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 (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

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

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

$P(0) = d$

$P(1) = a + b + c + d$

$P^u(0) = c$

$P^u(1) = 3a + 2b + c$
Hermite Curves

• 12 degrees of freedom (4 3-d vector constraints)

• Specify endpoints and tangent vectors at endpoints

\[ P(0) = d \]
\[ P(1) = a + b + c + d \]
\[ P''(0) = c \]
\[ P''(1) = 3a + 2b + c \]

• Solving for the coefficients:

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

D. Fussell – UT, Austin
Hermite Curves

- Putting it all together

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

\[ a = 2p(0) - 2p(1) + p^u(0) + p^u(1) \]
\[ b = -3p(0) + 3p(1) - 2p^u(0) - p^u(1) \]
\[ c = p^u(0) \]
\[ d = p(0) \]

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

• Substituting for the coefficients and collecting terms gives

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

• Call

\[
\begin{align*}
    H_1(u) &= (2u^3 - 3u^2 + 1) \\
    H_2(u) &= (-2u^3 + 3u^2) \\
    H_3(u) &= (u^3 - 2u^2 + u) \\
    H_4(u) &= (u^3 - u^2)
\end{align*}
\]

the Hermite blending functions or basis functions

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

D. Fussell – UT, Austin
Blending Functions

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

\[ P'(u) = (6u^2 - 6u)p(0) + (-6u^2 + 6u)p(1) + (3u^2 - 4u + 1)p''(0) + (3u^2 - 2u)p''(1) \]

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

- At \( u = 1 \):
  - \( H_1 = H_3 = H_4 = 0, H_2 = 1 \)
  - \( H_1' = H_2' = H_3' = 0, H_4' = 1 \)
  - \( P(1) = p1 \)
  - \( P'(1) = T1 \)
Hermite Curves - Matrix Form

- Putting this in matrix form

\[
\mathbf{H} = \begin{bmatrix} H_1(u) & H_2(u) & H_3(u) & H_4(u) \end{bmatrix}
\]

\[
= \begin{bmatrix} 2 & -3 & 0 & 1 \\ -2 & 3 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} u^3 \\ u^2 \\ u \\ 1 \end{bmatrix}
\]

\[= \mathbf{M}_H \mathbf{U} \]

- \(\mathbf{M}_H\) is called the Hermite characteristic matrix
- Collecting the Hermite geometric coefficients into a geometry vector \(\mathbf{G}\),

\[
\mathbf{G} = [p(0) \ p(1) \ p'(0) \ p'(1)]
\]

D. Fussell – UT, Austin
Hermite and Algebraic Forms

• Putting it all together produces the matrix formulation for the Hermite curve $P(u)$

$$P(u) = GM_H U$$
$$P(u) = GB_H$$

• $M_H$ transforms geometric coefficients ("coordinates") from the Hermite basis to the algebraic coefficients of the monomial basis
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
Hermite to Bézier

\[ p_0 = q_0; \quad p_3 = q_1; \]

\[ p_1 = q_0 + \frac{1}{3}t_0; \quad p_2 = q_1 - \frac{1}{3}t_1 \]

– note derivative is defined as 3 times offset
Bezier 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
Back to Bézier Curves

• k+1 control points defines a degree k curve
• Endpoint interpolation
• Convex hull property
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, $G^i$:
  – 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: $C$ 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 $x^l$ and $x^r$?
  - tangent vectors at end points equal
  - end points equal

$$Q^l(1) = Q^r(0), \quad \frac{dQ^l}{dt}(1) = \frac{dQ^r}{dt}(0)$$
Continuity

• The derivative of $Q(t)$ is the parametric tangent vector of the curve:

$$\frac{d}{dt}Q(t) = Q'(t) = \begin{bmatrix} \frac{d}{dt}x(t) & \frac{d}{dt}y(t) & \frac{d}{dt}z(t) \end{bmatrix}^T = \frac{d}{dt}C \cdot T = C \cdot \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix}^T =$$

$$\begin{bmatrix} 3a_x t^2 + 2b_x t + c_x & 3a_y t^2 + 2b_y t + c_y & 3a_z t^2 + 2b_z t + c_z \end{bmatrix}^T$$
Continuity

• In 3D, compute this for each component of the parametric function
  – For the x component:
    
    \[ x^l(1) = x^r(0) = P_{4x}, \quad \frac{d}{dt} x^l(1) = 3(P_{4x} - P_{3x}), \quad \frac{d}{dt} x^r(0) = 3(P_{5x} - P_{4x}) \]

• Similar for the y and z components.
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 C1 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 \((p_k)\)
- 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
  
  \[ T_0' = (1 - T) \ T_0 \]
  
  \[ T_N' = (1 - T) \ T_N \]
Adding Tension

Tension = -1

Tension = 0

Tension = 1

Figure 7. Bending of the curve under various tensions.
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