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, Steve Marschner, Cornell University and Sun-Jeong Kim, Hallym University

Outline
- Hermite Curves
- Continuity
- Catmull-Rom Splines
- $C^2$ Piecewise Splines

Algebraic Representation
- All of these curves are just parametric algebraic polynomials expressed in different bases
- Parametric cubic curve (in $\mathbb{R}^3$)

$$P(u) = au^3 + bu^2 + cu + d$$

- First derivative of curve

$$P'(u) = 3au^2 + 2bu + c$$

- Specifying endpoints and tangent vectors at endpoints

$$P(0) = d$$
$$P'(0) = c$$

- Solving for the coefficients:

$$a = \frac{2p(0) - 2p(1) + p'(0) + p'(1)}{2}$$
$$b = \frac{3p(0) + 3p(1) - 2p'(0) - 2p'(1)}{3}$$
$$c = p''(0)$$
$$d = p(0)$$

Hermite Curve
- 3D curve of polynomial bases
- Geometrically defined by position and tangents at endpoints
- No convex hull guarantees
- Supports tangent-continuous ($C^1$) composite curves

Algebraic Representation
- All of these curves are just parametric algebraic polynomials expressed in different bases
- Parametric cubic curve (in $\mathbb{R}^3$)

$$P(u) = au^3 + bu^2 + cu + d$$

- First derivative of curve

$$P'(u) = 3au^2 + 2bu + c$$

**Outline**

- Hermite Curves
- Continuity
- Catmull-Rom Splines
- $C^2$ Piecewise Splines

**Algebraic Representation**

- All of these curves are just parametric algebraic polynomials expressed in different bases
- Parametric cubic curve (in $\mathbb{R}^3$)

$$P(u) = au^3 + bu^2 + cu + d$$

- First derivative of curve

$$P'(u) = 3au^2 + 2bu + c$$

**Hermite Curves**

- 12 degrees of freedom (4 3-d vector constraints)
- Specify endpoints and tangent vectors at endpoints

$$P(0) = d$$
$$P'(0) = c$$

- Solving for the coefficients:

$$a = \frac{2p(0) - 2p(1) + p'(0) + p'(1)}{2}$$
$$b = \frac{3p(0) + 3p(1) - 2p'(0) - 2p'(1)}{3}$$
$$c = p''(0)$$
$$d = p(0)$$

**Algebraic Representation**

- All of these curves are just parametric algebraic polynomials expressed in different bases
- Parametric cubic curve (in $\mathbb{R}^3$)

$$P(u) = au^3 + bu^2 + cu + d$$

- First derivative of curve

$$P'(u) = 3au^2 + 2bu + c$$

**Hermite Curves**

- 12 degrees of freedom (4 3-d vector constraints)
- Specify endpoints and tangent vectors at endpoints

$$P(0) = d$$
$$P'(0) = c$$

- Solving for the coefficients:

$$a = \frac{2p(0) - 2p(1) + p'(0) + p'(1)}{2}$$
$$b = \frac{3p(0) + 3p(1) - 2p'(0) - 2p'(1)}{3}$$
$$c = p''(0)$$
$$d = p(0)$$
Hermite and Algebraic Forms

- Putting it all together produces the matrix formulation for the Hermite curve \( P(u) \)

\[
P(u) = au^3 + bu^2 + cu + d
\]

\[
a = 2p(0) - 2p(1) + p''(0) + p''(1)
\]

\[
b = -3p(0) + 3p(1) - 2p''(0) - p''(1)
\]

\[
c = p''(0)
\]

\[
d = p(0)
\]

Putting it all together produces the matrix

\[
P(u) = (2u^3 - 3u^2 + 1)p(0) + (-2u^3 + 3u^2 - u)p''(0) + (u^3 - u)p''(1)
\]

\[
Hermite Curves

- Substituting for the coefficients and collecting terms gives

\[
P(u) = (2u^3 - 3u^2 + 1)p(0) + (-2u^3 + 3u^2 - u)p''(0) + (u^3 - u)p''(1)
\]

Hermite Basis

- Putting this in matrix form

\[
\begin{bmatrix}
H_0(u) & H_1(u) & H_2(u) & H_3(u)
\end{bmatrix} = \begin{bmatrix}
2 & -3 & 0 & 1 \\
-2 & 3 & 0 & 0 \\
1 & -2 & 1 & 0 \\
1 & -1 & 0 & 1
\end{bmatrix} \begin{bmatrix}
u^3 \\
u^2 \\
u \\
1
\end{bmatrix}
\]

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

<table>
<thead>
<tr>
<th>( M_H ) (Hermite characteristic matrix)</th>
</tr>
</thead>
</table>
| \( M_H = \begin{bmatrix}
2 & -3 & 0 & 1 \\
-2 & 3 & 0 & 0 \\
1 & -2 & 1 & 0 \\
1 & -1 & 0 & 1
\end{bmatrix} \) |

- Collecting the Hermite geometric coefficients into a geometry vector \( G \)

\[
G = [p(0) \quad p(1) \quad p'(0) \quad p'(1)]
\]

Hermite Curves - Matrix Form

- Putting this in matrix form

\[
\begin{bmatrix}
H_0(u) & H_1(u) & H_2(u) & H_3(u)
\end{bmatrix} = \begin{bmatrix}
2 & -3 & 0 & 1 \\
-2 & 3 & 0 & 0 \\
1 & -2 & 1 & 0 \\
1 & -1 & 0 & 1
\end{bmatrix} \begin{bmatrix}
u^3 \\
u^2 \\
u \\
1
\end{bmatrix}
\]

