Objects 2: Curves & Splines

Christian Miller
CS 354 - Fall 2011
Parametric curves

- Curves that are defined by an equation and a parameter $t$
  - Usually $t \in [0, 1]$, and curve is finite
- Can be discretized at arbitrary resolution
- Often defined with the aid of control points
- Used as modeling primitives
Lerp

p(t) = p_0 + t(p_1 - p_0) \quad p(t) = (1 - t)p_0 + tp_1

• Linear interpolation looks too harsh
• We’d like something smoother
**Lerp**

\[ p(t) = p_0 + t(p_1 - p_0) \]
\[ p(t) = (1 - t)p_0 + tp_1 \]

- Linear interpolation looks too harsh
- We’d like something smoother
Continuity

- A function is $C^k$ continuous on a given interval if its $k^{th}$ derivative exists on that interval
  - $C^0$ means the function is continuous, but its tangents are not
  - $C^1$ means tangents are continuous too
  - $C^2$ means curvature is continuous, etc...
SPLINES

- Originally used in drafting and design
- A flexible piece of wood or metal that is constrained to pass through certain points
Splines

• Turns out that the interpolation and smoothness properties of splines can be modeled by piecewise polynomial curves

• Enforcing some constraints at control points can give $C^1$ continuity

• There’s a bunch of different types with different properties

• Cubics are the most common pieces since they minimize accelerations along the curve
Representing cubic curves

\[ p(t) = CT(t) = \begin{bmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix} \]

- Matrix representation: take powers of \( t \) and multiply them by a matrix
- Cubic splines require 4 control points
\[ p(t) = \mathbf{GMT}(t) = \begin{bmatrix} g_1 & g_2 & g_3 & g_4 \end{bmatrix} \begin{bmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ a_3 & b_3 & c_3 & d_3 \\ a_4 & b_4 & c_4 & d_4 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix} \]

- Generally easier to factor \( \mathbf{C} \) into two matrices:
  - Geometry matrix (\( \mathbf{G} \)): contains the control points
  - Basis matrix (\( \mathbf{M} \)): contains weights for those points
Derivative w.r.t. $t$ is just the derivative of the polynomial part, multiplied by the rest of the matrices.

\[ p'(t) = \text{GMT}'(t) = \text{GM} \begin{bmatrix} 0 \\ 1 \\ 2t \\ 3t^2 \end{bmatrix} \]
This case is linear, so only use 1 and $t$
Cubic Hermite curves

- Cubic curve that interpolates two points
- Also matches given tangents at those points
Deriving cubic Hermite curves
Deriving cubic Hermite curves

\[
\begin{bmatrix}
  p_0 & p_1 & m_0 & m_1
\end{bmatrix}
M_H
\begin{bmatrix}
  1 & 1 & 0 & 0 \\
  0 & 1 & 1 & 1 \\
  0 & 1 & 0 & 2 \\
  0 & 1 & 0 & 3
\end{bmatrix}
= \begin{bmatrix}
  p_0 & p_1 & m_0 & m_1
\end{bmatrix}
\]
Deriving cubic Hermite curves

\[
\begin{bmatrix}
p_0 & p_1 & m_0 & m_1
\end{bmatrix}
M_H
\begin{bmatrix}
1 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 \\
0 & 1 & 0 & 2 \\
0 & 1 & 0 & 3
\end{bmatrix}
=
\begin{bmatrix}
p_0 & p_1 & m_0 & m_1
\end{bmatrix}
\]

\[
M_H = \begin{bmatrix}
1 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 \\
0 & 1 & 0 & 2 \\
0 & 1 & 0 & 3
\end{bmatrix}^{-1}
= \begin{bmatrix}
1 & 0 & -3 & 2 \\
0 & 0 & 3 & -2 \\
0 & 1 & -2 & 1 \\
0 & 0 & -1 & 1
\end{bmatrix}
\]
• Multiply $M_H$ by $T(t)$ to get the basis functions for the control points

• These determine how each control point contributes to the curve’s shape
Properties of Hermite curves

- Smoothly interpolates control points
- Matches tangents
  - Connecting two end-to-end will be $C^1$ continuous as long as you set tangents equal
- Not exactly the most intuitive to use
Cubic Bézier curves

- The most common splines in graphics
- Very cool properties
- Used in Adobe Illustrator and much more
Matrix representation

\[ p(t) = G M_B T(t) = \begin{bmatrix} p_0 & p_1 & p_2 & p_3 \end{bmatrix} \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix} \]

- Geometry matrix contains 4 points (no tangents)
- Otherwise very similar to Hermite curves
Cubic Bézier basis functions

