Course Outline

- 3D Graphics Pipeline
  - Modeling (Creating 3D Geometry)
  - Rendering (Creating, shading images from geometry, lighting, materials)

Unit 1: Transformations
- Weeks 1, 2
- Assignment 1 due Sep 25

Unit 2: Spline Curves
- Modeling geometric objects
- Weeks 3, 4
- Assignment 2 due Oct 7 (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 a few key user-specified control points

  Control points (all that user specifies, edits)

  Smooth Bezier curve (drawn automatically)

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

**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)

**Geometric interpretation: Quadratic**

**deCasteljau: Quadratic Bezier Curve**
- Quadratic (Degree 2, Order 3)
  - F(0) = P0, F(1) = P2
  - F(u) = (1-u)^2 P0 + 2u(1-u) P1 + u^2 P2

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

Cubic Degree 3, Order 4
\[ F(0) = P_0, F(1) = P_3 \]

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

---

**Summary: deCasteljau Algorithm**

Linear
Degree 1, Order 2
\[ F(0) = P_0, F(1) = P_1 \]

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

Quadratic
Degree 2, Order 3
\[ F(0) = P_0, F(1) = P_2 \]

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

Cubic
Degree 3, Order 4
\[ F(0) = P_0, F(1) = P_3 \]

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

---

**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

```plaintext
1 for (level == n; level > 0; level--) {
  2 if (level == n) \{ // Initial control points
  3 \forall 0 \leq i \leq n, p_{i,0} = C_i \} continue ;
  4 for (i = 0; i < level; i++) \{
  5 \quad p_{k,i} = (1-u) * p_{k-1,i} + u * p_{k+1,i} ;
  6 \}
  7 F(u) = p_{n,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, \ldots, 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: \( F(u) = P_0(1-u) + P_n u \)
  - Quadratic: \( F(u) = P_0(1-u)^2 + P_1[2u(1-u)] + P_2 u^2 \)
  - Cubic: \( F(u) = P_0(1-u)^3 + P_1[3u(1-u)^2] + P_2[3u^2(1-u)] + P_3 u^3 \)

Degree \( n \): \( F(u) = \sum_{k=0}^{n} B_k^n(u) \) \( B_k^n (u) \) are Bernstein-Bezier polynomials
- Explicit form for basis functions? Guess it?
Recap formulae

- Linear combination of basis functions
  
  Linear: \[ F(u) = P_0 (1-u)^3 + P_0 u \]
  
  Quadratic: \[ F(u) = P_0 (1-u)^2 + P_1 [2u(1-u)] + P_2 u^2 \]
  
  Cubic: \[ F(u) = P_0 (1-u)^3 + P_1 [3u(1-u)^2] + P_2 [3u^2(1-u)] + P_3 u^3 \]

  Degree n: \( F(u) = \sum_k p_k B^k(u) \quad B^k(u) \) are Bernstein-Bezier polynomials

  - Explicit form for basis functions? Guess it?
  - Binomial coefficients in \((1-u)+u\)^n

Summary of Explicit Form

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

  Degree n: \( F(u) = \sum_k p_k B^k(u) \quad B^k(u) \) are Bernstein-Bezier polynomials

  \[ B^k_n(u) = \frac{n!}{k!(n-k)!} (1-u)^{n-k} u^k \]

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)

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

 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: hx2.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