Topics in Computer Graphics
Chap 5: The Bernstein Form of a Bézier Curve

fall, 2011

University of Seoul
School of Computer Science
Minho Kim

Table of contents

Bernstein Polynomials

- Useful when expressing Bézier curves *explicitly*.
- Definition:

$$B_i^n(t) := \binom{n}{i} t^i (1-t)^{n-i},$$

where the *binomial coefficient* is defined as

$$\binom{n}{i} := \begin{cases} \frac{n!}{i!(n-i)!} & \text{if } 0 \leqslant i \leqslant n \\ 0 & \text{else.} \end{cases}$$

Bernstein Polynomials: Properties

- Recursive relation

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t)$$

with

$$B_0^0(t) \equiv 1(t) \quad \text{and} \quad B_j^n(t) \equiv 0(t) \quad \text{for } j \notin 0, \ldots, n.$$

- *Partition of unity*

$$\sum_{j=0}^{n} B_j^n(t) \equiv 1(t).$$

* Binomial theorem

Bézier Curves Expressed by Bernstein Polynomials

$$\mathbf{b}(t) = \mathbf{b}[t^{<t>}] = \sum_{i=0}^{n} \mathbf{b}_i B_i^n(t)$$

Can be proved by Leibniz formula

- Intermediate de Casteljau point

$$\begin{aligned} \mathbf{b}_i^r(t) &= \mathbf{b}[0^{<n-r-i>}, t^{<r>}, 1^{<i>}] \\ &= \sum_{j=0}^{r} \mathbf{b}[0^{<n-r-i>}, 1^{<i>}, 0^{<r-j>}, 1^{<j>}] B_j^r(t) \\ &= \sum_{j=0}^{r} \mathbf{b}[0^{<n-i-j>}, 1^{<i+j>}] B_j^r(t) = \sum_{j=0}^{r} \mathbf{b}_{i+j} B_j^r(t) \end{aligned} \tag{4.10}$$

- Bézier curve in terms of intermediate points

$$\begin{aligned} \mathbf{b}^n(t) = \mathbf{b}[t^{<n-r>}, t^{<r>}] &= \sum_{i=0}^{n-r} \mathbf{b}[t^{<r>}, 1^{<i>}, 0^{<n-r-i>}] B_i^{n-r}(t) \\ &= \sum_{i=0}^{n-r} \mathbf{b}_i^r(t) B_i^{n-r}(t) \end{aligned}$$

Properties of Bézier Curves

- Affine invariance
- Invariance under affine parameter transformations
- Convex hull property
- Endpoint interpolation
- Symmetry
- Invariance under barycentric combinations
- Linear precision
- Pseudolocal control

→ Can be proved algebraically using Bernstein polynomials.

Properties of Bézier Curves (cont'd)

- Affine invariance
  $\leftarrow$ Any point on a Bézier curve is a barycentric combination of its control points. (Why?)

$$\sum_{i=0}^{n} B_i^n(t) \equiv 1(t)$$

- Invariance under affine parameter transformations

$$\sum_{i=0}^{n} \mathbf{b}_i B_i^n(t) = \sum_{i=0}^{n} \mathbf{b}_i B_i^n\left(\frac{u-a}{b-a}\right)$$

- Convex hull property
- Endpoint interpolation

$$B_i^n(0) = \delta_{i,0} \quad B_i^n(1) = \delta_{i,n}$$

  * Kronecker delta function

Properties of Bézier Curves (cont'd)

- Symmetry

$$\sum_{j=0}^{n} \mathbf{b}_j B_j^n(t) = \sum_{j=0}^{n} \mathbf{b}_{n-j} B_j^n(1-t) \leftarrow B_j^n(t) = B_{n-j}^n(1-t)$$

- Invariance under barycentric combinations ($\alpha + \beta = 1$)

$$\sum_{j=0}^{n} (\alpha \mathbf{b}_j + \beta \mathbf{c}_j) B_j^n(t) = \alpha \sum_{j=0}^{n} \mathbf{b}_j B_j^n(t) + \beta \sum_{j=0}^{n} \mathbf{c}_j B_j^n(t)$$

