Topics in Computer Graphics
Chap 4: The de Casteljau Algorithm

fall, 2011

University of Seoul
School of Computer Science
Minho Kim

Table of contents

Parabolas via Linear Interpolation

Let $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2 \in \mathbb{E}^3$ and $t \in \mathbb{R}$. Construct

$$\begin{aligned}
\mathbf{b}_0^1(t) &= (1-t)\mathbf{b}_0 + t\mathbf{b}_1 \\
\mathbf{b}_1^1(t) &= (1-t)\mathbf{b}_1 + t\mathbf{b}_2 \\
\mathbf{b}_0^2(t) &= (1-t)\mathbf{b}_0^1(t) + t\mathbf{b}_1^1(t)
\end{aligned}$$

which becomes
$\mathbf{b}^2 := \mathbf{b}^2(t) = \mathbf{b}_0^2(t) = (1-t)^2\mathbf{b}_0 + 2t(1-t)\mathbf{b}_1 + t^2\mathbf{b}_2.$

- $\mathbf{b}_0^2(t)$ traces out a *parabola* as t varies from $-\infty$ to ∞.
- Constructed by *repeated linear interpolation*.
- $\mathbf{b}^2(0) = \mathbf{b}_0$ and $\mathbf{b}^2(1) = \mathbf{b}_2$.
- $\mathrm{ratio}(\mathbf{b}_0, \mathbf{b}_0^1, \mathbf{b}_1) = \mathrm{ratio}(\mathbf{b}_1, \mathbf{b}_1^1, \mathbf{b}_2) = \mathrm{ratio}(\mathbf{b}_0^1, \mathbf{b}_0^2, \mathbf{b}_1^1) = t/(1-t)$
- *Affinely invariant* (Why?)
- Plane curve (Why?)
- *Three tangent theorem*

The de Casteljau Algorithm

- Generalization of parabola construction to a polynomial curve of arbitrary degree n

de Casteljau algorithm

Given: $\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{E}^3$ and $t \in \mathbb{R}$, set

$$\mathbf{b}_i^r(t) = (1-t)\mathbf{b}_i^{r-1}(t) + t\mathbf{b}_{i+1}^{r-1}(t), \quad \begin{cases} r = 1, \ldots, n \\ i = 0, \ldots, n-r \end{cases}$$

and $\mathbf{b}_i^0(t) = \mathbf{b}_i$. Then $\mathbf{b}_0^n(t)$ is the point with parameter value t on the *Bézier curve* $\mathbf{b}^n$, hence $\mathbf{b}^n(t) = \mathbf{b}_0^n(t)$.

The de Casteljau Algorithm (cont'd)

- Used to evaluate the point $\mathbf{b}^n(t)$ on the curve.
- *Bézier polygon* or *control polygon* $\mathbf{P}$ of $\mathbf{b}^n$
- *Bézier points* or *control points*
- Alternative notations
  $\mathbf{b}^n(t) = \mathbf{B}[\mathbf{b}_0, \ldots, \mathbf{b}_n; t] = \mathbf{B}[\mathbf{P}; t]$ or
  $\mathbf{b}^n = \mathbf{B}[\mathbf{b}_0, \ldots \mathbf{b}_n] = \mathbf{BP}$
- "The curve is the *Bernstein-Bézier approximation* to the control polygon."
- *de Casteljau scheme* (Example 4.1)

$$\begin{array}{llll}
\mathbf{b}_0 & & & \\
\mathbf{b}_1 & \mathbf{b}_0^1 & & \\
\mathbf{b}_2 & \mathbf{b}_1^1 & \mathbf{b}_0^2 & \\
\mathbf{b}_3 & \mathbf{b}_2^1 & \mathbf{b}_1^2 & \mathbf{b}_0^3
\end{array}$$

How many storage is required?

Some Properties of Bézier Curves

Can be inferred by de Casteljau algorithm

- **Affine invariance** $\leftarrow$ Sequence of linear interpolations
  cf) Not projectively invariant
- **Invariance under affine parameter transformations**

$$\mathbf{b}_i^r(u) = \frac{b-u}{b-a}\mathbf{b}_i^{r-1}(t) + \frac{u-a}{b-a}\mathbf{b}_{i+1}^{r-1}(t)$$

  where $t = (u-a)/(b-a)$
- **Convex hull property** For $t \in [0,1]$, $\mathbf{b}^n(t)$ lies in the convex hull of the control polygon.
  Useful for *interference checking* (Why?)
- **Endpoint interpolation**

$$\mathbf{b}^n(0) = \mathbf{b}_0 \text{ and } \mathbf{b}^n(1) = \mathbf{b}_n$$

- **Designing with Bézier curves** “The Bézier curve mimics the Bézier polygon.”

The Blossom

Use a new parameter t_r at rth step of the de Casteljau algorithm:

$$
\begin{array}{llll}
\mathbf{b}_0 & & & \\
\mathbf{b}_1 & \mathbf{b}_0^1[t_1] & & \\
\mathbf{b}_2 & \mathbf{b}_1^1[t_1] & \mathbf{b}_0^2[t_1, t_2] & \\
\mathbf{b}_3 & \mathbf{b}_2^1[t_1] & \mathbf{b}_1^2[t_1, t_2] & \mathbf{b}_0^3[t_1, t_2, t_3]
\end{array}
$$

- $\mathbf{b}[t_1, t_2, t_3] := \mathbf{b}_0^3[t_1, t_2, t_3]$ traces out a region in $\mathbb{E}^3$. (Why?)
- $\mathbf{b}[t_1, t_2, t_3]$ is a blossom. (Check it)
- The original Bézier curve is recovered when $t = t_1 = t_2 = t_3$. (Diagonality)

The Blossom (cont'd)

- The original Bézier points can be found by evaluating $\mathbf{b}[t_1, t_2, t_3]$ at arguments consisting only of 0's and 1's.
  ex) $\mathbf{b}[0, 0, 1] = \mathbf{b}_1$
- Intermediate entries $\mathbf{b}_i^r(t)$ can be also found.
  ex) $\mathbf{b}[0, 0, t] = \mathbf{b}_0^1(t)$

$$
\begin{array}{llll}
\mathbf{b}_0 = \mathbf{b}[0,0,0] & & & \\
\mathbf{b}_1 = \mathbf{b}[0,0,1] & \mathbf{b}[0,0,t] & & \\
\mathbf{b}_2 = \mathbf{b}[0,1,1] & \mathbf{b}[0,t,1] & \mathbf{b}[0,t,t] & \\
\mathbf{b}_3 = \mathbf{b}[1,1,1] & \mathbf{b}[t,1,1] & \mathbf{b}[t,t,1] & \mathbf{b}[t,t,t]
\end{array}
$$

de Casteljau Algorithm Using Blossom

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

and

$$\mathbf{b}_i = \mathbf{b}[0^{<n-i>}, 1^{<i>}]$$

- Explicit formula for blossoms?