CSE 167:  
Introduction to Computer Graphics  
Lecture #21: Bezier Curves

Jürgen P. Schulze, Ph.D.  
University of California, San Diego  
Fall Quarter 2020
Announcements

- **Sunday, December 13th at 11:59pm:**
  - Homework Project 4 due
- **Thursday, Dec 17th 2:30pm until Dec 18th 2:30pm**
  - Final Exam
  - Timed 3-hour Canvas quiz, to be taken within 24h
- **Sunday, December 20th at 11:59pm:**
  - Homework Project 4 late deadline
Lecture Overview

- Polynomial Curves
  - Introduction
  - Polynomial functions

- Bézier Curves
  - Introduction
  - Drawing Bézier curves
  - Piecewise Bézier curves
Modeling

- Creating 3D objects
- How to construct complex surfaces?
- Goal
  - Specify objects with control points
  - Objects should be visually pleasing (smooth)
- Start with curves, then surfaces
Curves

- Surface of revolution
Curves

- Extruded/swept surfaces
Curves

- **Animation**
  - Provide a “track” for objects
  - Use as camera path
Video

- Bezier Curves
  - [http://www.youtube.com/watch?v=hlDYJNEiYvU](http://www.youtube.com/watch?v=hlDYJNEiYvU)
Curves

- Can be generalized to surface patches
Curve Representation

Why not specify many points along a curve and connect with lines:

- Can’t get smooth results when magnified – more points needed
- Large storage and CPU requirements

Instead: specify a curve with a small number of “control points”

- Known as a *spline curve* or *spline*. 

---

![control point diagram]
Spline: Definition

Wikipedia:

- Term comes from flexible spline devices used by shipbuilders and draftsmen to draw smooth shapes.
- Spline consists of a long strip fixed in position at a number of points that relaxes to form a smooth curve passing through those points.
Lecture Overview

- Polynomial Curves
  - Introduction
  - Polynomial functions
- Bézier Curves
  - Introduction
  - Drawing Bézier curves
  - Piecewise Bézier curves
Interpolating Control Points

- “Interpolating” means that curve goes through all control points
- Seems most intuitive
- But hard to control exact behavior
Approximating Control Points

- Curve is “influenced” by control points

- Various types
  - Most common: polynomial functions
    - Bézier spline (our focus)
    - B-spline (generalization of Bézier spline)
    - NURBS (Non Uniform Rational Basis Spline): used in CAD tools
A vector valued function of one variable \( \mathbf{x}(t) \)

- Given \( t \), compute a 3D point \( \mathbf{x}=(x,y,z) \)
- Could be interpreted as three functions: \( x(t), y(t), z(t) \)
- Parameter \( t \) “moves a point along the curve”
Tangent Vector

- Derivative $\mathbf{x}'(t) = \frac{d\mathbf{x}}{dt} = (x'(t), y'(t), z'(t))$

- Vector $\mathbf{x}'$:
  - Points in direction of movement
  - Length corresponds to speed
Lecture Overview

- Polynomial Curves
  - Introduction
  - Polynomial functions

- Bézier Curves
  - Introduction
  - Drawing Bézier curves
  - Piecewise Bézier curves
Polynomial Functions

- **Linear:** \[ f(t) = at + b \] (1\textsuperscript{st} order)

- **Quadratic:** \[ f(t) = at^2 + bt + c \] (2\textsuperscript{nd} order)

- **Cubic:** \[ f(t) = at^3 + bt^2 + ct + d \] (3\textsuperscript{rd} order)
Polynomial Curves in 3D

- Linear: \( \mathbf{x}(t) = \mathbf{a}t + \mathbf{b} \)
  \[
  \mathbf{x} = (x, y, z), \quad \mathbf{a} = (a_x, a_y, a_z), \quad \mathbf{b} = (b_x, b_y, b_z)
  \]

- Evaluated as:
  \[
  \begin{align*}
  x(t) &= a_xt + b_x \\
  y(t) &= a_yt + b_y \\
  z(t) &= a_zt + b_z
  \end{align*}
  \]
Polynomial Curves in 3D

- **Quadratic:** $x(t) = at^2 + bt + c$ (2nd order)

- **Cubic:** $x(t) = at^3 + bt^2 + ct + d$ (3rd order)

- We usually define the curve for $0 \leq t \leq 1$
Control Points

- Polynomial coefficients $a, b, c, d$ can be interpreted as control points
  - Remember: $a, b, c, d$ have $x,y,z$ components each
  - But: they do not intuitively describe the shape of the curve
- Goal: intuitive control points
Weighted Average

- Based on linear interpolation (LERP)
  - Weighted average between two values
  - “Value” could be a number, vector, color, ...
- Interpolate between points \( p_0 \) and \( p_1 \) with parameter \( t \)
  - Defines a “curve” that is straight (first-order spline)

\[
x(t) = Lerp(t, p_0, p_1) = (1-t)p_0 + t \ p_1
\]
Linear Polynomial

\[ x(t) = \left(\vec{p}_1 - \vec{p}_0\right) t + \vec{p}_0 \]

