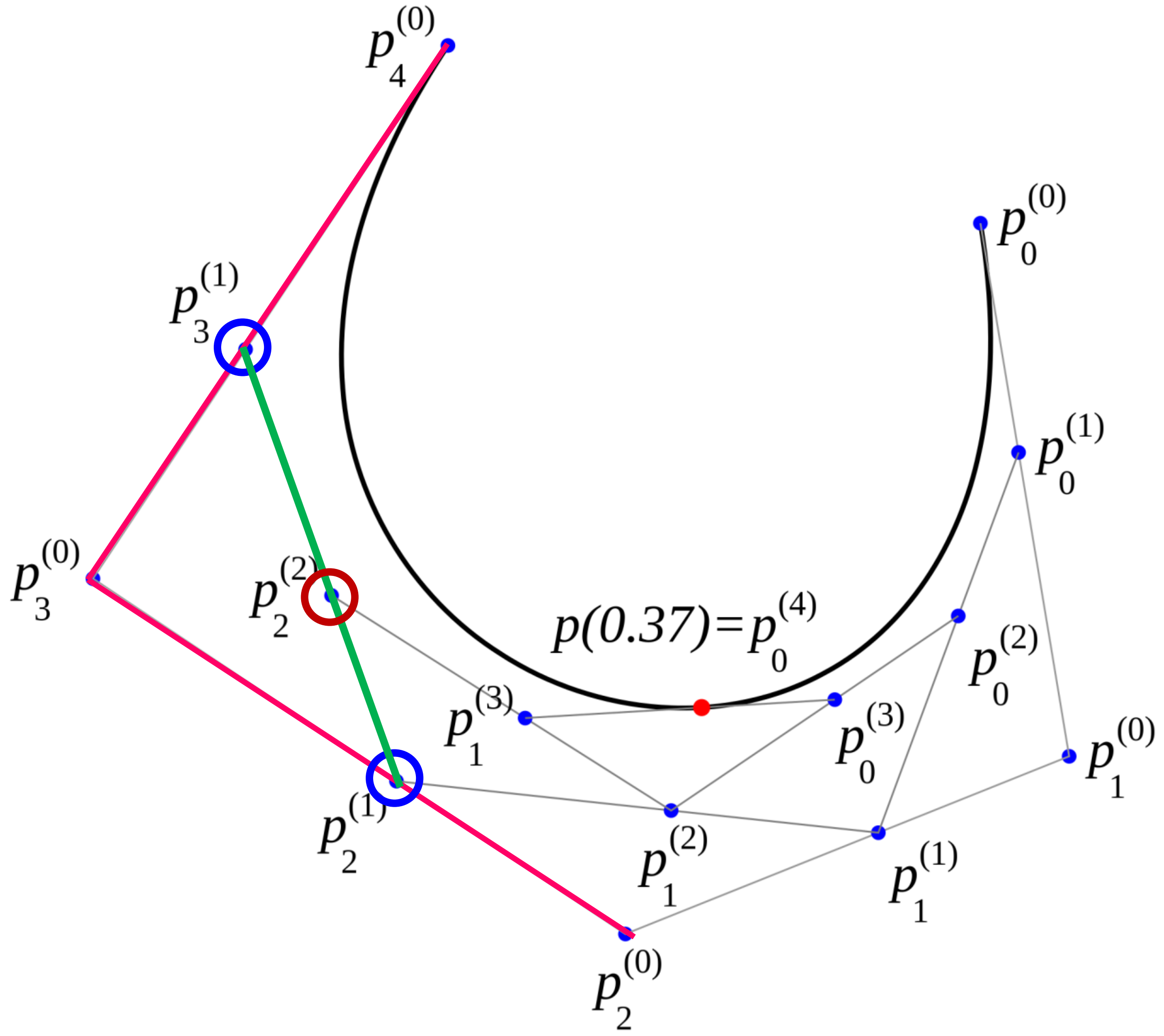
$$\mathbf{C}(u) = \sum_{i=0}^{n} B_{n,i}(u)\mathbf{P}_i$$

