

Parametric Curves Bezier Models



Modeling for computer graphics
Faramorz Samavati

Goal/outline

- ❖ Goals of this and the next lecture
 - To determine the key property of curves for computer Graphics
 - To introduce interactive parametric curves starting with Bezier curves
 - To give Bezier curve definition
 - To provide a good algorithm for Bezier curves
- ❖ Importance of this lecture
 - it is used in many software
 - can be easily extended to Bezier patches
 - ready to use for generic surfaces
 - can be used for free-form space deformation
 - It is not the best interactive curve method
- ❖ Outline
 - [Polynomial curves](#)
 - Bezier curve definition and basic properties
 - deCatseljau algorithm

Modeling for Computer
Graphics
Faramorz Samavati



Polynomial curves

- ❖ Polynomials are fundamental mathematical objects
- ❖ Easy and efficient to compute
- ❖ standard representation $P(u) = a_0 + a_1u + \dots + a_nu^n$
- ❖ Polynomial curves of degree one, two and three

$$Q(u) = P_0 + P_1u + \dots + P_nu^n$$

where

$$P_i = \begin{bmatrix} a_i \\ b_i \end{bmatrix}$$



User input for a polynomial curve

- ❖ Formula!!! No way
- ❖ Giving 2D(or 3D) points as P_i “a good idea”
- ❖ P_i : point => arbitrary combination of them is not possible. Why?
- ❖ Affine combination
- ❖ It is not possible to satisfy this condition by this set of polynomials

Bernstein basis functions

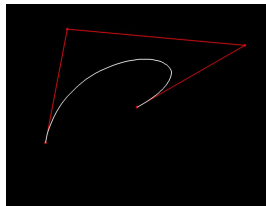
- ❖ Terms in the expansion of: $1 \equiv ((1 - u) + u)^n$
- ❖ Second degree polynomials
- ❖ Third degree polynomials
- ❖ Resulting curves

Bezier Model

A major breakthrough in geometric modeling

- ❖ P. Bezier,
- ❖ 1974, Unisurf in Renault automobile designing.
- ❖ Mathematical definition

Q(u)



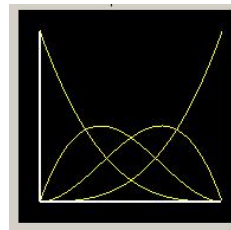
- ❖ CAD and graphics software
- ❖ Postscript
- ❖ METAFONT

Bezier Curve Definition

$$\ast Q(u) = \sum_{i=0}^d P_i B_{i,d}(u) \quad , 0 \leq u \leq 1$$

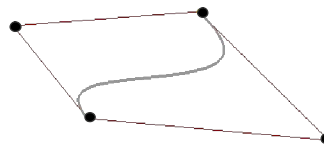
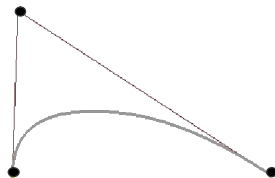
$$\ast B_{i,d}(u) = \binom{d}{i} u^i (1-u)^{d-i} \text{ Berenstein polynomial of degree } d$$

$\ast P_i$: Control points (inputs)



Bezier Curve

- ❖ Bezier curve of order 1
- ❖ Bezier curve of order 2
- ❖ Bezier curve of order 3
- ❖ Basic properties





Implementation (brute force)

```
Input P[i], d
//P[i]: control point, d : degree , u : parameter
for u=0 to 1 step 0.01
  for i=0 to d
    b = Bernstein( i , d , u )
    q = q + b * P[i]
  end
  plot(q);
end
```

❖ What is wrong here?

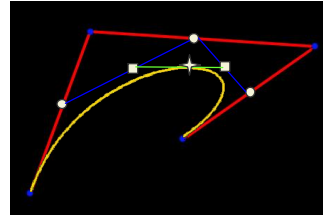


deCasteljau algorithm

- ❖ A bottom-up approach: dynamic programming
- ❖ It avoids redundant computations of the basis functions
- ❖ Demonstration by the second degree Bezier curve
- ❖ Cubic Bezier curve
- ❖ General case

Geometric Interpretation

- ❖ Q(0.5): bisection method
- ❖ example:



