Lecture 9:
Splines, Bezier Curves & Surfaces

Computer Graphics and Imaging
UC Berkeley CS184/284A, Spring 2017
Smooth Curves and Surfaces

So far we can make:

- Things with corners (lines, triangles, squares, …)
- Specialty shapes (circles, ellipses, …)

Many applications require designed, smooth shapes

- Camera paths, vector fonts, …
- Resampling filter functions
- CAD design, object modeling, …
Camera Paths

Flythrough of proposed Perth Citylink subway, https://youtu.be/rIJMuQPwr3E
Animation Curves

Maya Animation Tutorial: https://youtu.be/b-o5wtZlJPc
Vector Fonts

The Quick Brown Fox Jumps Over The Lazy Dog

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789

Baskerville font - represented as cubic Bézier splines
CAD Design

3D Car Modeling with Rhinoceros
Splines
A Real Draftsman’s Spline

http://www.alatown.com/spline-history-architecture/
Spline Topics

Interpolation

• Cubic Hermite interpolation
• Catmull-Rom interpolation

Bezier curves

Bezier surfaces
Cubic Hermite Interpolation
Goal: Interpolate Values
Nearest Neighbor Interpolation

Problem: values not continuous
Linear Interpolation

Problem: derivatives not continuous
Smooth Interpolation?
Cubic Hermite Interpolation

Inputs: values and derivatives at endpoints
Cubic Polynomial Interpolation

Cubic polynomial

\[ P(t) = a \ t^3 + b \ t^2 + c \ t + d \]

Why cubic?

4 input constraints – need 4 degrees of freedom

\[ P(0) = h_0 \]
\[ P(1) = h_1 \]
\[ P'(0) = h_2 \]
\[ P'(1) = h_3 \]
Cubic Polynomial Interpolation

Cubic polynomial

\[ P(t) = a \ t^3 + b \ t^2 + c \ t + d \]
\[ P'(t) = 3a \ t^2 + 2b \ t + c \]

Set up constraint equations

\[ P(0) = h_0 = d \]
\[ P(1) = h_1 = a + b + c + d \]
\[ P'(0) = h_2 = c \]
\[ P'(1) = h_3 = 3a + 2b + c \]
Solve for Polynomial Coefficients

\[ h_0 = d \]
\[ h_1 = a + b + c + d \]
\[ h_2 = c \]
\[ h_3 = 3a + 2b + c \]

\[
\begin{bmatrix}
  h_0 \\
  h_1 \\
  h_2 \\
  h_3 \\
\end{bmatrix}
= \begin{bmatrix}
  0 & 0 & 0 & 1 \\
  1 & 1 & 1 & 1 \\
  0 & 0 & 1 & 0 \\
  3 & 2 & 1 & 0 \\
\end{bmatrix}
\begin{bmatrix}
  a \\
  b \\
  c \\
  d \\
\end{bmatrix}
\]
Solve for Polynomial Coefficients

\[
\begin{bmatrix}
a \\
b \\
c \\
d \\
\end{bmatrix} = \begin{bmatrix}
0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 \\
3 & 2 & 1 & 0 \\
\end{bmatrix}^{-1} \begin{bmatrix}
h_0 \\
h_1 \\
h_2 \\
h_3 \\
\end{bmatrix}
\]

\[
= \begin{bmatrix}
2 & -2 & 1 & 1 \\
-3 & 3 & -2 & -1 \\
0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 \\
\end{bmatrix} \begin{bmatrix}
h_0 \\
h_1 \\
h_2 \\
h_3 \\
\end{bmatrix}
\]

(Check that these matrices are inverses)
Matrix Form of Hermite Function

\[
P(t) = a \ t^3 + b \ t^2 + c \ t + d
\]

\[
= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}
\]

\[
= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{bmatrix}
\]
Matrix Form of Hermite Function

\[ P(t) = a t^3 + b t^2 + c t + d \]

\[ = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{bmatrix} \]

Matrix rows = coefficient formulas
Matrix Form of Hermite Function

\[ P(t) = a \, t^3 + b \, t^2 + c \, t + d \]

\[ = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{bmatrix} \]

\[ = H_0(t) \, h_0 + H_1(t) \, h_1 + H_2(t) \, h_2 + H_3(t) \, h_3 \]