- Bézier basis functions sum to 1 at any $t$
- Has a very simple form

$$B_0 = (1-t)^3$$
$$B_1 = 3 (1-t)^2 t$$
$$B_2 = 3 (1-t) t^2$$
$$B_3 = t^3$$
**General Bézier curves**

- Bézier curves can be of any order by generating a basis using Bernstein polynomials:
  \[ B_n^k(t) = \binom{n}{k} t^k (1 - t)^{n-k} \]
- Basis always sums to 1
de Casteljau’s algorithm

• Bézier curves have a cool property: they can be evaluated as a sequence of linear interpolations
  • Means we don’t have to evaluate or even form the polynomials
  • Lerp between each adjacent pair of points
    • Lerp between those intermediate results
    • Keep repeating until you get just one point
Starting with the control points and a given value of $u$
In this example, $u \approx 0.25$
de Casteljau’s

\[ q_0(u) = (1 - u)p_0 + up_1 \]
\[ q_1(u) = (1 - u)p_1 + up_2 \]
\[ q_2(u) = (1 - u)p_2 + up_3 \]
de Casteljau’s

\[
\begin{align*}
\mathbf{r}_0(u) &= (1 - u)\mathbf{q}_0(u) + u\mathbf{q}_1(u) \\
\mathbf{r}_1(u) &= (1 - u)\mathbf{q}_1(u) + u\mathbf{q}_2(u)
\end{align*}
\]
de Casteljau’s

\[ p(u) = (1 - u)r_0(u) + ur_1(u) \]
de Casteljau’s
Bézier properties

• The two exterior points are interpolated
• The two interior points aren’t, but they do define tangents
• The curve will always stay within the convex hull of its control points
• Efficient to evaluate thanks to de Casteljau’s algorithm
Building splines

• Once you have some cubic curves, you can put them together piecewise to create splines

• This will generate a smooth curve from an ordered set of points

• To evaluate: find out which particular curve you’re in, compute $t$, then evaluate the curve
Bézier spline

- Take each group of 4 points and connect with a Bézier curve
- Must be careful to match tangents
Catmull-Rom spline

- Given a set of points, estimate tangents by differencing the neighbors of each point, then connect them together with piecewise cubic Hermite curves
Catmull-Rom formula

\[ CR_i(t) = \left[ \begin{array}{cccc} p_{i-1} & p_i & p_{i+1} & p_{i+2} \end{array} \right] \frac{1}{2} \left[ \begin{array}{cccc} 0 & -1 & 2 & -1 \\ 2 & 0 & -5 & 3 \\ 0 & 1 & 4 & -3 \\ 0 & 0 & -1 & 1 \end{array} \right] \left[ \begin{array}{c} 1 \\ t \\ t^2 \\ t^3 \end{array} \right] \]
Catmull-Rom Spline

- Interpolates the set of control points
- \( C^1 \) continuous everywhere, local support
- First and last point must be left unconnected, since we can't compute tangents for them
B-splines

- Short for “basis spline”
- Works the same way as the others
- Does not interpolate any points or tangents
- \( C^2 \) continuous everywhere
- Locally supported
- I’ll skip the derivation...
B-spline formula

\[ B_i(t) = \left[ \begin{array}{cccc} p_{i-1} & p_i & p_{i+1} & p_{i+2} \end{array} \right] \frac{1}{6} \begin{bmatrix} 1 & -3 & 3 & -1 \\ 4 & 0 & -6 & 3 \\ 1 & 3 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix} \]
Wide basis

- Each point affects a wide, but finite area
What use is that?

- It’s really smooth and locally supported, but doesn’t interpolate anything...
Multiplicities

- Repeating a point will cause a one degree loss of continuity.
- In exchange, the spline matches the control points at that degree - 1.
- 3 repetitions = $C^0$ continuity, points are interpolated.
- 2 repetitions = $C^1$ continuity, tangents interpolated.
Nonuniform B-Splines

• Up until now, we’ve assumed all control points are equally spaced in $t$

• That is, each $t$ is in the range of $[0, 1]$

• We can relax that restriction and put them an arbitrary nonnegative distance apart, even 0

• Squashes or stretches the basis functions, changes speed along the curve
Nonuniform Rational B-Splines

• A.K.A. NURBS
• Add a $w$ coordinate, interpolate that as well
  • $w$ acts as a weight for each control point, making it pull the curve towards it more or less
  • General enough to represent any other type of curve, used all the time in modeling software
FIGURES COURTESY...

- Real-Time Rendering, 3rd ed. [RTR]
  - Tomas Akenine-Moller, Eric Haines, Naty Hoffman
  - Eric Lengyel
  - Edward Angel, Dave Shreiner
- Wikipedia [WP]
- Don Fussell’s CS 384G slides, The University of Texas at Austin [DF]