  $\rightarrow$ tensor-product surfaces

- Linear precision

$$\sum_{j=0}^{n} \frac{j}{n} B_j^n(t) = t$$

- Pseudolocal control $\operatorname*{argmax}_{0 \leqslant t \leqslant 1} B_j^n(t) = j/n$

The Derivative of a Bézier Curve

$$\dot{\mathbf{x}}(t) = \frac{d\mathbf{x}(t)}{dt} = n\mathbf{b}[t^{<n-1>}, \vec{1}]$$

- First interpretation

$$= n \sum_{j=0}^{n} \mathbf{b}[1^{<j>}, 0^{<n-1-j>}, \vec{1}] B_j^{n-1}(t)$$

$$= n \sum_{j=0}^{n} \left(\mathbf{b}[1^{<j+1>}, 0^{<n-(j+1)>}] - \mathbf{b}[1^{<j>}, 0^{<n-j>}]\right) B_j^{n-1}(t)$$

$$(\mathbf{b}[\vec{1}, *] = \mathbf{b}[1, *] - \mathbf{b}[0, *])$$

$$= n \sum_{j=0}^{n-1} (\mathbf{b}_{j+1} - \mathbf{b}_j) B_j^{n-1}(t)$$

$$= n \sum_{j=0}^{n-1} \Delta\mathbf{b}_j B_j^{n-1}(t) \qquad (\Delta\mathbf{b}_j := \mathbf{b}_{j+1} - \mathbf{b}_j \in \mathbb{R}^3)$$

  - Bézier curve where its coefficients are vectors, not points.
    → *hodograph*

The Derivative of a Bézier Curve (cont'd)

- Second interpretation

$$
\begin{aligned}
n\mathbf{b}[t^{<n-1>}, \vec{1}] &= n\left(\mathbf{b}[t^{<n-1>}, 1] - \mathbf{b}[t^{<n-1>}, 0]\right) \\
&= n\left(\mathbf{b}_1^{n-1}(t) - \mathbf{b}_0^{n-1}(t)\right) \qquad (4.10)
\end{aligned}
$$

Higher Derivatives

$$\frac{d^r \mathbf{x}(t)}{dt^r} = \frac{n!}{(n-r)!}\mathbf{b}[t^{<n-r>}, \vec{1}^{<r>}]$$

$$= \frac{n!}{(n-r)!}\sum_{j=0}^{n-r} \Delta^r \mathbf{b}_j B_j^{n-r}(t)$$

where the *iterated forward difference operator* Δ^r is defined recursively as

$$\Delta^r \mathbf{b}_j = \Delta^{r-1}\mathbf{b}_{j+1} - \Delta^{r-1}\mathbf{b}_j$$

of which explicit form is

$$\Delta^r \mathbf{b}_i = \sum_{j=0}^{r} \binom{r}{j} (-1)^{r-j}\mathbf{b}_{i+j}.$$

Higher Derivatives (cont'd)

$$\frac{d^r}{dt^r}\mathbf{b}^n(0) = \frac{n!}{(n-r)!}\Delta^r \mathbf{b}_0$$
$$\frac{d^r}{dt^r}\mathbf{b}^n(1) = \frac{n!}{(n-r)!}\Delta^r \mathbf{b}_{n-r}$$

- "The rth derivative of a Bézier curve at an endpoint depends only the $r+1$ Bézier points near (and including) that endpoint."

Bézier Curves in Monomial Form

$$\mathbf{b}^n(t) = \sum_{j=0}^{n} \binom{n}{j} \Delta^j \mathbf{b}_0 t^j$$

- Can be derived from Taylor series

$$\mathbf{x}(t) = \sum_{j=0}^{n} \frac{1}{j!} \mathbf{x}^{(j)}(0) t^j$$

where

$$\mathbf{x}^{(j)}(t) = \frac{n!}{(n-j)!} \sum_{i=0}^{n-j} \Delta^j \mathbf{b}_i B_i^{n-j}(t).$$