Matrix columns = Hermite basis functions
Call this matrix the Hermite basis matrix
Hermite Basis Functions

\[ P(t) = a \ t^3 + b \ t^2 + c \ t + d \]

\[ = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{bmatrix} \]

\[ = H_0(t) \ h_0 + H_1(t) \ h_1 + H_2(t) \ h_2 + H_3(t) \ h_3 \]

\[ H_0(t) = 2t^3 - 3t^2 + 1 \]
\[ H_1(t) = -2t^3 + 3t^2 \]
\[ H_2(t) = t^3 - 2t^2 + t \]
\[ H_3(t) = t^3 - t^2 \]
Hermite Basis Functions

\[ P(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} H_0(t) & H_1(t) & H_2(t) & H_3(t) \end{bmatrix} \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{bmatrix} \]

Simply two different sets of “basis functions”
Either basis can represent any cubic polynomial
Hermite Basis Functions

\[ H_0(t) = 2t^3 - 3t^2 + 1 \]
\[ H_1(t) = -2t^3 + 3t^2 \]
\[ H_2(t) = t^3 - 2t^2 + t \]
\[ H_3(t) = t^3 - t^2 \]
Ease Function

A very useful function

In animation, start and stop gently (zero velocity)

\[ H_1(t) = -2t^3 + 3t^2 = t^2(3 - 2t) \]
Hermite Spline Interpolation

Inputs: sequence of values and derivatives
Catmull-Rom Interpolation
Catmull-Rom Interpolation

Inputs: sequence of values

$y_0$, $y_1$, $y_2$, $y_3$
Catmull-Rom Interpolation

Rule for derivatives:
Match slope between previous and next values

\[ \frac{1}{2} (y_3 - y_1) \]
\[ \frac{1}{2} (y_2 - y_0) \]
Catmull-Rom Interpolation

Then use Hermite interpolation

\[ \frac{1}{2}(y_2 - y_0) \quad \frac{1}{2}(y_3 - y_1) \]
Catmull-Rom Spline

Input: sequence of points
Output: spline that interpolates all points with C1 continuity
Interpolating Points & Vectors
Can Interpolate Points As Easily As Values

E.g. point (0,1,3) in 3D space, or even a general vector in N dimensions.

Catmull-Rom 3D spline control points
Can Interpolate Points As Easily As Values

\[
\frac{1}{2}(p_4 - p_2)
\]

\[
\frac{1}{2}(p_3 - p_1)
\]

\[
\frac{1}{2}(p_2 - p_0)
\]

Tangent Vectors

Catmull-Rom 3D tangent vectors
Can Interpolate Points As Easily As Values

\[ \frac{1}{2}(p_4 - p_2) \]

\[ \frac{1}{2}(p_3 - p_1) \]

\[ \frac{1}{2}(p_2 - p_0) \]

Catmull-Rom 3D space curve
Use Basis Functions to Define Curves

General formula for a particular interpolation scheme: \( p(t) = \sum_{i=0}^{n} p_i F_i(t) \)

\[
x(t) = \sum_{i=0}^{n} x_i F_i(t) \quad y(t) = \sum_{i=0}^{n} y_i F_i(t) \quad z(t) = \sum_{i=0}^{n} z_i F_i(t)
\]

Coefficients \( p_i \) can be points & vectors, not just values. \( F_i(t) \) are basis functions for the interpolation scheme.

Saw \( H_i(t) \) for Hermite interpolation earlier. Will see \( C_i(t) \) for Catmull-Rom shortly, and \( B_i(t) \) for Bézier scheme later. The basis functions are properties of the interpolation scheme.
Matrix Form of Catmull-Rom Space Curve?

Use Hermite matrix form

- Points & tangents given by Catmull-Rom rules

Hermite points

\[ h_0 = p_1 \]
\[ h_1 = p_2 \]
\[ h_2 = \frac{1}{2}(p_2 - p_0) \]
\[ h_3 = \frac{1}{2}(p_3 - p_1) \]

Hermite tangents

\[
\begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}^T \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}
\]
Matrix Form of Catmull-Rom Space Curve

\[
P(t) = \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}^T \begin{bmatrix} -\frac{1}{2} & \frac{3}{2} & \frac{3}{2} & \frac{1}{2} \\ 1 & -\frac{5}{2} & 2 & -\frac{1}{2} \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}
\]

\[= C_0(t) \ p_0 + C_1(t) \ p_1 + C_2(t) \ p_2 + C_3(t) \ p_3\]

