Lecture 4

178 359
Simulation and Modeling

Pattarawit Polpinit

Department of Computer Engineering
Khon Kaen Uiversity

Overview

Review of Bezier curve

- Parametric equation
- the properties
- de Casteljau Algorithm

Bezier Curve Modeling (cont.)

- Derivative of Bezier curves
- Degree Elevation

Review

- Given $n + 1$ points $b_0, b_1, \ldots, b_n$ we can compute the Bezier curve of degree n by

  $$\mathbf{B}(t) = \sum_{i=0}^{n} b_i B_i^n(t)$$

  where a coefficient $B_i^n(t)$ is Bernstein polynomial which is defined as

  $$B_i^n(t) = \frac{n!}{i!(n-i)!} \cdot t^i \cdot (1-t)^{n-i}$$

  and t is our control weight that is in the range $[0, 1]$

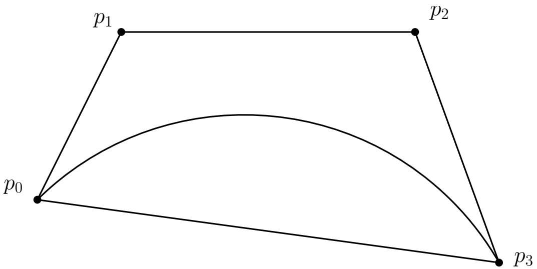
Review 2

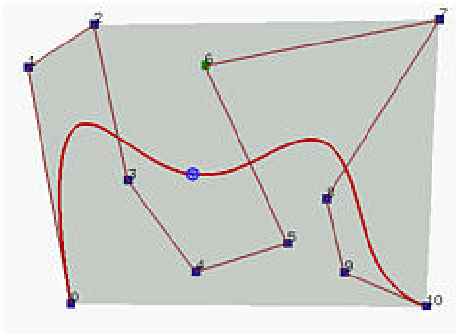
- The line segment between control points $b_0 b_1, b_1 b_2, \dots, b_{n-1} b_n$ are called legs.

Bezier curve properties:

- The degree of a Bezier curve is defined by $n + 1$ control points in n.
- $\mathbf{B}(t)$ passes through (interpolates) points b_0 and b_n.
- All Bernstien polynomials are nonnegative. In other words, the coefficients of the curve are nonnegative.
- Partition of Unity. All the coefficient sum to 1. This implies affine combination
- Convex combination. This imply that the curve will lie in the convex hull defined by the control points. Note that not all control points will be on the boundary of the convex hull.

Review 3



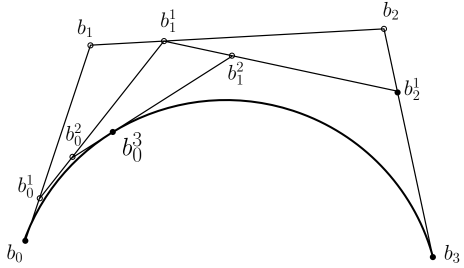


The domain is not $[0, 1]$

- Sometimes you are in situation that the domain is not in the proper range $[0, 1]$.
- So you do scaling.
- Say $t \in [a, b]$ then scale t to $[0, 1]$ by

$$\bar{t} = \frac{t - a}{b - a}$$

de Casteljau Algoritm: Review



The goal is to find a point $\mathbf{B}(t)$ on the curve for a given t. The steps are as follows:

- From a given $n+1$ control points $b_0, b_1, \ldots, b_n$, create n points $b_0^1, b_1^1, \ldots, b_n^1$ using the formula:

$$b_i^r(t) = (1-t)b_i^{r-1}(t) + tb_{i+1}^{r-1}(t)$$

de Casteljau Algoritm: Review 2

- repeat the process to create $n-1$ points $b_0^2, b_1^2, \ldots, b_n^2$.
- Eventually there will be only one point left. This point, proved by De Calteljau, is a point on the curve corresponding to t.
- The algorithm is recursive in nature.
- It is more numerical stable.
- The time complexity is $O(n^2)$.

Derivative of Bezier Curves

- Several times, we will want to compute tangent lines over a point on the curve.
- Take derivative of the curve which is rather simple. Recall

$$\mathbf{B}(t) = \sum_{i=0}^{n} b_i B_i^n(t)$$

where

$$B_i^n(t) = \frac{n!}{i!(n-i)!} \cdot t^i \cdot (1-t)^{n-i}$$

- The derivative of the Bernstien polynomial is then

$$\frac{\mathrm{d}}{\mathrm{d}t} B_i^n(t) = n\left(B_{i-1}^{n-1}(t) - B_i^{n-1}(t)\right)$$

Note the derivative is sometimes denoted as $(B_i^n)'(t)$

Derivative of Bezier Curves 2

- Then compute the derivative the curve

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{B}(t) = \sum_{i=0}^{n-1} B_i^{n-1}(t)\left(n(b_{i+1} - b_i)\right)$$