$$\begin{aligned}\mathbf{D}(u) &= \sum_{i=0}^{k-1} B_{n,i}(u)\mathbf{P}_i + B_{n,k}(u)(\mathbf{P}_k + \mathbf{v}) + \sum_{i=k+1}^{n} B_{n,i}(u)\mathbf{P}_i \\ &= \sum_{i=0}^{n} B_{n,i}(u)\mathbf{P}_i + B_{n,k}(u)\mathbf{v} \\ &= \mathbf{C}(u) + B_{n,k}(u)\mathbf{v}\end{aligned}$$



The corresponding point of *u* on the new curve is obtained by translating the corresponding point of *u* on the original curve in the direction of v with a distance of $|B_{n,k}(u)\mathbf{v}|$.

DE CASTELJAU'S ALGORITHM: ILLUSTRATION



Play Animation

DE CASTELJAU'S ALGORITHM

- Task is to find the point $\boldsymbol{P}(t)$ on the curve for a particular t

Input: Array $P[0:n]$ of $n+1$ points and real number t in $[0,1]$

Output: Point on curve, $P(t)$

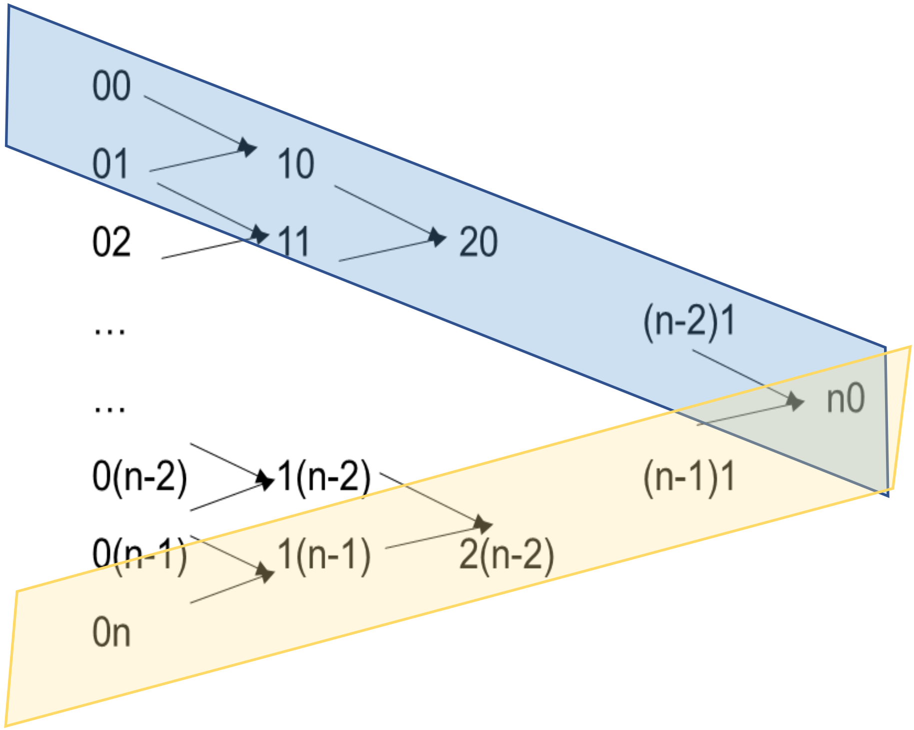
Working: Point array $Q[0:n]$

```
for i = 0 to n do
      Q[i] = P[i]; // save input
for r = 1 to n do
      for i = 0 to n - r do
            Q[i] = (1 - u)Q[i] + u Q[i + 1];
return Q[0];
```

