CS 536
Computer Graphics

Hermite and Catmull-Rom Curves
Week 2, Lecture 4
David Breen, William Regli and Maxim Peysakhov
Department of Computer Science
Drexel University

Additional slides from Don Fussell, University of Texas and Steve Marschner, Cornell University

Outline
• Hermite Curves
• Continuity
• Catmull-Rom Curves

Hermite Curve
• 3D curve of polynomial bases
• Geometrically defined by position and tangents at end points
• No convex hull guarantees
• Supports tangent-continuous (C¹) composite curves

Algebraic Representation
• All of these curves are just parametric algebraic polynomials expressed in different bases
• Parametric cubic curve (in R³)
  \[ P(u) = au^3 + bu^2 + cu + d \]

  First derivative of curve
  \[ P'(u) = 3au^2 + 2bu + c \]

  Solving for the coefficients:
  \[ a = 3p(0) - 2p(1) + p'(0) + p''(1) \]
  \[ b = -3p(0) + 3p(1) - 2p'(0) - p''(1) \]
  \[ c = p'(0) \]
  \[ d = p(0) \]
**Hermite Curves**

- Putting it all together
  \[ P(u) = au^3 + bu^2 + cu + d \]
  \[ a = 2p(0) - 2p(1) + p''(0) + p''(1) \]
  \[ b = -3p(0) + 3p(1) - 2p''(0) - p''(1) \]
  \[ c = p''(0) \]
  \[ d = p(0) \]
  \[ P(u) = (2u^3 - 3u^2 + 1)p(0) + (-2u^2 + 3u)p(1) + (u^2 - 2u + 1)p''(0) + (u - 1)p''(1) \]

**Blending Functions**

- At \( u = 0 \):
  - \( H_1 = 1, H_2 = H_3 = H_4 = 0 \)
  - \( H_1' = H_2' = H_3' = H_4' = 0, H_5' = 1 \)
  \[ P(0) = p0 \]
  \[ P'(0) = T0 \]

- At \( u = 1 \):
  - \( H_1 = H_2 = H_3 = H_4 = 0, H_5 = 1 \)
  - \( H_1' = H_2' = H_3' = H_4' = 0, H_5' = 1 \)
  \[ P(1) = p1 \]
  \[ P'(1) = T1 \]

**Hermite and Algebraic Forms**

- Putting it all together produces the matrix formulation for the Hermite curve \( P(u) \)
  \[ P(u) = GMu \]
  \[ P'(u) = GBH \]

- \( M \) transforms geometric coefficients ("coordinates") from the Hermite basis to the algebraic coefficients of the monomial basis

---

**Hermite Basis**

- Substituting for the coefficients and collecting terms gives
  \[ P(u) = (2u^3 - 3u^2 + 1)p(0) + (-2u^2 + 3u)p(1) + (u^2 - 2u + 1)p''(0) + (u - 1)p''(1) \]

- Call
  \[ H_1(u) = (2u^3 - 3u^2 + 1) \]
  \[ H_2(u) = (-2u^2 + 3u) \]
  \[ H_3(u) = (u^2 - 2u + 1) \]
  \[ H_4(u) = (u - 1) \]

the Hermite blending functions or basis functions

- Then \( P(u) = H_1(u)p(0) + H_2(u)p(1) + H_3(u)p''(0) + H_4(u)p''(1) \)

---

**Hermite Curves - Matrix Form**

- Putting this in matrix form
  \[ H = \begin{bmatrix}
  H_1(u) & H_2(u) & H_3(u) & H_4(u)
  \end{bmatrix} \]
  \[ = \begin{bmatrix}
  2 & -3 & 0 & 1 \\
  1 & -2 & 1 & 0 \\
  1 & -1 & 0 & 1
  \end{bmatrix}u^3 \\
  \begin{bmatrix}
  0 & 0 & 0 & 0
  \end{bmatrix}u^2 \\
  \begin{bmatrix}
  0 & 0 & 0 & 0
  \end{bmatrix}u \\
  \begin{bmatrix}
  0 & 0 & 0 & 0
  \end{bmatrix}
  \]
  \[ = M_u U \]