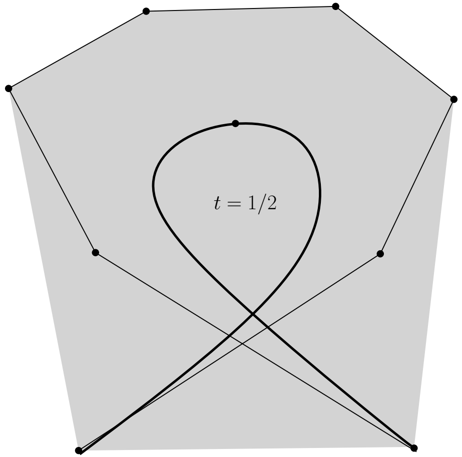
- Let $p_0 = n(b_1 - b_0)$, $p_1 = n(b_2 - b_1)$, $\ldots$, $p_{n-1} = n(b_n - b_{n-1})$ we reduce the above equation to

$$\mathbf{B}'(t) = \sum_{i=0}^{n-1} B_i^{n-1}(t) p_i$$

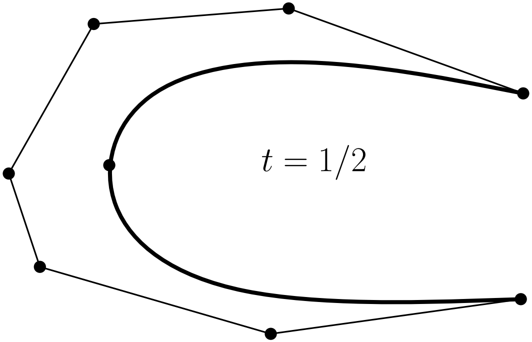
- This means that the derivative of a Bezier curve is also a Bezier curve, but with lower degree $n-1$ and the control points are $n(b_1 - b_0), \ldots, n(b_n - b_{n-1})$.

Hodograph

- The derivative of Bezier curves are usually referred to as Hodograph.
- The next figure show the Bezier curve created from 7 control points

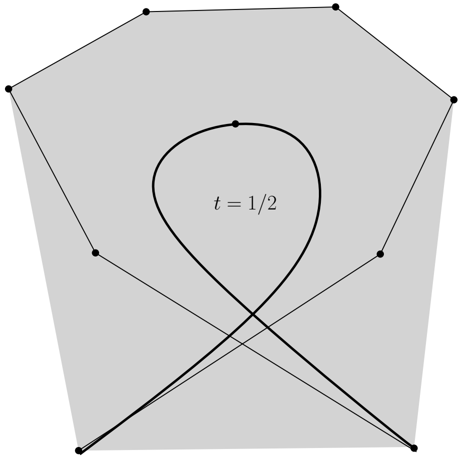


Hodograph2



Bezier Curves are tangent to their first and last legs

- Recall that a line segment between control points is called leg.
- The line segments of the control point $b_0 b_1$ and $b_{n-1} b_n$ are the tangent lines to the point b_0 and b_n respectively.



Joining Two Bezier Curves

- Since Bezier curves have tangent lines to the first and last legs we can simply join two curves together.
- We want to achieve C^1 continuity. (Recall C^0, C^1 and C^1 continuity)

Suppose we have two Bezier curves $\mathbf{B}(t)$ and $\mathbf{C}(t)$ where $b_0, \ldots, b_m$ and $p_0, \ldots, p_n$ are the control points respectively. The technique is as follows:

- We must match b_m and p_n. C^0-continuity
- To get a smooth transition b_{m-1}, b_m, p_0 and p_1 should be in the same line.
- This will make the joining look smooth. However it doesn't guarantee C^1-continuity.

Joining Two Bezier Curves

- To ensure C^1-continuity, the tangent line at $t = 1$ of $\mathbf{B}(t)$ and the tangent line at $t = 0$ of $\mathbf{C}(t)$ must be equal.
- That is

$$\mathbf{B}'(t) = m(b_m - b_{m-1}) = \mathbf{C}'(t) = n(p_1 - p_0)$$

- This means that to obtain C^1-continuity at the joining point the ratio of the last leg of the first curve an the length of the first leg of the second curve must be n/m.
- This in practical can be done by adjusting the control points b_m, b_{m-1}, p_1 or p_0).