DE CASTELJAU'S ALGORITHM: RECURRENCE

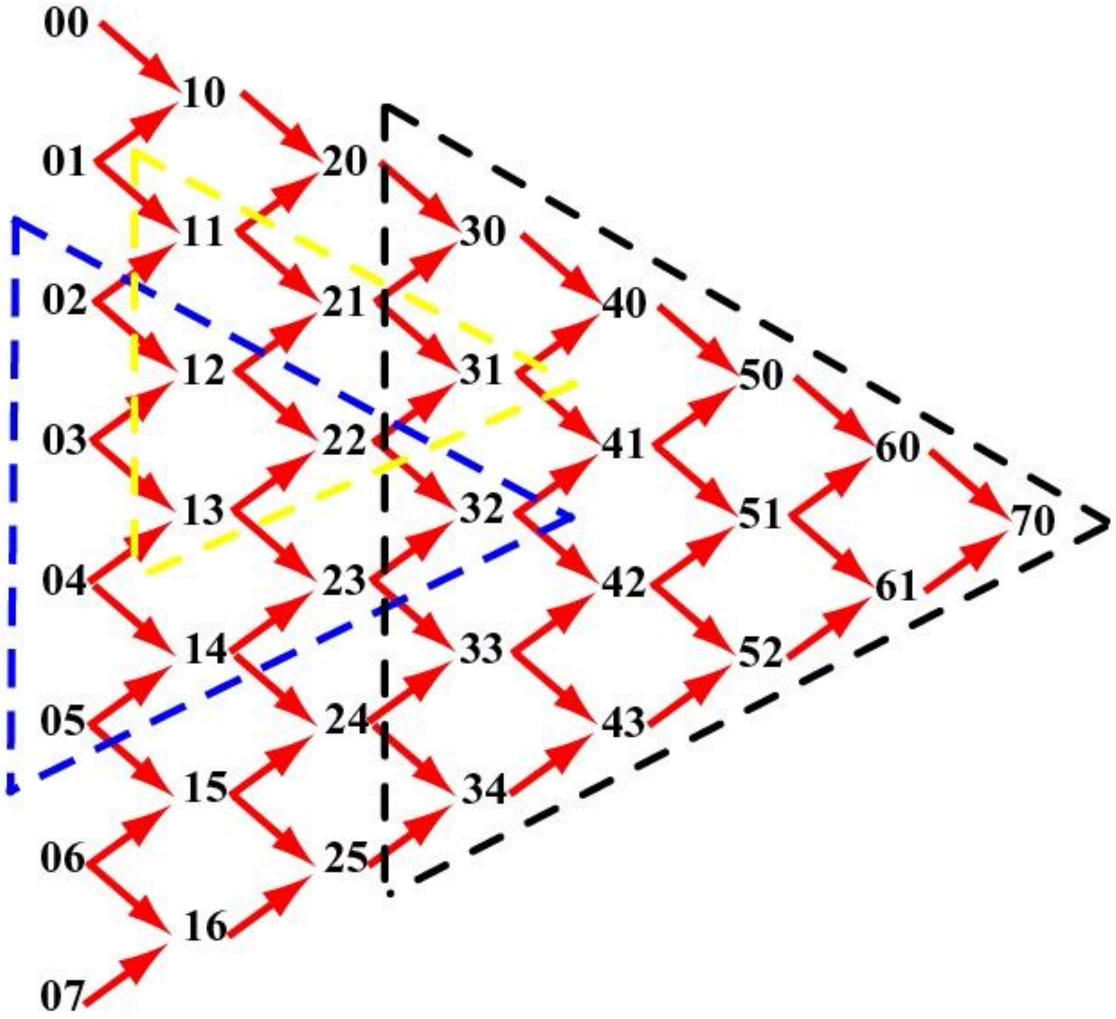
$$P_{i,j} = (1-u)P_{i-1,j} + uP_{i-1,j+1}$$

$$\text{for } i = 1, \dots, n; j = 1, \dots, n-i$$




```
function deCasteljau(i,j)
begin
if i = 0 then
        return P0,j
else
return (1-u)*
deCasteljau(i-1,j) + u* deCasteljau(i-1,j+1)
end
```

DE CASTELJAU'S ALGORITHM



SUBDIVIDING A BÉZIER CURVE

- Cut a given Bézier curve at $\boldsymbol{P}(t)$ for some t into two curve segments

- Each of which is still a Bézier curve.

