Announcements

- Homework project #6 due Friday, Nov 18 at 1:30pm
  - To be presented in lab 260
- No grading this Friday, Nov 11 (Veterans Day)
  - Last day for late grading project 5: Thursday, Nov 10

- Final project presentations date+time confirmed: December 2nd, 1-3pm, CSE 1202
Lecture Overview

- Polynomial Curves cont’d
- Bézier Curves
  - Introduction
  - Drawing Bézier curves
  - Piecewise Bézier curves
Polynomial Curves

- **Quadratic:** \( x(t) = at^2 + bt + c \)  
  (2\(^\text{nd}\) order)

- **Cubic:** \( x(t) = at^3 + bt^2 + ct + d \)  
  (3\(^\text{rd}\) 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
- Unfortunately, they do not intuitively describe the shape of the curve
- Goal: intuitive control points
Control Points

- How many control points?
  - Two points define a line ($1^{st}$ order)
  - Three points define a quadratic curve ($2^{nd}$ order)
  - Four points define a cubic curve ($3^{rd}$ order)
  - $k+1$ points define a $k$-order curve

- Let’s start with a line…
First Order Curve

- 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)
  - $t=0$ corresponds to $p_0$
  - $t=1$ corresponds to $p_1$
  - $t=0.5$ corresponds to midpoint

$$x(t) = Lerp(t, p_0, p_1) = (1 - t)p_0 + t \ p_1$$
Linear Interpolation

- Three equivalent ways to write it
  - Expose different properties

1. Regroup for points $p \rightarrow$ weighted sum of control points
   \[
   x(t) = p_0(1-t) + p_1t
   \]

2. Regroup for $t \rightarrow$ polynomial in $t$
   \[
   x(t) = (p_1 - p_0)t + p_0
   \]

3. Matrix form
   \[
   x(t) = \begin{bmatrix} p_0 & p_1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} t \\ 1 \end{bmatrix}
   \]
Weighted Average

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

\[ = B_0(t) p_0 + B_1(t)p_1, \text{ where } B_0(t) = 1 - t \text{ and } B_1(t) = t \]

- Weights are a function of \( t \)
  - Sum is always 1, for any value of \( t \)
  - Also known as *blending functions*
Linear Polynomial

\[ x(t) = \underbrace{(p_1 - p_0)}_{\text{vector} \ a} t + \underbrace{p_0}_{\text{point} \ b} \]

- Curve is based at point \( p_0 \)
- Add the vector, scaled by \( t \)
Matrix Form

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

- Geometry matrix \( \mathbf{G} = \begin{bmatrix} p_0 & p_1 \end{bmatrix} \)
- Geometric basis \( \mathbf{B} = \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \)
- Polynomial basis \( \mathbf{T} = \begin{bmatrix} t \\ 1 \end{bmatrix} \)
- In components

\[ \mathbf{x}(t) = \begin{bmatrix} p_{0x} & p_{1x} \\ p_{0y} & p_{1y} \\ p_{0z} & p_{1z} \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} t \\ 1 \end{bmatrix} \]
Tangent

- For a straight line, the tangent is constant
  \[ x'(t) = p_1 - p_0 \]

- 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 cont’d
- Bézier Curves
  - Introduction
  - Drawing Bézier curves
  - Piecewise Bézier curves
Bézier Curves

- Are a higher order extension of linear interpolation
Bézier Curves

- Give intuitive control over curve with control points
  - Endpoints are interpolated, intermediate points are approximated
  - Convex Hull property
  - Variation-Diminishing property

- Many demo applets online, for example:
  - [http://www.sunsite.ubc.ca/LivingMathematics/V001N01/UBCExamples/Bezier/bezier.html](http://www.sunsite.ubc.ca/LivingMathematics/V001N01/UBCExamples/Bezier/bezier.html)
Cubic Bézier Curve

- Most common 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 \( t \)
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

\[ q_0(t) = Lerp(t, p_0, p_1) \]
\[ q_1(t) = Lerp(t, p_1, p_2) \]
\[ q_2(t) = Lerp(t, p_2, p_3) \]
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

Applets
- Demo: [http://www2.mat.dtu.dk/people/J.Gravesen/cagd/decast.html](http://www2.mat.dtu.dk/people/J.Gravesen/cagd/decast.html)
Recursive Linear Interpolation