Matrix columns = Catmull-Rom basis functions
Catmull-Rom Basis Functions
Bézier Curves
Examples of Geometry
Defining Cubic Bézier Curve With Tangents

\[ t_0 = 3(p_1 - p_0) \]

\[ t_1 = 3(p_3 - p_2) \]
Matrix Form of Cubic Bézier Curve?

\[ P(t) = \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}^T \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} = B_0^3(t) \ p_0 + B_1^3(t) \ p_1 + B_2^3(t) \ p_2 + B_3^3(t) \ p_3 \]

Good exercise to derive this matrix yourself. One way: use Hermite matrix equation again. What are the points and tangents?
Demo – Piecewise Cubic Bézier Curve

Evaluating Bézier Curves
De Casteljau Algorithm
Bézier Curves – de Casteljau Algorithm

Consider three points (quadratic Bezier)

\[ b_0, b_1, b_2 \]
Bézier Curves – de Casteljau Algorithm

Insert a point using linear interpolation

\[ b_0 \rightarrow t \rightarrow (1 - t) \rightarrow b_1 \rightarrow b_2 \]

Pierre Bézier
1910 – 1999

Paul de Casteljau
b. 1930
Bézier Curves – de Casteljau Algorithm

Insert on both edges

\[ b_0, b_1, b_2 \]

\[ b_0^1, t, b_1^1 \]

\[ (1 - t) \]

\[ 0, t, 1 \]

Pierre Bézier
1910 – 1999

Paul de Casteljau
b. 1930
Bézier Curves – de Casteljau Algorithm

Repeat recursively

\[ (1 - t)b_0^1 + tb_1^1 + tb_2^1 = b_1 \]

Pierre Bézier
1910 – 1999

Paul de Casteljau
b. 1930
Bézier Curves – de Casteljau Algorithm

Algorithm defines the curve

“Corner cutting” recursive subdivision

Successive linear interpolation

Pierre Bézier
1910 – 1999

Paul de Casteljau
b. 1930
Visualizing de Casteljau Algorithm

Cubic Bezier Curve

Cubic Bézier Curve – de Casteljau

Consider four points
Same recursive linear interpolations
de Casteljau Algorithm Subdivides Curve

Control polygon for full curve

Control polygon for left curve

Control polygon for right curve
Evaluating Bézier Curves
Algebraic Formula
Bézier Curve – Algebraic Formula

de Casteljau algorithm gives a pyramid of coefficients
Bézier Curve – Algebraic Formula

Example: quadratic Bézier curve from three points

\[ b_0^1(t) = (1 - t)b_0 + tb_1 \]
\[ b_1^1(t) = (1 - t)b_1 + tb_2 \]
\[ b_0^2(t) = (1 - t)b_0^1 + tb_1^1 \]
\[ b_0^2(t) = (1 - t)^2b_0 + 2t(1 - t)b_1 + t^2b_2 \]
Bézier Curve – General Algebraic Formula

Bernstein form of a Bézier curve of order $n$:

$$b^n(t) = b^n_0(t) = \sum_{j=0}^{n} b_j B^n_j(t)$$

Bernstein polynomials:

$$B^n_i(t) = \binom{n}{i} t^i (1 - t)^{n-i}$$

Bézier control points
Cubic Bézier Basis Functions

Bernstein polynomials:

\[ B_i^n(t) = \binom{n}{i} t^i (1 - t)^{n-i} \]
Piecewise Bézier Curves
(Bézier Spline)
Higher-Order Bézier Curves?

High-degree Bernstein polynomials don’t interpolate well

Very hard to control!

Uncommon

\[ n = 10 \]
Piecewise Bézier Curves

Instead, chain many low-order Bézier curve

Piecewise cubic Bézier the most common technique

Widely used (fonts, paths, Illustrator, Keynote, …)
Piecewise Bézier Curve – Continuity

Two Bézier curves

\[ a : [k, k + 1] \rightarrow \mathbb{R}^N \]
\[ b : [k + 1, k + 2] \rightarrow \mathbb{R}^N \]

Assuming integer partitions here, can generalize
Piecewise Bézier Curve – Continuity

$C^0$ continuity: \[ a_n = b_0 \]
$C^1$ continuity: \[ a_n = b_0 = \frac{1}{2} (a_{n-1} + b_1) \]
C² continuity: “A-frame” construction
Properties of Bézier Curves