- What happens to the control point?

- What about the degree of the resultant curves?

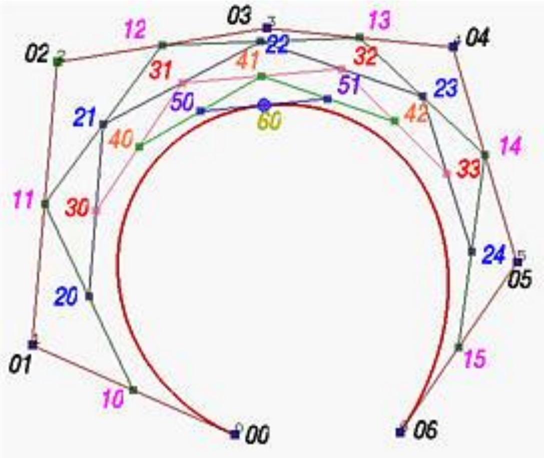
PROBLEM STATEMENT

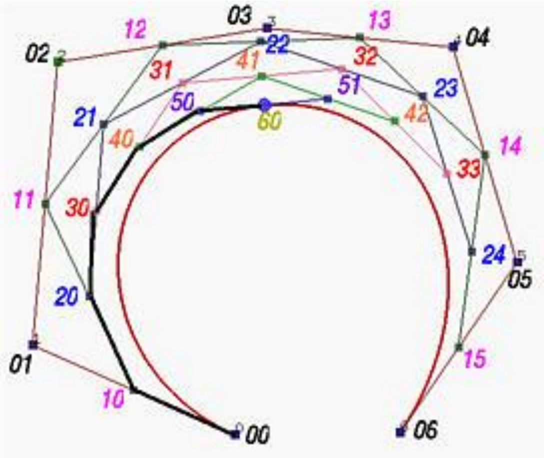
Given a set of $\boldsymbol{n} + \mathbf{1}$ **control points** $\boldsymbol{P_0}, \boldsymbol{P_1}, \boldsymbol{P_2}, \dots, \boldsymbol{P_n}$ and a parameter value $\boldsymbol{t}$ **in** $[\mathbf{0}, \mathbf{1}]$,

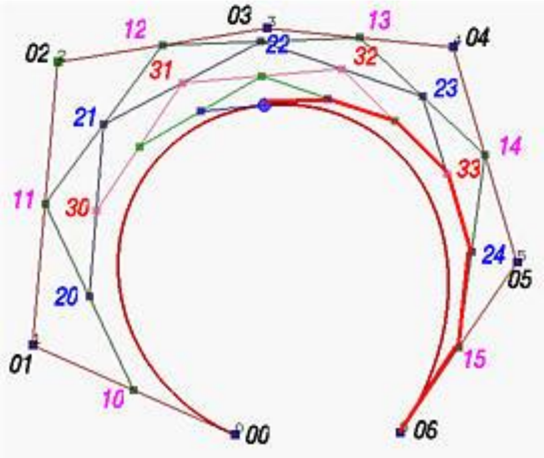
We want to find **two sets of** $\boldsymbol{n} + \mathbf{1}$ **control points** $\boldsymbol{Q_0}, \boldsymbol{Q_1}, \boldsymbol{Q_2}, \dots, \boldsymbol{Q_n}$ and $\boldsymbol{R_0}, \boldsymbol{R_1}, \boldsymbol{R_2}, \dots, \boldsymbol{R_n}$ such that

