Foundations of Computer Graphics (Fall 2012)
CS 184, Lecture 9: Curves 1
http://inst.eecs.berkeley.edu/~cs184

Course Outline
- 3D Graphics Pipeline
  - Modeling
  - Animation
  - Rendering

Graphics Pipeline
- In HW 1, HW 2, draw, shade objects
- But how to define geometry of objects?
- How to define, edit shape of teapot?
- We discuss modeling with spline curves
  - Demo of HW 3 solution

Curves for Modeling
Rachel Shiner, Final Project Spring 2010

Motivation
- How do we model complex shapes?
  - In this course, only 2D curves, but can be used to create interesting 3D shapes by surface of revolution, lofting etc
- Techniques known as spline curves
- This unit is about mathematics required to draw these spline curves, as in HW 3
- History: From using computer modeling to define car bodies in auto-manufacturing. Pioneers are Pierre Bezier (Renault), de Casteljau (Citroen)

Outline of Unit
- Bezier curves
- deCasteljau algorithm, explicit form, matrix form
- Polar form labeling (next time)
- B-spline curves (next time)
- Not well covered in textbooks (especially as taught here). Main reference will be lecture notes. If you do want a printed ref, handouts from CAGD, Seidel
**Bezier Curve (with HW2 demo)**

- **Motivation**: Draw a smooth intuitive curve (or surface) given few key user-specified control points.
- Control points (all that user specifies, edits)

**Demo HW 3**

- Control polygon
- Smooth Bezier curve (drawn automatically)

---

**Issues for Bezier Curves**

Main question: Given control points and constraints (interpolation, tangent), how to construct curve?

- **Algorithmic**: deCasteljau algorithm
- Explicit: Bernstein-Bezier polynomial basis
- 4x4 matrix for cubics
- Properties: Advantages and Disadvantages

---

**deCasteljau: Linear Bezier Curve**

- Just a simple linear combination or interpolation (easy to code up, very numerically stable)

**Linear (Degree 1, Order 2)**

\[
\begin{align*}
F(0) &= P_0, \\
F(1) &= P_2 \\
F(u) &= (1-u) P_0 + u P_1
\end{align*}
\]

---

**deCasteljau: Quadratic Bezier Curve**

- Quadratic
- Degree 2, Order 3

\[
F(0) = P_0, \ F(1) = P_2, \ F(u) = ?
\]

\[
F(u) = (1-u)^2 P_0 + 2u(1-u) P_1 + u^2 P_2
\]

---

**Geometric interpretation: Quadratic**

- 1-u
- u
- 1-u

---
Geometric Interpretation: Cubic

**Summary: deCasteljau Algorithm**

- **Linear**
  - Degree 1, Order 2
  - \( P(0) = P_0, P(1) = P_1 \)

- **Quadratic**
  - Degree 2, Order 3
  - \( P(0) = P_0, P(1) = P_1, P(1) = P_2 \)

- **Cubic**
  - Degree 3, Order 4
  - \( P(0) = P_0, P(1) = P_1, P(2) = P_2, P(3) = P_3 \)

\[
F(u) = (1-u)^3 P_0 + 3u(1-u)^2 P_1 + 3u^2(1-u) P_2 + u^3 P_3
\]

**DeCasteljau Implementation**

Input: Control points \( C_i \) with \( 0 \leq i \leq n \) where \( n \) is the degree.
Output: \( f(u) \) where \( u \) is the parameter for evaluation.

1. for (level = n; level > 0; level --) {
   2. if (level == n) { // Initial control points
   3. for (i = 0; i < level; i++) {
   4. \( p^{level + i} = C_i \); continue ;
   5. \( p^{level + i} = (1-u) p^{level + i-1} + u * p^{level + i} \);
   6. }
   7. \( f(u) = p^0 \)
}

- Can be optimized to do without auxiliary storage

**Summary of HW2 Implementation**

- **Bezier (Bezier2 and Spline discussed next time)**
  - Arbitrary degree curve (number of control points)
  - Break curve into detail segments: Line segments for these
  - Evaluate curve at locations 0, 1/detail, 2/detail, ..., 1
  - Evaluation done using deCasteljau

- Key implementation: deCasteljau for arbitrary degree
  - Is anyone confused? About handling arbitrary degree?

