Computer Graphics

CSE 167 [Win 17], Lecture 9: Curves 1

Ravi Ramamoorthi

http://viscomp.ucsd.edu/classes/cse167/wi17

Course Outline

- 3D Graphics Pipeline

| *Modeling* | → | Animation | → | Rendering |
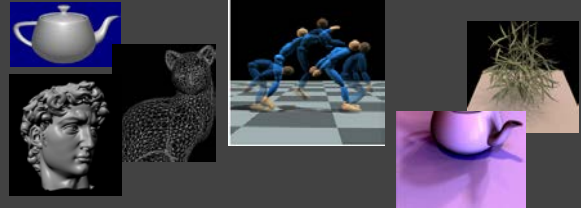
Graphics Pipeline

- 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

- Homework submission (Feb 29)
  - After midterm, but please start on it before
  - Not on edX edge, http://52.42.39.203/cs167hw3/
  - Same password as for readings (and code grade only)

Curves for Modeling
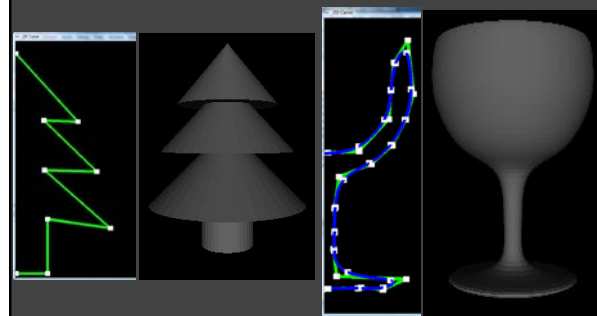
Rachel Shiner, Final Project Spring 2010
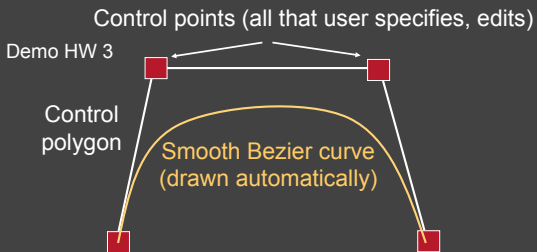

Motivation

- 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)

Bezier Curve: (Desirable) properties

- Interpolates, is tangent to end points
- Curve within convex hull of control polygon

Control points (all that user specifies, edits)
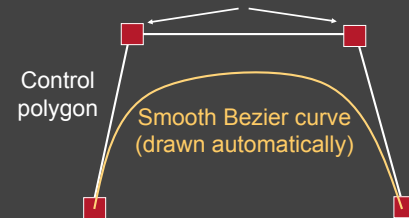
Control polygon

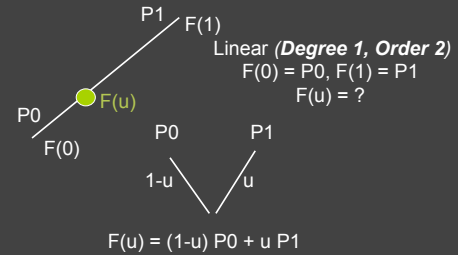Smooth Bezier curve (drawn automatically)

Issues for Bezier Curves

Main question: Given control points and constraints (interpolation, tangent), how to construct curve?
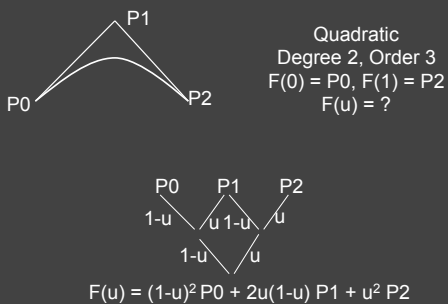
- *Algorithmic: deCasteljau algorithm*
- Explicit: Bernstein-Bezier polynomial basis
- 4x4 matrix for cubics
- Properties: Advantages and Disadvantages

---

deCasteljau: Linear Bezier Curve

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

P1 F(1)

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

P0   F(u)

F(0)

P0   P1

1-u   u

$$F(u) = (1-u)\, P0 + u\, P1$$

---

deCasteljau: Quadratic Bezier Curve

P1

P0   P2

Quadratic
Degree 2, Order 3
F(0) = P0, F(1) = P2
F(u) = ?

P0   P1   P2

1-u   u 1-u   u

1-u   u

$$F(u) = (1-u)^2\, P0 + 2u(1-u)\, P1 + u^2\, P2$$

Geometric interpretation: Quadratic

u

1-u

1-u

u

u

Geometric Interpretation: Cubic
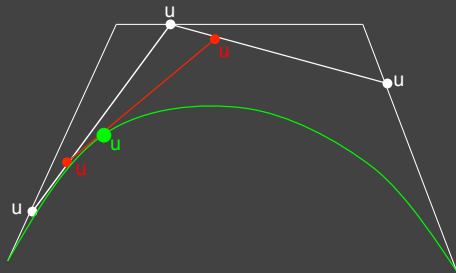
deCasteljau: Cubic Bezier Curve

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
$F(0) = P0$, $F(1) = P1$

$$F(u) = (1-u) P0 + u P1$$

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

$$F(u) = (1-u)^2 P0 + 2u(1-u) P1 + u^2 P2$$

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

$$F(u) = (1-u)^3 P0 + 3u(1-u)^2 P1 + 3u^2(1-u) P2 + u^3 P3$$
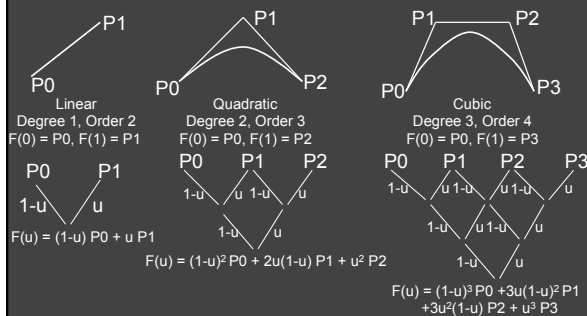
DeCasteljau Implementation

Input: Control points C_i with $0 \le i \le n$ 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        ∀i : 0 ≤ i ≤ n : p_i^{level} = C_i ; continue ; }
4     for (i = 0 ; i ≤ level ; i + +)
5        p_i^{level} = (1 − u) ∗ p_i^{level+1} + u ∗ p_{i+1}^{level+1} ;
6  }
7  f(u) = p_0^0
```

- 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

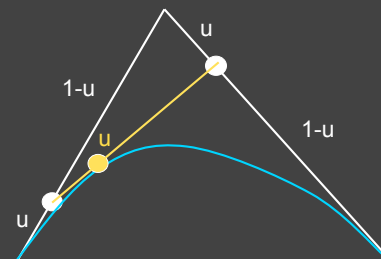- Properties: Advantages and Disadvantages

Recap formulae

- Linear combination of basis functions

Linear: $F(u) = P_0(1-u) + P_1 u$

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

$B_k^n(u)$ are Bernstein-Bezier polynomials

- Explicit form for basis functions? Guess it?

Recap formulae

- Linear combination of basis functions

Linear: $F(u) = P_0(1-u) + P_1 u$

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

$B_k^n(u)$ are Bernstein-Bezier polynomials

- Explicit form for basis functions? Guess it?

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

Summary of Explicit Form

Linear: $F(u) = P_0(1-u) + P_1 u$

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

$B_k^n(u)$ 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)

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

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

Cubic 4x4 Matrix (derive)

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

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

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 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