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

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

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

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

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

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

\[ p_0 \]

\[ p_1 \]

\[ p_2 \]

\[ p_3 \]
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 \( p \):

\[
x(t) = (1 - t) \left( (1 - t) \left( (1 - t)p_0 + tp_1 \right) + t \left( (1 - t)p_1 + tp_2 \right) \right) + t \left( (1 - t) \left( (1 - t)p_1 + tp_2 \right) + t \left( (1 - t)p_2 + tp_3 \right) \right)
\]

\[
x(t) = (1 - t)^3 p_0 + 3(1 - t)^2 tp_1 + 3(1 - t)t^2 p_2 + t^3 p_3
\]
Cubic Bernstein Polynomials

\[ x(t) = B_0(t)p_0 + B_1(t)p_1 + B_2(t)p_2 + B_3(t)p_3 \]

The cubic Bernstein polynomials:

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

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

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

\( B_0^1(t) = -t + 1 \)
\( B_1^1(t) = t \)

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

\( B_0^3(t) = -t^3 + 3t^2 - 3t + 1 \)
\( B_3^3(t) = t^3 \)

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

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

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

\( n! = \text{factorial of } n \)
\( (n+1)! = n! \times (n+1) \)
General Bézier Curves

- $n$th-order Bernstein polynomials form $n$th-order Bézier curves

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

\[ x(t) = \sum_{i=0}^{n} B_i^n(t) p_i \]
Bézier Curve Properties

Overview:
- Convex Hull property
- Variation Diminishing property
- Affine Invariance
Definitions

- **Convex hull** of a set of points:
  - Polyhedral volume created such that all lines connecting any two points lie completely inside it (or on its boundary)

- **Convex combination** of a set of points:
  - Weighted average of the points, where all weights between 0 and 1, sum up to 1

- Any convex combination always of a set of points lies within the convex hull
Convex Hull Property

- A Bézier curve is a convex combination of the control points (by definition, see Bernstein polynomials)
- Bézier curve is always inside the convex hull
  - Makes curve predictable
  - Allows culling, intersection testing, adaptive tessellation
Variation Diminishing Property

- If the curve is in a plane, this means no straight line intersects a Bézier curve more times than it intersects the curve's control polyline
- “Curve is not more wiggly than control polyline”
Affine Invariance

Transforming Bézier curves

- Two ways to transform:
  - Transform the control points, then compute resulting spline points
  - Compute spline points then transform them
- Either way, we get the same points
  - Curve is defined via affine combination of points
  - Invariant under affine transformations
  - Convex hull property remains true
Cubic Polynomial Form

Start with Bernstein form:

\[ x(t) = (-t^3 + 3t^2 - 3t + 1)p_0 + (3t^3 - 6t^2 + 3t)p_1 + (-3t^3 + 3t^2)p_2 + (t^3)p_3 \]

Regroup into coefficients of \( t \):

\[ x(t) = (-p_0 + 3p_1 - 3p_2 + p_3)t^3 + (3p_0 - 6p_1 + 3p_2)t^2 + (-3p_0 + 3p_1)t + (p_0)l \]

\[
\begin{align*}
  x(t) & = at^3 + bt^2 + ct + d \\
  a &= (-p_0 + 3p_1 - 3p_2 + p_3) \\
  b &= (3p_0 - 6p_1 + 3p_2) \\
  c &= (-3p_0 + 3p_1) \\
  d &= (p_0)
\end{align*}
\]

- Good for fast evaluation
  - Precompute constant coefficients \((a,b,c,d)\)
- Not much geometric intuition
Cubic Matrix Form

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

\[ \bar{a} = (-p_0 + 3p_1 - 3p_2 + p_3) \]
\[ \bar{b} = (3p_0 - 6p_1 + 3p_2) \]
\[ \bar{c} = (-3p_0 + 3p_1) \]
\[ d = (p_0) \]

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

- Other types of cubic splines use different basis matrices \( B_{\text{Bez}} \)
Cubic Matrix Form