$$u = \frac{1}{2}$$

$$\begin{array}{l}
 j=3 \\
 j=2 \\
 j=1 \\
 j=0
 \end{array}
 \begin{array}{l}
 \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ 2 \\ 8 \\ 2 \\ 4 \\ 0 \end{array} \right] \\
 \left[\begin{array}{c} 0 \\ 1 \\ 4 \\ 2 \\ 6 \\ 1 \end{array} \right] \\
 \left[\begin{array}{c} 2 \\ 1.5 \\ 5 \\ 1.5 \end{array} \right] \\
 \left[\begin{array}{c} 3.5 \\ 1.5 \end{array} \right]
 \end{array}
 \begin{array}{l}
 \leftarrow \\
 \leftarrow \\
 \leftarrow \\
 \leftarrow \\
 \leftarrow \\
 \leftarrow \\
 \leftarrow \\
 \leftarrow
 \end{array}
 \begin{array}{l}
 \left[\begin{array}{c} 0 \\ 1 \\ 4 \\ 2 \\ 6 \\ 1 \end{array} \right] \\
 \left[\begin{array}{c} 2 \\ 1.5 \\ 5 \\ 1.5 \end{array} \right] \\
 \left[\begin{array}{c} 3.5 \\ 1.5 \end{array} \right]
 \end{array}
 \begin{array}{l}
 \leftarrow \\
 \leftarrow \\
 \leftarrow \\
 \leftarrow
 \end{array}
 \begin{array}{l}
 \left[\begin{array}{c} 2 \\ 1.5 \\ 5 \\ 1.5 \end{array} \right] \\
 \left[\begin{array}{c} 3.5 \\ 1.5 \end{array} \right]
 \end{array}
 \begin{array}{l}
 \leftarrow \\
 \leftarrow
 \end{array}
 \begin{array}{l}
 \left[\begin{array}{c} 3.5 \\ 1.5 \end{array} \right]
 \end{array}
 \begin{array}{l}
 \leftarrow \\
 \leftarrow
 \end{array}
 \begin{array}{l}
 \left[\begin{array}{c} 3.5 \\ 1.5 \end{array} \right]
 \end{array}
 \end{array}
 \begin{array}{l}
 Q(\frac{1}{2})
 \end{array}
 \begin{array}{l}
 \leftarrow u \\
 \leftarrow 1-u
 \end{array}$$



Pseudo Code

```

Input P[j], d, u
//P[j]: control point, d: degree, u: parameter
//output will be Q(u)
for i = 1 to d
  for j = 0 to d-i
    P[j] = (1-u)*P[j] + u*P[j+1]
  end
end
Output ???

```

- ❖ Column-by column updating rule
- ❖ Trace it for the following example
- ❖ Why is this algorithm correct?

$$\begin{array}{l}
 j=3 \\
 j=2 \\
 j=1 \\
 j=0
 \end{array}
 \begin{array}{l}
 \left[\begin{array}{c} 0 \\ 0 \\ 0 \\ 2 \\ 8 \\ 2 \\ 4 \\ 0 \end{array} \right] \\
 \left[\begin{array}{c} 0 \\ 1 \\ 4 \\ 2 \\ 6 \\ 1 \end{array} \right] \\
 \left[\begin{array}{c} 2 \\ 1.5 \\ 5 \\ 1.5 \end{array} \right] \\
 \left[\begin{array}{c} 3.5 \\ 1.5 \end{array} \right]
 \end{array}
 \begin{array}{l}
 \leftarrow \\
 \leftarrow \\
 \leftarrow \\
 \leftarrow \\
 \leftarrow \\
 \leftarrow \\
 \leftarrow \\
 \leftarrow
 \end{array}
 \begin{array}{l}
 \left[\begin{array}{c} 0 \\ 1 \\ 4 \\ 2 \\ 6 \\ 1 \end{array} \right] \\
 \left[\begin{array}{c} 2 \\ 1.5 \\ 5 \\ 1.5 \end{array} \right] \\
 \left[\begin{array}{c} 3.5 \\ 1.5 \end{array} \right]
 \end{array}
 \begin{array}{l}
 \leftarrow \\
 \leftarrow \\
 \leftarrow \\
 \leftarrow
 \end{array}
 \begin{array}{l}
 \left[\begin{array}{c} 2 \\ 1.5 \\ 5 \\ 1.5 \end{array} \right] \\
 \left[\begin{array}{c} 3.5 \\ 1.5 \end{array} \right]
 \end{array}
 \begin{array}{l}
 \leftarrow \\
 \leftarrow
 \end{array}
 \begin{array}{l}
 \left[\begin{array}{c} 3.5 \\ 1.5 \end{array} \right]
 \end{array}
 \begin{array}{l}
 \leftarrow \\
 \leftarrow
 \end{array}
 \begin{array}{l}
 \left[\begin{array}{c} 3.5 \\ 1.5 \end{array} \right]
 \end{array}
 \end{array}
 \begin{array}{l}
 Q(u)
 \end{array}
 \begin{array}{l}
 \leftarrow \\
 \leftarrow
 \end{array}
 \begin{array}{l}
 \left[\begin{array}{c} 3.5 \\ 1.5 \end{array} \right]
 \end{array}
 \end{array}
 \begin{array}{l}
 (1-u)^3 \begin{bmatrix} 4 \\ 0 \end{bmatrix} = B_{0,3}(u) \begin{bmatrix} 4 \\ 0 \end{bmatrix}
 \end{array}$$