

## Computer Graphics (Fall 2004)

COMS 4160, Lecture 6: Curves 1

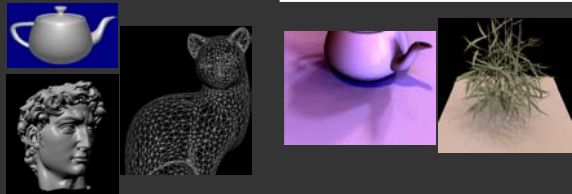
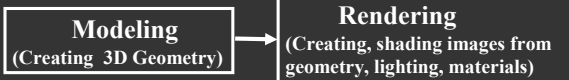
<http://www.cs.columbia.edu/~cs4160>

## To Do

- Download and compile skeleton for assignment 2  
By now, shouldn't be a problem getting set up
- Start thinking about partners for HW 3 and HW 4
  - Remember though, that HW2 is done individually
  - Your submission of HW 2 must include partner for HW 3

## Course Outline

- 3D Graphics Pipeline



## Course Outline

- 3D Graphics Pipeline



Unit 1: Transformations  
Weeks 1,2. Ass 1 due Sep 23

Unit 2: Spline Curves  
Modeling geometric objects  
Weeks 3,4. [hw2.exe](#)  
Ass 2 due Oct 7 (demo)

## Maya Demo (Aner, after HW 2 demo)

## Motivation

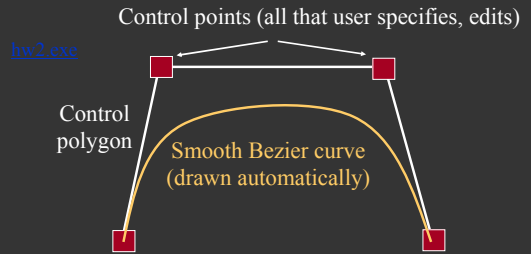
- How do we model complex shapes?
  - In this course, only 2D curves, but can be used to create interesting shapes as just seen in the demo
- Techniques known as spline curves
- This unit is about mathematics required to draw these spline curves, as in HW 2
- History: From using computer modeling to define car bodies in auto manufacturing. Pioneers are Pierre Bezier (Renault), de Casteljaou (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 want a printed ref, handouts from CAGD, Seidel

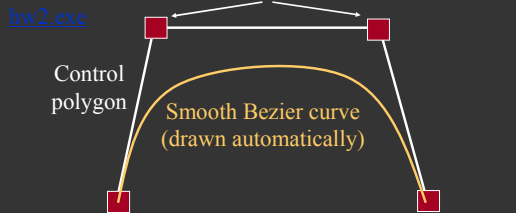
## Bezier Curve (with HW2 demo)

- Motivation: Draw a smooth intuitive curve (or surface) given a few key user-specified control points



## Bezier Curve: (Desirable) properties

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



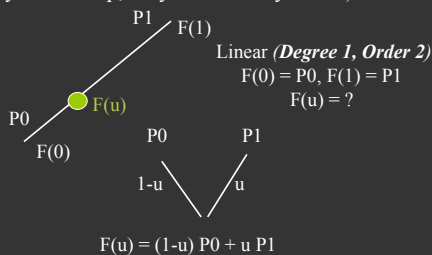
## Issues for Bezier Curves

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

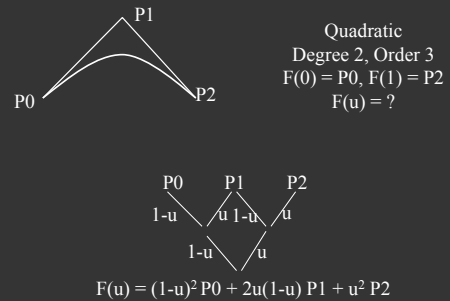
- Algorithmic: deCasteljau algorithm
- Explicit: Bernstein Bezier polynomial basis
- 4x4 matrix for cubics
- Properties: Advantages and Disadvantages

## deCasteljau: Linear Bezier Curve

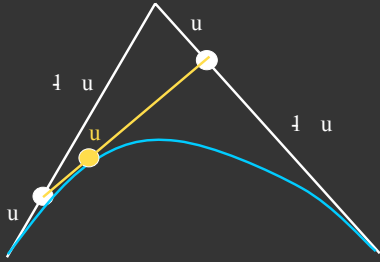
- Just a simple linear combination or interpolation (easy to code up, very numerically stable)



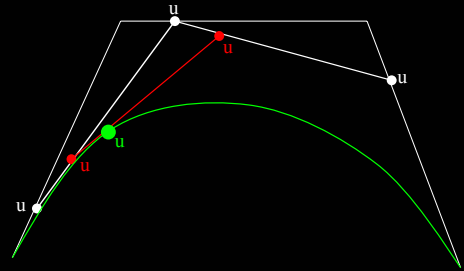
## deCasteljau: Quadratic Bezier Curve



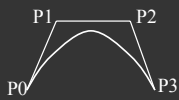
## Geometric interpretation: Quadratic



## Geometric Interpretation: Cubic



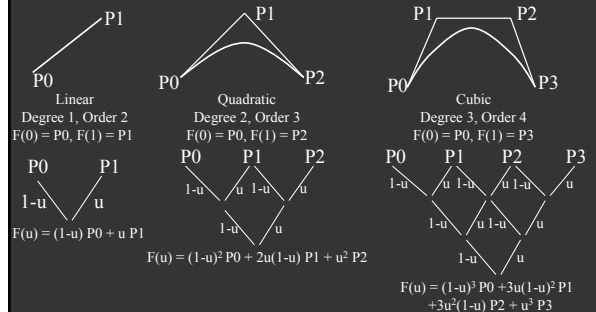
## deCasteljau: Cubic Bezier Curve



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



## 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  $0, 1/\text{detail}, 2/\text{detail}, \dots, 1$
  - 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:  $F(u) = P_0(1-u) + P_1u$

Quadratic:  $F(u) = P_0(1-u)^2 + P_1[2u(1-u)] + P_2u^2$

Cubic:  $F(u) = P_0(1-u)^3 + P_1[3u(1-u)^2] + P_2[3u^2(1-u)] + P_3u^3$

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

- Explicit form for basis functions? Guess it?

## Recap formulae

- Linear combination of basis functions

Linear:  $F(u) = P_0(1-u) + P_1u$

Quadratic:  $F(u) = P_0(1-u)^2 + P_1[2u(1-u)] + P_2u^2$

Cubic:  $F(u) = P_0(1-u)^3 + P_1[3u(1-u)^2] + P_2[3u^2(1-u)] + P_3u^3$

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

- Explicit form for basis functions? Guess it?
- Binomial coefficients in  $[(1-u)+u]^n$**

## Summary of Explicit Form

Linear:  $F(u) = P_0(1-u) + P_1u$

Quadratic:  $F(u) = P_0(1-u)^2 + P_1[2u(1-u)] + P_2u^2$

Cubic:  $F(u) = P_0(1-u)^3 + P_1[3u(1-u)^2] + P_2[3u^2(1-u)] + P_3u^3$

Degree n:  $F(u) = \sum_k P_k B_k^n(u)$   $B_k^n(u)$  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)

$$F(u) = P_0(1-u)^3 + P_1[3u(1-u)^2] + P_2[3u^2(1-u)] + P_3u^3$$

$$= \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \begin{matrix} M = ? \\ \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix} \end{matrix}$$

## Cubic 4x4 Matrix (derive)

$$F(u) = P_0(1-u)^3 + P_1[3u(1-u)^2] + P_2[3u^2(1-u)] + P_3u^3$$

$$= \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}$$

---

## 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