- Curve is based at point \( \vec{p}_0 \)
- Add the vector, scaled by \( t \)
Matrix Form

\[ x(t) = \begin{bmatrix} p_0 & p_1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} t \\ 1 \end{bmatrix} = GBT \]

- **Geometry matrix**
  \[ G = \begin{bmatrix} p_0 & p_1 \end{bmatrix} \]

- **Geometric basis**
  \[ B = \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \]

- **Polynomial basis**
  \[ T = \begin{bmatrix} t \\ 1 \end{bmatrix} \]

- **In components**
  \[ x(t) = \begin{bmatrix} p_{0x} & p_{1x} \\ p_{0y} & p_{1y} \\ p_{0z} & p_{1z} \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} t \\ 1 \end{bmatrix} \]
Summary

1. Grouped by points \( p \): weighted average

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

2. Grouped by \( t \): linear polynomial

\[
x(t) = (p_1 - p_0)t + p_0
\]

3. Matrix form:

\[
x(t) = \begin{bmatrix} p_0 & p_1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} t \\ 1 \end{bmatrix}
\]
Tangent

- Weighted average \( x'(t) = (-1)p_0 + (+1)p_1 \)
- Polynomial \( x'(t) = 0t + (p_1 - p_0) \)
- Matrix form \( x'(t) = \begin{bmatrix} p_0 & p_1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \)
Lecture Overview

- Polynomial Curves
  - Introduction
  - Polynomial functions
- Bézier Curves
  - Introduction
  - Drawing Bézier curves
  - Piecewise Bézier curves
Bézier Curves

- Invented by Pierre Bézier in the 1960s for designing curves for the bodywork of Renault cars
- Are a higher order extension of linear interpolation
- Give intuitive control over curve with control points
  - Endpoints are interpolated, intermediate points are approximated

![Bézier Curves Diagram]

Linear

Quadratic

Cubic
Cubic Bézier Curve

- Most commonly used case
- Defined by four control points:
  - Two interpolated endpoints (points are on the curve)
  - Two points control the tangents at the endpoints
- Points $x$ on curve defined as function of parameter $t$
Demo

Algorithmic Construction

- Algorithmic construction
  - *De Casteljau* algorithm, developed at Citroen in 1959, named after its inventor Paul de Casteljau (pronounced “Cast-all-’Joe”)
  - Developed independently from Bézier’s work: Bézier created the formulation using blending functions, Casteljau devised the recursive interpolation algorithm
De Casteljau Algorithm

- A recursive series of linear interpolations
  - Works for any order Bezier function, not only cubic
- Not very efficient to evaluate
  - Other forms more commonly used
- But:
  - Gives intuition about the geometry
  - Useful for subdivision
De Casteljau Algorithm

- Given:
  - Four control points
  - A value of \( t \) (here \( t \approx 0.25 \))
De Casteljau Algorithm

\[ q_0(t) = \text{Lerp}(t, p_0, p_1) \]
\[ q_1(t) = \text{Lerp}(t, p_1, p_2) \]
\[ q_2(t) = \text{Lerp}(t, p_2, p_3) \]
De Casteljau Algorithm

\[ r_0(t) = \text{Lerp}(t, q_0(t), q_1(t)) \]

\[ r_1(t) = \text{Lerp}(t, q_1(t), q_2(t)) \]
De Casteljau Algorithm

\[ x(t) = \text{Lerp}(t, r_0(t), r_1(t)) \]
De Casteljau Algorithm

- Demo
  - https://www.jasondavies.com/animated-bezier/
Recursive Linear Interpolation

\[ x = \text{Lerp}(t, r_0, r_1) \]

\[ r_0 = \text{Lerp}(t, q_0, q_1) \quad q_0 = \text{Lerp}(t, p_0, p_1) \]

\[ r_1 = \text{Lerp}(t, q_1, q_2) \quad q_1 = \text{Lerp}(t, p_1, p_2) \]

\[ q_2 = \text{Lerp}(t, p_2, p_3) \]
Expand the LERPs

\[ q_0(t) = Lerp(t, p_0, p_1) = (1 - t)p_0 + tp_1 \]
\[ q_1(t) = Lerp(t, p_1, p_2) = (1 - t)p_1 + tp_2 \]
\[ q_2(t) = Lerp(t, p_2, p_3) = (1 - t)p_2 + tp_3 \]

\[ r_0(t) = Lerp(t, q_0(t), q_1(t)) = (1 - t)((1 - t)p_0 + tp_1) + t((1 - t)p_1 + tp_2) \]
\[ r_1(t) = Lerp(t, q_1(t), q_2(t)) = (1 - t)((1 - t)p_1 + tp_2) + t((1 - t)p_2 + tp_3) \]

\[ x(t) = Lerp(t, r_0(t), r_1(t)) \]
\[ = (1 - t)((1 - t)((1 - t)p_0 + tp_1) + t((1 - t)p_1 + tp_2)) + t((1 - t)((1 - t)p_1 + tp_2) + t((1 - t)p_2 + tp_3)) \]
Weighted Average of Control Points