- Can also use alternative formula if you want
  - Explicit Bernstein-Bezier polynomial form (next)

- Questions?

**Issues for Bezier Curves**

Main question: Given control points and constraints (interpolation, tangent), how to construct curve?

- Algorithmic: deCasteljau algorithm
- **Explicit**: Bernstein-Bezier polynomial basis
- 4x4 matrix for cubics
- Properties: Advantages and Disadvantages
Recap formulae

- Linear combination of basis functions
  Linear: \( F(u) = P_1(1-u) + P_2 u \)
  Quadratic: \( F(u) = P_1(1-u)^2 + P_2 u(1-u) + P_3 u^2 \)
  Cubic: \( F(u) = P_1(1-u)^3 + P_2 u(1-u)^2 + P_3 [3u^2(1-u)] + P_4 u^3 \)

Degree \( n \): \( F(u) = \sum_k P_k B^n_k(u) \)
\( B^n_k(u) \) are Bernstein-Bezier polynomials

- Explicit form for basis functions? Guess it?

Summary of Explicit Form

Linear: \( F(u) = P_1(1-u) + P_2 u \)
Quadratic: \( F(u) = P_1(1-u)^2 + P_2 u(1-u) + P_3 u^2 \)
Cubic: \( F(u) = P_1(1-u)^3 + P_2 [3u^2(1-u)] + P_3 u^3 \)

Degree \( n \): \( F(u) = \sum_k P_k B^n_k(u) \)
\( B^n_k(u) \) are Bernstein-Bézier polynomials

\( B^n_k(u) = \frac{n!}{k!(n-k)!} (1-u)^{n-k} u^k \)

Issues for Bezier Curves

Main question: Given control points and constraints (interpolation, tangent), how to construct curve?

- Algorithmic: deCasteljau algorithm
- Explicit: Bernstein-Bezier polynomial basis
- 4x4 matrix for cubics
- Properties: Advantages and Disadvantages

Cubic 4x4 Matrix (derive)

\[
F(u) = P_1(1-u)^3 + P_2 [3u^2(1-u)] + P_3 [3u^2(1-u)] + P_4 u^3
\]

\[
= \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}
\]

\[
M = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}
\]

\[
= \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}
\]

Recap formulae

- Linear combination of basis functions
  Linear: \( F(u) = P_1(1-u) + P_2 u \)
  Quadratic: \( F(u) = P_1(1-u)^2 + P_2 u(1-u) + P_3 u^2 \)
  Cubic: \( F(u) = P_1(1-u)^3 + P_2 [3u^2(1-u)] + P_3 u^3 \)

Degree \( n \): \( F(u) = \sum_k P_k B^n_k(u) \)
\( B^n_k(u) \) are Bernstein-Bézier polynomials

- Explicit form for basis functions? Guess it?

\( \text{Binomial coefficients in } [(1-u)+u]^n \)
Issues for Bezier Curves

Main question: Given control points and constraints (interpolation, tangent), how to construct curve?

- Algorithmic: deCasteljau algorithm
- Explicit: Bernstein-Bezier polynomial basis
- 4x4 matrix for cubics
- Properties: Advantages and Disadvantages

Properties (brief discussion)

- Demo of HW 3
- Interpolation: End-points, but approximates others
- Single piece, moving one point affects whole curve (no local control as in B-splines later)
- Invariant to translations, rotations, scales etc. That is, translating all control points translates entire curve
- Easily subdivided into parts for drawing (next lecture): Hence, Bezier curves easiest for drawing