CMPSCI 473: Introduction to Computer Graphics (Fall 2010)

Lecture 7: Curves

Rui Wang

\[ V_0, V_1, V_2, V_3 \]
Overview

- Curves
  - Motivation
  - Polynomial curves
  - Bezier curve
  - Cubic B-spline

- Curved Surfaces
- Subdivision Surfaces
Motivation

- Mathematical way to express and model smooth curves.
- Motivated by “draftsman’s spline”
  - Long, narrow strip of wood/plastic
  - Fit curves through specified data points
  - Shaped by lead weights called “ducks”
  - Gives curves that are “smooth” or “fair”
Motivation

The ducks and spline are used to make tighter curves.
Motivation

☐ Many applications in graphics
  - Fonts
  - Animation path
  - Shape modeling

☐ Many applications in science & engi.
  - Evaluate smooth surface from sensor data
  - Automobile design
  - Aircraft design
  - ...

ABC
Design Goals

☐ Desired Attributes:
- Predictable control
- Fast to compute
- Analytic representation
- Local control
- Continuity

☐ Approach
- Piecewise
- Parametric
- Polynomials
Parametric Polynomials

Definition

- Polynomial (degree n)

\[ f(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_n t^n \]

0 ≤ t ≤ 1

- The a’s are the coefficients.
- Note that the coefficients can be vectors.

\[ Q(t) = a_0 + a_1 t + a_2 t^2 + \cdots + a_n t^n \]
Parametric Polynomials

- **Definition**
  - **Canonical form**
    \[ Q(t) = \sum_{i=0}^{n} a_i t^i \]
  - **Generalized form**

Can you write down the derivative \( Q'(t) \)?

- **Generalized form**
  \[ Q(t) = \sum_{i=0}^{n} P_i b_i(t) \]
  - **Basis polynomial**
Piecewise Polynomials

- Use different polynomials on different sections of the curve
  - Provide flexibility and local control
  - How to guarantee continuity/smoothness at joints (knots)?

- Piecewise line segments (degree 1 polynomial)
- Piecewise cubic
In the following

- We will look at several polynomial curves
  - Bezier curves
  - Hermite curves
  - Cubic B-Splines
Bézier curves

- Developed independently in 1960s by
  - Pierre Bézier (at Renault)
  - Paul de Casteljau (at Citroen)
- It’s an approximation curve

$P_i$'s are control points

$Q(t)$
Bézier curves

Let’s first look at the recursive definition:
- De Casteljau algorithm
- Recursion: \[ P_i^j = (1 - t) P_i^{j-1} + t P_{i+1}^{j-1} \]
- \( j \) indicates the level of recursion

\[ P_i \rightarrow \text{are control points} \]
Bézier curves

Let’s first look at the recursive definition:

- De Casteljau algorithm
- Recursion: $P_i^j = (1 - t)P_i^{j-1} + tP_{i+1}^{j-1}$
- $j$ indicates the level of recursion

$P_i^0 \rightarrow$ control points
Bézier curves

Let’s first look at the recursive definition:

- De Casteljau algorithm
- Recursion: \( P_i^j = (1 - t) P_i^{j-1} + t P_{i+1}^{j-1} \)
- \( j \) indicates the level of recursion

\( P_i^0 \rightarrow \text{control points} \)
Bézier curves

Let’s first look at the recursive definition:

- De Casteljau algorithm
- Recursion: \( P_i^j = (1 - t) P_i^{j-1} + t P_{i+1}^{j-1} \)
- \( j \) indicates the level of recursion

\( P_i^0 \rightarrow \text{control points} \)
Bézier curves

Let’s first look at the recursive definition:

- De Casteljau algorithm
- Recursion: \( P_i^j = (1-t)P_i^{j-1} + tP_{i+1}^{j-1} \)
- \( j \) indicates the level of recursion

\[ P_i^0 \rightarrow \text{control points} \]
Bézier curves

Next, let’s look at the explicit formula:
- The simple case of 3 control points
Bézier curves