- Regroup for $p$:
  \[
x(t) = (1-t)((1-t)(p_0 + tp_1) + t((1-t)p_1 + tp_2)) + t((1-t)p_1 + tp_2) + t((1-t)p_2 + tp_3))
  \]
  \[
x(t) = (1-t)^3 p_0 + 3(1-t)^2 tp_1 + 3(1-t)t^2 p_2 + t^3 p_3
  \]
  \[
x(t) = B_0(t)p_0 + B_1(t)tp_1 + B_2(t)t^2 p_2 + B_3(t)t^3 p_3
  \]
**Cubic Bernstein Polynomials**

\[ \mathbf{x}(t) = B_0(t)\mathbf{p}_0 + B_1(t)\mathbf{p}_1 + B_2(t)\mathbf{p}_2 + B_3(t)\mathbf{p}_3 \]

The cubic *Bernstein polynomials*:

\[
\begin{align*}
B_0(t) &= -t^3 + 3t^2 - 3t + 1 \\
B_1(t) &= 3t^3 - 6t^2 + 3t \\
B_2(t) &= -3t^3 + 3t^2 \\
B_3(t) &= t^3
\end{align*}
\]

\[ \sum B_i(t) = 1 \]

- Weights \( B_i(t) \) add up to 1 for any value of \( t \)
General Bernstein Polynomials

\[ B_0^1(t) = -t + 1 \]
\[ B_1^1(t) = t \]
\[ B_2^1(t) = t^2 \]

\[ B_0^2(t) = t^2 - 2t + 1 \]
\[ B_1^2(t) = -2t^2 + 2t \]
\[ B_2^2(t) = t^2 \]

\[ B_0^3(t) = -t^3 + 3t^2 - 3t + 1 \]
\[ B_1^3(t) = 3t^3 - 6t^2 + 3t \]
\[ B_2^3(t) = -3t^3 + 3t^2 \]
\[ B_3^3(t) = t^3 \]

\[ B_i^n(t) = \binom{n}{i} (1-t)^{n-i} t^i \]
\[ \sum B_i^n(t) = 1 \]

\[ \binom{n}{i} = \frac{n!}{i!(n-i)!} \]

n! = factorial of n
(n+1)! = n! x (n+1)
Any order Bézier Curves

nth-order Bernstein polynomials form nth-order Bézier curves

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

\[ x(t) = \sum_{i=0}^{n} B_i^n(t) p_i \]
Demo: Bezier handles

http://math.hws.edu/eck/cs424/notes2013/canvas/bezier.html
Rational Curves

- Weight causes point to “pull” more (or less)
- Can model circles with proper points and weights,
- Below: rational quadratic Bézier curve (three control points)
B-Splines

- B as in Basis-Splines
- Basis is blending function
- Difference to Bézier blending function:
  - B-spline blending function can be zero outside a particular range (limits scope over which a control point has influence)
- B-Spline is defined by control points and range in which each control point is active.
NURBS

- **Non Uniform Rational B-Splines**
- Generalization of Bézier curves
- Non uniform:
- Combine B-Splines (limited scope of control points) and Rational Curves (weighted control points)
- Can exactly model conic sections (circles, ellipses)
- **OpenGL support:** see `gluNurbsCurve`
- **Demos:**
  - [http://bentonian.com/teaching/AdvGraph0809/demos/Nurbs2d/index.html](http://bentonian.com/teaching/AdvGraph0809/demos/Nurbs2d/index.html)
where do we go from here?
Computer Graphics Courses 2020/21

Winter:
- CSE 167: Computer Graphics (?)
- CSE 165: 3D User Interfaces (Schulze)
- CSE 169: Computer Animation (Rotenberg?)

Spring:
- CSE 168: Computer Graphics II: Rendering (Ramamoorthi)
- CSE 190: Virtual Reality Technologies (Schulze)
- CSE 163: Advanced Computer Graphics (not this year)
Computer Graphics: State of The Art

- ACM SIGGRAPH Asia 2020 Technical Papers Trailer
  - https://www.youtube.com/watch?v=Q45KT0lGd7A

- Unreal Engine 5 Feature Highlights
  - https://www.youtube.com/watch?v=EFyWEMe27Dw

- Cyberpunk 2077 — Official Launch Trailer
  - https://www.youtube.com/watch?v=UnA7tepsc7s
Good luck with your final exams!