Hermite Curves

- Geometrically defined by position and tangents at end points

\[
\begin{bmatrix}
H_0(u) & H_1(u) & H_2(u) & H_3(u)
\end{bmatrix} = \begin{bmatrix}
2 & -3 & 0 & 1 \\
-2 & 3 & 0 & 0 \\
1 & -2 & 1 & 0 \\
1 & -1 & 0 & 1
\end{bmatrix} \begin{bmatrix}
u^3 \\
u^2 \\
u \\
1
\end{bmatrix}
\]

D. Fussell  – UT, Austin
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

Beziers 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 (C¹) lots of low degree (cubic) Bézier curves end-to-end

Continuity

Two types:
- Geometric Continuity, \( G^i \):
  - endpoints meet
  - tangent vectors’ directions are equal
- Parametric Continuity, \( C^i \):
  - endpoints meet
  - tangent vectors’ directions are equal
  - tangent vectors’ magnitudes are equal
- In general: \( C^i \) implies \( G^i \) 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 $p$.

![Diagram of continuity types](image)

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

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

![Diagram of continuity conditions](image)

Chaining Bézier curves

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

![Diagram of chaining Bézier curves](image)

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 C1 continuity

![Diagram of Catmull-Rom splines](image)
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

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
  
  \[
  T_0' = (1 - T)T_0 \\
  T_N' = (1 - T)T_N
  \]

Curvature (C^2) Continuity

- Q: Suppose you want even higher degrees of continuity - e.g., not just slopes but curvatures - what additional geometric constraints are imposed?
  
  \[
  Q_0(0) = Q_0, (0) \quad Q_0(1) = Q_0, (1) \\
  Q_1(0) = Q_1, (0) \quad Q_1(1) = Q_1, (1) \\
  Q_0'(0) = Q_0', (0) \quad Q_0'(1) = Q_0', (1)
  \]

- We’ll begin by developing some more mathematics....
Specializing to n=3

- What’s the derivative $Q'(u)$ for a cubic Bezier curve?
  
  $$Q'(u) = 3(uD + (1-u)V_3) \cdot (D - 1)V_0 + (Q'(1) - 3D - 1)V_4$$

- Note that:
  - When $u=0$: $Q'(u) = 3(V_0 - V_3)$
  - When $u=1$: $Q'(u) = 3(V_3 - V_4)$

- Geometric interpretation:

- So for $C^1$ continuity, we need to set:
  
  $$3(V_3 - V_2) = 3(W_1 - W_0)$$

Second-Order Continuity

- So the conditions for second-order continuity are:
  $$V_1 - V_0 = (W_1 - W_0)$$
  $$V_4 - V_3 = (W_4 - W_3) - (W_1 - W_0)$$

- Putting these together gives:
  $$W_0 = V_0$$
  $$W_1 = V_1 - V_0$$
  $$W_2 = 2(V_1 - V_0) - W_1 - W_0$$
  $$W_3 = 2(V_3 - V_2) - W_2 - W_1$$
  $$W_4 = V_3 - 2V_2 + W_3$$

- Geometric interpretation:

Creating Continuous Splines

- We’ll look at three ways to specify splines with $C^1$ and $C^2$ continuity
  - $C^0$ interpolating splines
  - B-splines
  - Catmull-Rom splines

$C^2$ Interpolating Splines

- The control points specified by the user, called "joints", are interpolated by the spline

  $\text{joint}$

- For each of $x$ and $y$, we needed to specify 3 conditions for each cubic Bezier segment.
- So if there are $m$ segments, we’ll need $3m - 1$ conditions
- Q: How many these constraints are determined by each joint?

In-Depth Analysis, cont.

- At each interior joint $j$, we have:
  - Last curve ends at $j$
  - Next curve begins at $j$
  - Tangents of two curves at $j$ are equal
  - Curvature of two curves at $j$ are equal
- The $m$ segments give:
  - $m-1$ interior joints
  - 3 conditions
- The 2 end joints give 2 further constraints:
  - First curve begins at first joint
  - Last curve ends at last joint
- Gives $3m + 1$ constraints altogether

End Conditions

- The analysis shows that specifying $m+1$ joints for $m$ segments leaves 2 extra degree of freedom
- These 2 extra constraints can be specified in a variety of ways:
  - An interactive system
    - Constraints specified as user inputs
  - "Natural" cubic splines
    - Second derivatives at endpoints defined to be 0
    - Maximal continuity
      - Require $C^2$ continuity between first and last pairs of curves
Another Explanation
- Define the first Bezier curve
  \( (V_0, V_1, V_2, V_3) \)
- Next Bezier curve \( (W_0, W_1, W_2, W_3) \) has 3 constraints
  \( Q_n(1) = Q_{n+1}(0), Q_n'(1) = Q_{n+1}'(0) \)
  \( Q_n''(1) = Q_{n+1}''(0) \)
- It only has 1 degree of freedom
  – The location of its last control point \( (W_3) \)

Global vs. Local Control
- These \( C^2 \) interpolating splines yield only "global control" - moving any one joint (or control point) changes the entire curve!
- Global control is problematic:
  \- Makes splines difficult to design
  \- Makes incremental display inefficient
- There's a fix, but nothing comes for free. Two choices:
  \- Envelope
    \- Keep \( C^2 \) continuity
    \- Give up interpolation
  \- Cartesian Spline
    \- Keep interpolation
    \- Give up \( C^2 \) continuity - provides \( C^1 \) only

Programming Assignment 2
- Process command-line arguments
- Read in 3D input points and tangents
- Compute tangents at interior input points
- Modify tangents with tension parameter
- Compute Bezier control points for curves defined by each two input points
- 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