- \( M_u \) is called the Hermite characteristic matrix

- Collecting the Hermite geometric coefficients into a geometry vector \( G \),
  \[ G = [p(0) \quad p(1) \quad p''(0) \quad p''(1)] \]

---

**Hermite Curves**

- Geometrically defined by position and tangents at end points
Hermite to Bézier

- Mixture of points and vectors is awkward and unintuitive
- Specify tangents as differences of points

\[ p_0 = q_0; \quad p_3 = q_1; \]
\[ p_1 = q_0 + (1/3)t_0; \quad p_2 = q_1 - (1/3)t_1 \]

- note derivative is defined as 3 times offset

Bezizer to Hermite

\[ q_0 = p_0; \quad q_1 = p_3; \]
\[ t_0 = 3(p_1 - p_0); \quad t_1 = 3(p_3 - p_2); \]

- note derivative is defined as 3 times offset

Issues with Bézier Curves

- Creating complex curves requires many control points
  - potentially a very high-degree polynomial with many wiggles
- Bézier blending functions have global support over the whole curve
  - move just one point, change whole curve
- Improved Idea: link \((C^1)\) lots of low degree (cubic) Bézier curves end-to-end

Continuity

Two types:
- Geometric Continuity, \(G\):
  - endpoints meet
  - tangent vectors’ directions are equal
- Parametric Continuity, \(C\):
  - endpoints meet
  - tangent vectors’ directions are equal
  - tangent vectors’ magnitudes are equal
- In general: \(C\) implies \(G\) but not vice versa
Parametric Continuity

- **Continuity** (recall from the calculus):
  - Two curves are $C^i$ continuous at a point $p$ iff the $i$-th derivatives of the curves are equal at $p$

Continuity

- What are the conditions for $C^0$ and $C^1$ continuity at the joint of curves $x'$ and $x''$?
  - Tangent vectors at end points equal
  - End points equal

$$Q'(1) = Q'(0), \quad \frac{dQ}{dt}(1) = \frac{dQ}{dt}(0)$$

Chaining Bézier curves

- No continuity built in
- Achieve $C^1$ using collinear control points around join points

Catmull-Rom splines

- Our first example of an interpolating spline
- Like Bézier, equivalent to Hermite
  - In fact, all splines of this form are equivalent
- First example of a spline based on just an input point sequence
- Does not have convex hull property
- Only has $C^1$ continuity
Catmull-Rom splines

- A sequence of Hermite/Bezier curves
- Would like to define tangents automatically
  - use adjacent control points
- end tangents: user-defined or fit a parabola

Catmull-Rom splines

- Tangents are \((p_{k+1} - p_{k-1})/2\) for interior control points
- User specifies tangents at first \((T_0)\) and last \((T_N)\) input points
- Or fit parabola to first/last 3 points
  
  \[
  q_0 = p_k \\
  q_1 = p_{k+1} \\
  t_0 = 0.5(p_{k+1} - p_{k-1}) \\
  t_1 = 0.5(p_{k+2} - p_k)
  \]

Adding tension

- Adding tension to Catmull-Rom spline involves adjusting tangents at interior join points, \(p_i\)
  
  \[
  t_0 = (1 - T)0.5(p_{k+1} - p_{k-1}) \\
  t_1 = (1 - T)0.5(p_{k+2} - p_k)
  \]

- When \(T=0\), standard C-R spline
- When \(T=1\), tangent is zero

Adding tension

- Scale user-provided tangent vectors
  
  \[
  T_0' = (1 - T) T_0 \\
  T_N' = (1 - T) T_N
  \]

Programming Assignment 2

- Process command-line arguments
- Read in 3D input points and tangents
- Compute Bezier control points for curves defined by each two input points
- Modify tangents with tension parameter
- Use HW1 code to compute points on each Bezier curve
- Each Bezier curve should be a polyline
- Output points by printing them to the console as an IndexedLineSet with multiple polylines, and control points as spheres in Open Inventor format