- Numerically unstable

Domain Changes and Subdivision

- "Given a Bézier curve defined over $[0, 1]$, what are the control points $\{\mathbf{c}_i\}$ of the part defined over $[0, c]$?" (See (4.11) on p.52.)

$$
\begin{aligned}
\mathbf{x}_c(t) := \sum_{j=0}^{n} \mathbf{c}_j B_j^n(t) &= \mathbf{x}(ct) = \sum_{j=0}^{n} \mathbf{b}_j B_j^n(ct) \\
&= \sum_{j=0}^{n} \mathbf{b}_j \left(\sum_{i=0}^{n} B_j^i(c) B_i^n(t) \right) \qquad (6.22) \\
&= \sum_{i=0}^{n} \left(\sum_{j=0}^{i} \mathbf{b}_j B_j^i(c) \right) B_i^n(t)
\end{aligned}
$$

$$
\mathbf{c}_i = \mathbf{b}[0^{<n-i>}, c^{<i>}] = \mathbf{b}_0^i(c)
$$

$\rightarrow$ *Subdivision formula* for Bézier curves

Extrapolation

Control points $\{\mathbf{d}_i\}$ corresponding to an interval $[1, d]$

$$\mathbf{d}_j = \mathbf{b}[1^{<n-j>}, d^{<j>}] = \mathbf{b}^j_{n-j}(d)$$

- Numerically unstable for large d

Subdivision and Applications

- The control polygon converges to the curve *quickly* as the subdivision is repeated.
- Applications
  - Rendering a Bézier curve
  - Line-planar curve intersection test

Composite Bézier Curves

How to generate complex shapes using Bézier curves?

- Using high degree curves
  - Expensive to evaluate
  - Too smooth to generate complex shapes (Java applet)
- Using composite curves
  - Requires points properly positioned for continuity
    - C^0 continuity: two curves share the joint
    - G^1 continuity: the directions (not lengths) of tangent vectors at the joint are the same
    - C^1 continuity: both the directions & lengths of tangent vectors at the joint are the same

Continuity at the Joint

With $\mathbf{x}_-$ defined by $\mathbf{b}_0, \ldots, \mathbf{b}_3$ over $[a, b]$ and $\mathbf{b}_+$ defined by $\mathbf{b}_3, \ldots, \mathbf{b}_6$ over $[b, c]$, the following condition should be met for C^1 continuity:

$$\frac{3}{b-a}[\mathbf{b}_3 - \mathbf{b}_2] = \frac{3}{c-b}[\mathbf{b}_4 - \mathbf{b}_3]$$

$\rightarrow$ The ratio of the three points $\mathbf{b}_2, \mathbf{b}_3, \mathbf{b}_4$ is the same as the ratio of the three parameter values a, b, c.

- Continuities are guaranteed easily for B-splines. (Chap 8)

The Matrix Form of a Bézier Curve

A curve of the form

$$\mathbf{x}(t) = \sum_{j=0}^{n} \mathbf{c}_i C_i(t)$$

can be expressed as

$$\mathbf{x}(t) = \begin{bmatrix} \mathbf{c}_0 & \cdots & \mathbf{c}_n \end{bmatrix} \begin{bmatrix} C_0(t) \\ \vdots \\ C_n(t) \end{bmatrix}$$

where

$$\begin{bmatrix} C_0(t) \\ \vdots \\ C_n(t) \end{bmatrix} = M \begin{bmatrix} t^0 \\ \vdots \\ t^n \end{bmatrix} = \begin{bmatrix} m_{00} & \cdots & m_{0n} \\ \vdots & & \vdots \\ m_{n0} & \cdots & m_{nn} \end{bmatrix} \begin{bmatrix} t^0 \\ \vdots \\ t^n \end{bmatrix}$$

M: basis transformation between the basis polynomial $C_i(t)$ and the *monomial basis* t^i

The Matrix Form of a Bézier Curve (cont'd)

If $C_i(t) = B_i^n(t)$, (Bernstein polynomials),

$$m_{ij} = (-1)^{j-i} \binom{n}{j} \binom{j}{i}$$

Example (cubic case)

$$M = \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Again, monomial forms are not numerically stable.