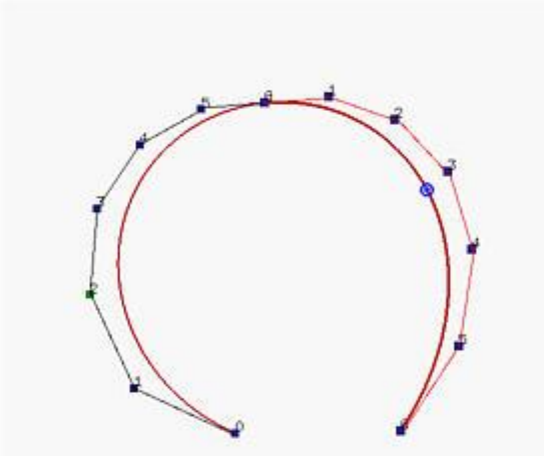
The Bézier curve defined by $\boldsymbol{Q_i}'\boldsymbol{s}$ (resp., $\boldsymbol{R_i}'\boldsymbol{s}$) is the piece of the original Bézier curve on $[\mathbf{0}, \boldsymbol{t}]$ (resp., $[\boldsymbol{t}, \mathbf{1}]$).

SUBDIVIDING BEZIER CURVE



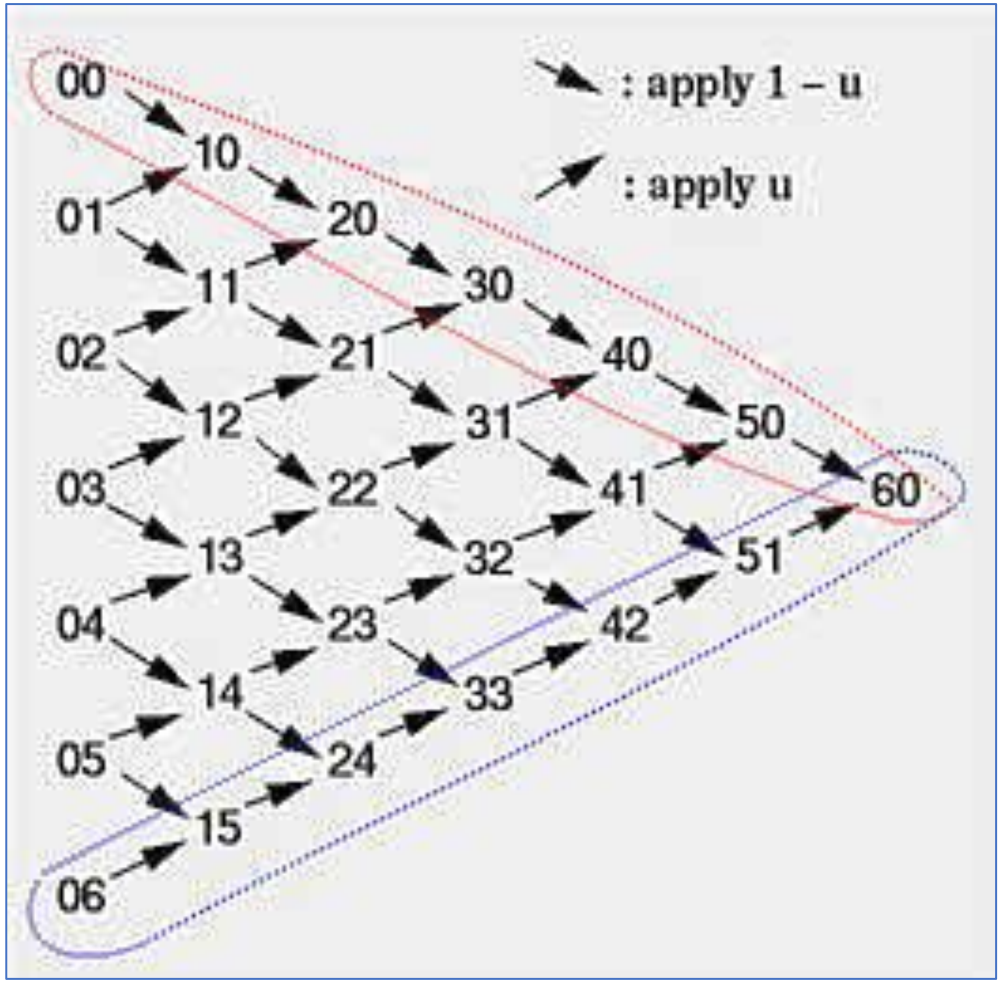






Play Animation

SUBDIVIDING BEZIER CURVE



- For a given u, it takes n iterations to compute C(u).
- Collect the first and the last points on each column
- The collection of the first (resp., last) points gives the subdivision corresponding to the piece of the original curve defined on [0,u] (resp., [u,1]).
- The control points of the first curve segments
  - The top edge in the direction of the arrows
- The control points of the first curve segments
  - The lower edge in the reversed direction of the arrows

DEGREE ELEVATION

- In an application it is desired that all involved curves to have the same degree
- Increase the degree of a Bézier curve *without* changing its shape.
- Degree elevation
- Existing degree = n; elevated degree = n+1