Next, let’s look at the explicit formula:
- The simple case of 3 control points

\[
Q(t) = P_0^2 \\
= (1 - t)P_0^1 + tP_1^1 \\
= (1 - t)[(1 - t)P_0^0 + tP_1^0] + t[(1 - t)P_1^0 + tP_2^0] \\
= (1 - t)^2P_0^0 + 2t(1 - t)P_1^0 + t^2P_2^0 \\
= (1 - t)^2P_0 + 2t(1 - t)P_1 + t^2P_2
\]
Bézier curves

- Next, let’s look at the explicit formula:
  - The general case of $n$ control points

\[ Q(t) = \sum_{i=0}^{n} P_i \binom{n}{i} t^i (1-t)^{n-i} \]

\[ \binom{n}{i} = \frac{n!}{i!(n-i)!} \]

- This means: a point on the curve corresponding to parameter $t$ is defined as a \textbf{weighted sum} of all control points.
Bézier curves

Next, let’s look at the explicit formula:

- The general case of \( n \) control points

\[
Q(t) = \sum_{i=0}^{n} P_i \binom{n}{i} t^i (1 - t)^{n-i}
\]

- Do all the weights sum up to 1?
Bézier curves

Next, let’s look at the explicit formula:

- The general case of n control points

\[ Q(t) = \sum_{i=0}^{n} P_i \binom{n}{i} t^i (1 - t)^{n-i} \]

\[ \binom{n}{i} = \frac{n!}{i!(n-i)!} \]

Properties:

- Polynomial of degree n.
- Defined using (n+1) control points.
- \( Q(0) = ? \)
- \( Q(1) = ? \)
Bézier curves

Next, let’s look at the explicit formula:

The general case of n control points

\[ Q(t) = \sum_{i=0}^{n} P_i \binom{n}{i} t^i (1 - t)^{n-i} \quad \left( \binom{n}{i} = \frac{n!}{i!(n-i)!} \right) \]

Properties:

- Polynomial of degree n.
- Defined using (n+1) control points.
- \( Q(0) = P_0 \) ➔ The curve passes through the two end points.
- \( Q(1) = P_n \)
Bézier curves

Next, let’s look at the explicit formula:

The general case of n control points

\[
Q(t) = \sum_{i=0}^{n} P_i \binom{n}{i} t^i (1 - t)^{n-i} \quad \binom{n}{i} = \frac{n!}{i!(n-i)!}
\]

Properties:

- \( Q'(0) = ? \)
- \( Q'(1) = ? \)
Bézier curves

Next, let’s look at the explicit formula:

The general case of n control points

\[ Q(t) = \sum_{i=0}^{n} P_i \binom{n}{i} t^i (1-t)^{n-i} \]

\[ \begin{pmatrix} n \\ i \end{pmatrix} = \frac{n!}{i!(n-i)!} \]

Properties:

- \( Q'(0) = n (P_1 - P_0) \)
- \( Q'(1) = n (P_n - P_{n-1}) \)

Convex hull

Affine invariance
Bézier curves

- Examples from the textbook:
Bézier curves – drawing algorithm

How to write a program to draw the Bézier curves?
- Generate some sampled points on the curve
- Connect these sample points to approximate the curve
- Why does this work?
- How to generate sampled curve points?
Bézier curves – drawing algorithm

- Generate sampled curve points by taking parameter $t$ at fixed intervals:

  ```java
  Point[] curve_pts = new Point[Nseg+1];
  for(int i=0; i<=Nseg; i++) {
    float t = (float)i / Nseg;
    calculate Q(t) using the recursive formula;
    curve_pts[i] = calculated Q(t);
  }
  for(int i=0; i<Nseg; i++) {
    drawline(curve_pts[i], curve_pts[i+1]);
  }
  ```

- Can you improve the algorithm?
Bézier curves – drawing algorithm

☐ Can you improve the algorithm?
Piecewise Bézier curves

- A disadvantages of Bézier curve:
  - Changing any control point leads to global change (i.e. no local control property)

- Piecewise Bézier curves
  - Connecting pieces of Bézier curves to form a longer curve
  - Difficult to guarantee smoothness (such as $C^1$ continuity)
Cubic curves