Two Bezier Curves Created From Derivative Bezier Curves

Let us rewrite the derivative of the Bezier curve:

$$\begin{aligned}\mathbf{B}'(t) &= \sum_{i=1}^{n-1} B_i^{n-1}(t)\left(n(b_{i+1} - b_i)\right) \\ &= n\left[\left(\sum_{i=0}^{n-1} B_i^n(t) b_{i+1}\right) - \left(\sum_{i=0}^{n-1} B_i^n(t) b_i\right)\right]\end{aligned}$$

- As you can see, the derivative is now in terms of linear combination of two Bezier curves:

$$\mathbf{B}_1'(t) = \sum_{i=0}^{n-1} B_i^n(t) b_{i+1} \qquad \mathbf{B}_2'(t) = \sum_{i=0}^{n-1} B_i^n(t) b_i$$

Higher Order Derivatives

- To compute derivative of Bezier curve of order higher than 1 is still simple. Recall

$$\mathbf{B}'(t) = \sum_{i=1}^{n-1} B_i^{n-1}(t) p_i$$

- Apply the derivative again yield

$$\mathbf{B}''(t) = \sum_{i=1}^{n-2} B_i^{n-2}(t) \left((n-1)(p_{i+1} - p_i)\right)$$

Note that $p_i = n(b_{i+1} - b_i)$.

Degree Elevation

- In real world applications, two or more Bezier curves are used.
- This requires that all the curves have the same degree.
- Higher degree means more complexity, but it's a trade off for high flexibility.
- GOAL: increase degree of the curve without changing the shape of the curve.
- This is called Degree Elevation.

Degree Elevation: The algorithm

- Suppose we have a Bezier curve of degree n which has control points $b_0, \ldots, b_n$. And we want to increase the degree of the curve to degree $n+1$ without changing its shape.
- The new curve will have $n+2$ control points.
- b_0 and b_n must in those set of points.
- Task: find n more points

The algorithm (cont.)

Let us define a set of control points for the new bezier curve $p_0, \ldots, p_{n+1}$.

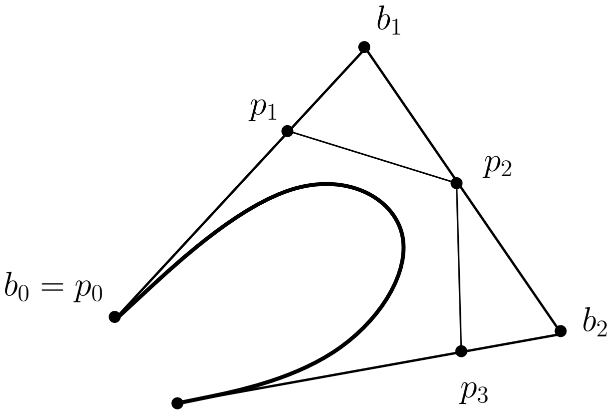
- As mentioned $p_0 = b_0$ and $p_{n+1} = b_n$.
- The rest of the n control points can be computed as

$$p_i = \frac{i}{n+1} b_{i-1} + \left(1 - \frac{i}{n+1}\right) b_i \text{ where } 1 \leq i \leq n$$

$$\begin{aligned}
p_1 &= \frac{1}{n+1} b_0 + \left(1 - \frac{1}{n+1}\right) b_1 \\
p_2 &= \frac{2}{n+1} b_1 + \left(1 - \frac{2}{n+1}\right) b_2 \\
&\vdots \\
p_n &= \frac{n}{n+1} b_{n-1} + \left(1 - \frac{n}{n+1}\right) b_n
\end{aligned}$$

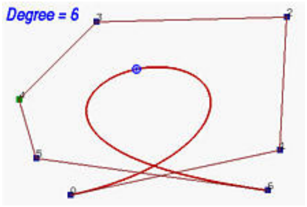
Degree Elevation: Note

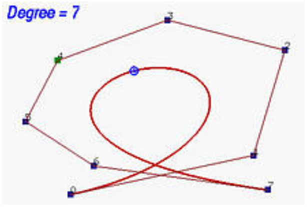
- Each leg from the original curve ($b_{i-1}b_i$) contains exactly one new control point (p_i).
- Similar to de Casteljau Algorithm, the new point divide the line segment in to ratio of $\frac{i}{n+1}$ and $1 - \frac{i}{n+1}$.
- Graphically the degree elevation illustrate a corner cutting effect.



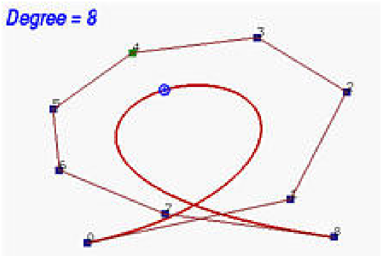
Degree Elevation: Note 2

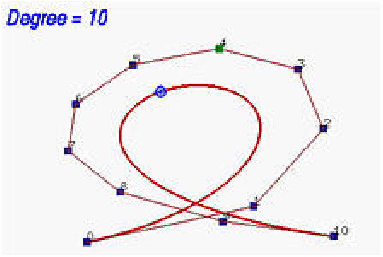
- The degree elevation can be repeated as may times as you want.
- The higher degree the more the new control points move toward the curve.





Degree Elevation: Note 3





Degree Elevation: Note 4