- In 3D: 3 parallel equations for x, y and z:

\[
x_x(t) = \begin{bmatrix} p_{0x} & p_{1x} & p_{2x} & p_{3x} \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}
\]

\[
x_y(t) = \begin{bmatrix} p_{0y} & p_{1y} & p_{2y} & p_{3y} \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}
\]

\[
x_z(t) = \begin{bmatrix} p_{0z} & p_{1z} & p_{2z} & p_{3z} \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix}
\]
Matrix Form

- Bundle into a single matrix

\[
x(t) = \begin{bmatrix}
p_{0x} & p_{1x} & p_{2x} & p_{3x} \\
p_{0y} & p_{1y} & p_{2y} & p_{3y} \\
p_{0z} & p_{1z} & p_{2z} & p_{3z}
\end{bmatrix} \begin{bmatrix}
-1 & 3 & -3 & 1 \\
3 & -6 & 3 & 0 \\
-3 & 3 & 0 & 0 \\
1 & 0 & 0 & 0
\end{bmatrix} \begin{bmatrix}
t^3 \\
t^2 \\
t \\
1
\end{bmatrix}
\]

- Efficient evaluation
  - Pre-compute \( C \)
  - Take advantage of existing 4x4 matrix hardware support
Lecture Overview

- Polynomial Curves cont’d
- Bézier Curves
  - Introduction
  - Drawing Bézier curves
  - Piecewise Bézier curves
Drawing Bézier Curves

- Draw *line segments* or individual pixels
- Approximate the curve as a series of line segments *(tessellation)*
  - Uniform sampling
  - Adaptive sampling
  - Recursive subdivision
Uniform Sampling

- Approximate curve with $N$ straight segments
  - $N$ chosen in advance
  - Evaluate $x_i = x(t_i)$ where $t_i = \frac{i}{N}$ for $i = 0, 1, \ldots, N$
  
  $$x_i = \tilde{a} \frac{i^3}{N^3} + \tilde{b} \frac{i^2}{N^2} + \tilde{c} \frac{i}{N} + d$$

- Connect the points with lines

- Too few points?
  - Poor approximation
  - “Curve” is faceted

- Too many points?
  - Slow to draw too many line segments
  - Segments may draw on top of each other
Adaptive Sampling

- Use only as many line segments as you need
  - Fewer segments where curve is mostly flat
  - More segments where curve bends
  - Segments never smaller than a pixel
Recursive Subdivision

- Any cubic curve segment can be expressed as a Bézier curve
- Any piece of a cubic curve is itself a cubic curve
- Therefore:
  - Any Bézier curve can be broken down into smaller Bézier curves
De Casteljau Subdivision

- De Casteljau construction points are the control points of two Bézier sub-segments.
Adaptive Subdivision Algorithm

- Use De Casteljau construction to split Bézier segment
- For each half
  - If “flat enough”: draw line segment
  - Else: recurse
- Curve is flat enough if hull is flat enough
- Test how far the approximating control points are from a straight segment
  - If less than one pixel, the hull is flat enough
Drawing Bézier Curves With OpenGL

- Indirect OpenGL support for drawing curves:
  - Define evaluator map \((\text{glMap})\)
  - Draw line strip by evaluating map \((\text{glEvalCoord})\)
  - Optimize by pre-computing coordinate grid \((\text{glMapGrid} \text{ and } \text{glEvalMesh})\)

- More details about OpenGL implementation:
  - [http://www.cs.duke.edu/courses/fall09/cps124/notes/12_curves/opengl_nurbs.pdf](http://www.cs.duke.edu/courses/fall09/cps124/notes/12_curves/opengl_nurbs.pdf)
Lecture Overview

- Polynomial Curves cont’d
- Bézier Curves
  - Introduction
  - Drawing Bézier curves
  - Piecewise Bézier curves
More Control Points

- Cubic Bézier curve limited to 4 control points
  - Cubic curve can only have one inflection (point where curve changes direction of bending)
  - Need more control points for more complex curves
- $k-1$ order Bézier curve with $k$ control points

- Hard to control and hard to work with
  - Intermediate points don’t have obvious effect on shape
  - Changing any control point changes the whole curve
  - Want local support: each control point only influences nearby portion of curve
Piecewise Curves

- Sequence of simple (low-order) curves, end-to-end
  - Known as a *piecewise polynomial curve*
- Sequence of line segments
  - *Piecewise linear* curve

- Sequence of cubic curve segments
  - *Piecewise cubic* curve (here piecewise Bézier)
Mathematical Continuity