- Existing control points = $\{P_0, \dots, P_n\}$; New control points = $\{Q_0, \dots, Q_{n+1}\}$, such that $P_0 = Q_0$ & $P_n = Q_{n+1}$

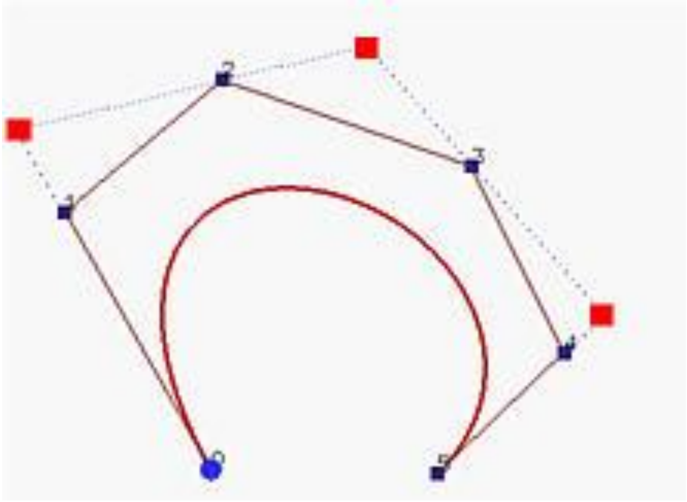
$$\boxed{\boldsymbol{Q_i = \frac{i}{n+1} P_{i-1} + \left(1 - \frac{i}{n+1}\right) P_i ; 1 \le i \le n}}$$

DEGREE ELEVATION OF A BÉZIER CURVE

$$\boxed{Q_i = \frac{i}{n+1} P_{i-1} + \left(1 - \frac{i}{n+1}\right) P_i ;\ 1 \le i \le n}$$

$$\begin{aligned}
\mathbf{Q}_1 &= \frac{1}{n+1}\mathbf{P}_0 + \left(1 - \frac{1}{n+1}\right)\mathbf{P}_1 \\
\mathbf{Q}_2 &= \frac{2}{n+1}\mathbf{P}_1 + \left(1 - \frac{2}{n+1}\right)\mathbf{P}_2 \\
\mathbf{Q}_3 &= \frac{3}{n+1}\mathbf{P}_2 + \left(1 - \frac{3}{n+1}\right)\mathbf{P}_3 \\
&\vdots \\
\mathbf{Q}_{n-1} &= \frac{n-1}{n+1}\mathbf{P}_{n-2} + \left(1 - \frac{n-1}{n+1}\right)\mathbf{P}_{n-1} \\
\mathbf{Q}_n &= \frac{n}{n+1}\mathbf{P}_{n-1} + \left(1 - \frac{n}{n+1}\right)\mathbf{P}_n
\end{aligned}$$

- Each leg of the original polyline contains exactly one new control point.
- This computation is very similar to that of de Casteljau's algorithm, though.
- Once the new set of control points is obtained, the original set can be discarded.



FLAME UNIVERSITY | EVERLASTING learning

THANK YOU