Course Outline

- 3D Graphics Pipeline
  - Modeling → Animation → Rendering

Unit 1: Transformations
Weeks 1,2. Ass 1 due Feb 11

Unit 2: Spline Curves
Modeling geometric objects
Weeks 3,4
Ass 2 due Feb 25 (Demo)

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 2

- 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)
- Smooth Bezier curve (drawn automatically)
### Bezier Curve: (Desirable) properties
- Interpolates, is tangent to end points
- Curve within convex hull of control polygon
- Control points (all that user specifies, edits)

![Smooth Bezier curve (drawn automatically)](image)

### 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)

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

Linear \( \text{Degree 1, Order 2} \)
\[F(0) = P_0, \ F(1) = P_1, \ F(u) = ?\]

### deCasteljau: Quadratic Bezier Curve
- Geometric interpretation: Quadratic

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

Quadratic \( \text{Degree 2, Order 3} \)
\[F(0) = P_0, \ F(1) = P_2, \ F(u) = ?\]

### Geometric interpretation: Cubic

- Geometric Interpretation: Cubic
**deCasteljau: Cubic Bezier Curve**

- **Cubic**
- **Degree 3, Order 4**
- **F(0) = P0, F(1) = P3**

\[ F(u) = (1-u)^3 P0 + 3u(1-u)^2 P1 + 3u^2(1-u) P2 + u^3 P3 \]

**Summary: deCasteljau Algorithm**

- **Linear**
  - **Degree 1, Order 2**
  - **F(0) = P0, F(1) = P1**

- **Quadratic**
  - **Degree 2, Order 3**
  - **F(0) = P0, F(1) = P2**

- **Cubic**
  - **Degree 3, Order 4**
  - **F(0) = P0, F(1) = P3**

\[ F(u) = (1-u)^3 P0 + 3u(1-u)^2 P1 + 3u^2(1-u) P2 + u^3 P3 \]

**DeCasteljau Implementation**

- **Input:** Control points \( P_0, \ldots, P_n \) with \( 0 \leq i \leq n \) where \( n \) is the degree.
- **Output:** \( F(u) \) where \( u \) is the parameter for evaluation.

1. for \( (\text{level} \neq n ; \text{level} \geq 0 ; \text{level}--) \) {
   2. if \( (\text{level} == n) \) {
      // initial control points
   3. \( \ldots, 0 \leq i \leq n, P^\text{level} = P_i ; \text{continue} ; \)
   4. for \( (i = 0; i \leq \text{level} ; i++) \) {
      \( F^\text{level} = (1-u) \times F^\text{level} + u \times \text{next} ; \)
   5. }\text{next} = \{ P^\text{level} + 1 \} ;
   6. }
   7. \( F(u) = P^0 \)

- Can be optimized to do without auxiliary storage

**Summary of HW2 Implementation**

- Bezier (Bezier2 and Bspline 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: \( F(u) = P_i(1-u) + Pu \)
  - Quadratic: \( F(u) = P_i(1-u)^2 + P(2u(1-u)) + Pu^2 \)
  - Cubic: \( F(u) = P_i(1-u)^3 + P(3u(1-u)^2) + P(3u^2(1-u)) + Pu^3 \)

- Degree \( n \): \( F(u) = \sum P_k B_k^r(u) \)
  - \( B_k^r(u) \) are Bernstein-Bezier polynomials

- Explicit form for basis functions? Guess it?
Recap formulae

- Linear combination of basis functions
  - Linear: \( F(u) = P_0 + P_1 u \)
  - Quadratic: \( F(u) = P_0 + P_1 u + P_2 u^2 \)
  - Cubic: \( F(u) = P_0 + P_1 u + P_2 u^2 + P_3 u^3 \)

- Explicit form for basis functions? Guess it?

- Binomial coefficients in \([(1-u) + u]^n\)

Summary of Explicit Form

- Linear: \( F(u) = P_0 + P_1 u \)
- Quadratic: \( F(u) = P_0 + P_1 u + P_2 u^2 \)
- Cubic: \( F(u) = P_0 + P_1 u + P_2 u^2 + P_3 u^3 \)

Degree: \( F(u) = \sum P_k B^n_k(u) \)

\( B^n_k(u) \) are Bernstein-Bezier polynomials

- Explicit form for basis functions? Guess it?

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)

\[
F(u) = P_0(1-u)^3 + P_1[3u(1-u)^2] + P_2[3u^2(1-u)] + P_3u^3
= (u^3 \quad u^2 \quad u \quad 1) \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}
\]

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:  hw2.exe
- 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