- **$C_0$ continuity:**
  - Curve segments are connected

- **$C_1$ continuity:**
  - $C_0$ & 1st-order derivatives agree at joints
  - Tangents/normals are $C_0$ continuous
  - Important for smooth shading

- **$C_2$ continuity:**
  - $C_1$ & 2nd-order derivatives agree at joints
  - Tangents/normals are $C_1$ continuous
  - Important for high quality reflections
Global Parameterization

- Given $N$ curve segments $x_0(t), x_1(t), \ldots, x_{N-1}(t)$
- Each is parameterized for $t$ from 0 to 1
- Define a piecewise curve
  - Global parameter $u$ from 0 to $N$
    - $x(u) = \begin{cases} x_0(u), & 0 \leq u \leq 1 \\ x_1(u-1), & 1 \leq u \leq 2 \\ \vdots & \vdots \\ x_{N-1}(u-(N-1)), & N-1 \leq u \leq N \end{cases}$
    - $x(u) = x_i(u-i), \text{ where } i = \lfloor u \rfloor$ (and $x(N) = x_{N-1}(1)$)
- Alternate: solution $u$ also goes from 0 to 1
  - $x(u) = x_i(Nu-i), \text{ where } i = \lfloor Nu \rfloor$
Piecewise-Linear Curve

- Given $N+1$ points $p_0, p_1, \ldots, p_N$
- Define curve
  \[ x(u) = \text{Lerp}(u - i, p_i, p_{i+1}), \quad i \leq u \leq i + 1 \]
  \[ = (1 - u + i)p_i + (u - i)p_{i+1}, \quad i = \lfloor u \rfloor \]

- $N+1$ points define $N$ linear segments
- $x(i) = p_i$
- $C^0$ continuous by construction
- $C^1$ at $p_i$ when $p_i - p_{i-1} = p_{i+1} - p_i$
Piecewise Bézier curve

- Given $3N + 1$ points $p_0, p_1, \ldots, p_{3N}$
- Define $N$ Bézier segments:

\[
x_0(t) = B_0(t)p_0 + B_1(t)p_1 + B_2(t)p_2 + B_3(t)p_3
\]
\[
x_1(t) = B_0(t)p_3 + B_1(t)p_4 + B_2(t)p_5 + B_3(t)p_6
\]
\[\vdots\]
\[
x_{N-1}(t) = B_0(t)p_{3N-3} + B_1(t)p_{3N-2} + B_2(t)p_{3N-1} + B_3(t)p_{3N}
\]
Piecewise Bézier Curve

- Parameter in $0 \leq u \leq 3N$
  \[ x(u) = \begin{cases} 
  x_0 \left( \frac{1}{3} u \right), & 0 \leq u \leq 3 \\
  x_1 \left( \frac{1}{3} u - 1 \right), & 3 \leq u \leq 6 \\
  \vdots \\
  x_{N-1} \left( \frac{1}{3} u - (N - 1) \right), & 3N - 3 \leq u \leq 3N 
  \end{cases} \]

  \[ x(u) = x_i \left( \frac{1}{3} u - i \right), \text{ where } i = \left\lfloor \frac{1}{3} u \right\rfloor \]
Piecewise Bézier Curve

- $3N+1$ points define $N$ Bézier segments
- $x(3i)=p_{3i}$
- $C_0$ continuous by construction
- $C_1$ continuous at $p_{3i}$ when $p_{3i} - p_{3i-1} = p_{3i+1} - p_{3i}$
- $C_2$ is harder to achieve

$C_1$ discontinuous

$C_1$ continuous
Piecewise Bézier Curves

- Used often in 2D drawing programs

- Inconveniences
  - Must have 4 or 7 or 10 or 13 or … (1 plus a multiple of 3) control points
  - Some points interpolate, others approximate
  - Need to impose constraints on control points to obtain $C^1$ continuity
  - $C^2$ continuity more difficult

- Solutions
  - User interface using “Bézier handles”
  - Generalization to B-splines or NURBS
Bézier Handles

- Segment end points (interpolating) presented as curve control points
- Midpoints (approximating points) presented as “handles”
- Can have option to enforce $C_1$ continuity
Next Lecture

- Parametric surfaces