Interpolates endpoints

• For cubic Bézier: \( b(0) = b_0; \quad b(1) = b_3 \)

Tangent to end segments

• Cubic case: \( b'(0) = 3(b_1 - b_0); \quad b'(1) = 3(b_3 - b_2) \)

Affine transformation property

• Transform curve by transforming control points

Convex hull property

• Curve is within convex hull of control points
Bézier Surfaces
Bézier Surfaces

Extend Bézier curves to surfaces

Ed Catmull’s “Gumbo” model

Utah Teapot

P. Rideout
Bicubic Bézier Surface Patch

Bezier surface and 4 x 4 array of control points
Visualizing Bicubic Bézier Surface Patch

Visualizing Bicubic Bézier Surface Patch

4x4 control points

- Each 4x1 control points in u define a Bezier curve
  - (4 Bezier curves in u)

- Corresponding points on these 4 Bezier curves define 4 control points for a “moving curve” in v
  - This “moving” curve sweeps out the 2D surface
Evaluating Bézier Surfaces
Evaluating Surface Position For Parameters \((u,v)\)

For bi-cubic Bezier surface patch, 
Input: 4x4 control points 
Output is 2D surface parameterized by \((u,v)\) in \([0,1]^2\)
Method 1: Separable 1D de Casteljau Algorithm

Goal: Evaluate surface position corresponding to \((u,v)\)

\((u,v)\)-separable application of de Casteljau algorithm

- Use de Casteljau to evaluate point \(u\) on each of the 4 Bezier curves in \(u\). This gives 4 control points for the “moving” Bezier curve
- Use 1D de Casteljau to evaluate point \(v\) on the “moving” curve

![Diagram showing Bezier curves and a surface point](Image)
Method 1: Separable 1D de Casteljau Algorithm
Method 2: 2D de Casteljau Algorithm

Repeated application of bilinear interpolation
Method 2: 2D de Casteljau Algorithm

Example: \((u, v) = \left(\frac{1}{2}, \frac{1}{2}\right)\)

\[
\begin{bmatrix}
0 & 2 & 4 \\
0 & 0 & 0 \\
0 & 2 & 4 \\
0 & 4 & 4 \\
4 & 2 & 0 \\
0 & 0 & 2 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{bmatrix}
\rightarrow
\begin{bmatrix}
1 & 3 \\
1 & 0 \\
1 & 1 \\
1 & 3 \\
1 & 2.5 \\
0.5 & 3 \\
1 & 3 \\
0 & 3
\end{bmatrix}
\rightarrow
\begin{bmatrix}
2 \\
2 \\
1
\end{bmatrix}
\]

\(r = 1\) \hspace{2cm} \(r = 2\) \hspace{2cm} \(r = 3\)
Method 3: Algebraic Evaluation

Let the moving curve be a degree $m$ Bézier curve

$$b^m(u) = \sum_{i=0}^{m} b_i B_i^m(u)$$

(remember, Bernstein polynomials)

$$B_i^n(t) = \binom{n}{i} t^i (1 - t)^{n-i}$$

Let each control point $b_i$ be moving along a Bézier curve of degree $n$

$$b_i = b_i(v) = \sum_{j=0}^{n} b_{i,j} B_j^n(v)$$

Tensor product Bézier patch

$$b^{m,n}(u, v) = \sum_{i=0}^{m} \sum_{j=0}^{n} b_{i,j} B_i^m(u) B_j^n(v)$$
Bézier Surface Continuity
Piecewise Bézier Surfaces

$C^0$ continuity: Boundary curves
Piecewise Bézier Surfaces

$C^1$ continuity: Collinearity
Piecewise Bézier Surfaces

$C^2$ continuity: A-frames
Things to Remember

Splines

• Cubic Hermite and Catmull-Rom interpolation
• Matrix representation of cubic polynomials

Bézier curves

• Easy-to-control spline
• Recursive linear interpolation – de Casteljau algorithm
• Properties of Bézier curves
• Piecewise Bézier curve – continuity types and how to achieve

Bézier surfaces

• Bicubic Bézier patches – tensor product surface
• 2D de Casteljau algorithm
Acknowledgments

Thanks to Pat Hanrahan, Mark Pauly and Steve Marschner for presentation resources.