- From now on, let’s focus on piecewise cubic curves
  - Each piece is in the form
    \[ Q(t) = at^3 + bt^2 + ct + d \]

- By connecting such pieces together, we can define a longer and more complex curve.
Hermite curves

- Given two end points $P_0$ and $P_1$ with two tangent vectors $P'_0$ and $P'_1$
- Similarity to cubic Bézier curves

- Find the cubic coefficients
  \[ Q(t) = at^3 + bt^2 + ct + d \]
Hermite curves

- Given two end points $P_0$ and $P_1$ with two tangent vectors $P'_0$ and $P'_1$
- Similarity to cubic Bézier curves

- Find the cubic coefficients

$$Q(t) = at^3 + bt^2 + ct + d$$

$$Q'(t) = 3at^2 + 2bt + c$$

- $Q(0) = d = P_0$
- $Q(1) = a + b + c + d = P_1$
- $Q'(0) = c = P'_0$
- $Q'(1) = 3a + 2b + c = P'_1$

4 unknowns
4 equations
Hermite curves

- Solve linear equations from previous slide:

\[ a = 2P_0 - 2P_1 + P'_0 + P'_1 \]
\[ b = -3P_0 + 3P_1 - 2P'_0 - P'_1 \]
\[ c = P'_0 \]
\[ d = P_0 \]

\[
\begin{bmatrix}
  a \\
  b \\
  c \\
  d
\end{bmatrix} =
\begin{bmatrix}
  2 & -2 & 1 & 1 \\
  -3 & 3 & -2 & -1 \\
  0 & 0 & 1 & 0 \\
  1 & 0 & 0 & 0
\end{bmatrix}
\times
\begin{bmatrix}
P_0 \\
P_1 \\
P'_0 \\
P'_1
\end{bmatrix}
\]
Hermite curves

- Remember:

\[ Q(t) = at^3 + bt^2 + ct + d = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \]

- Putting it together (from previous slide):

\[ Q(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P'_0 \\ P'_1 \end{bmatrix} \]
Hermite curves

- Example

A Hermite cubic spline made up of three segments.

- Interpolation curves
  - Different from approximation curves
Cubic B-Spline curves

- Arbitrary number of control points
- Curve segments are defined by every four adjacent control points
  - local control property
- Matrix form:

\[ Q(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \times \frac{1}{6} \begin{bmatrix}
-1 & 3 & -3 & 1 \\
3 & -6 & 3 & 0 \\
-3 & 0 & 3 & 0 \\
1 & 4 & 1 & 0
\end{bmatrix} \times \begin{bmatrix}
P_0 \\
P_1 \\
P_2 \\
P_3
\end{bmatrix} \]

You can also compute \( Q'(t) \) by simply differentiating \( t \)
Cubic B-Spline curves

In general, cubic curves can be represented in this matrix form:

\[ Q(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \times M \times \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} \]

For cubic Bézier curve
(4 control points)

\[
M = \begin{bmatrix}
-1 & 3 & -3 & 1 \\
3 & -6 & 3 & 0 \\
-3 & 3 & 0 & 0 \\
1 & 0 & 0 & 0 \\
\end{bmatrix}
\]
Cubic B-Spline curves

Properties
- Polynomial of degree 3
- Number of control points: arbitrary ($\geq 4$)
- $Q(0) =$ ?
- $Q(1) =$ ?
- $Q'(0) =$ ?
- $Q'(1) =$ ?
Cubic B-Spline curves

Properties

- Polynomial of degree 3
- Number of control points: arbitrary (\( \geq 4 \))
- \( Q(0) = \frac{1}{6}(P_0 + 4P_1 + P_2) \)  What does this mean geometrically?
- \( Q(1) = ... \)
- \( Q'(0) = \frac{1}{2}(P_2 - P_0) \)  And this?
- \( Q'(1) = ... \)
- Convex Hull
- Affine invariance
Cubic B-Spline curves

- Display algorithm
- Equivalence of cubic Bezier and B-Spline
- Improving controls
  - Closed curves (wrap around control points)
  - Interpolating certain control points (repeat those control points)
  - Changing the order of polynomials