Computer Graphics

CSE 167 [Win 24], Lecture 9: Curves 1

Ravi Ramamoorthi

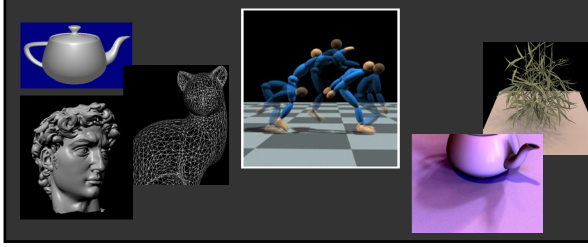
http://viscomp.ucsd.edu/classes/cse167/wi24

1

Course Outline

- 3D Graphics Pipeline

Modeling → Animation → Rendering



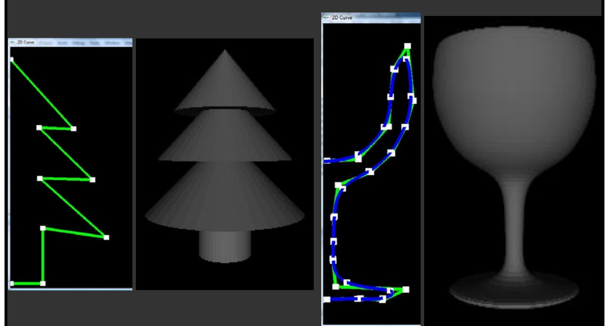
2

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 28)
  - After midterm, but please start on it before
  - Not on UCSD Online, link
  - See piazza for password (and code grade only)

3

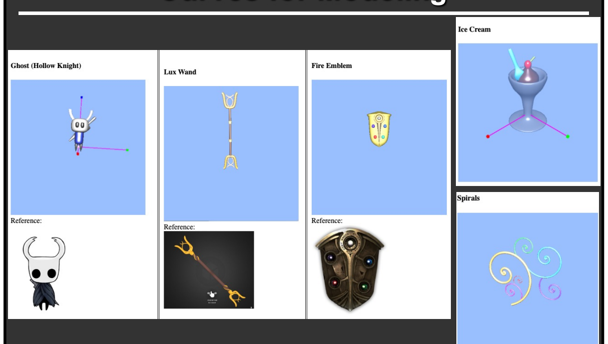
Curves for Modeling



Rachel Shiner, Final Project Spring 2010

4

Curves for Modeling



James Zhao, HW 3 Extra Credit Winter 2023

5

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)

6

Splines Video

- https://www.youtube.com/watch?v=YMl25iCCRew&list=PLWfDJ5nla8UpwShx-lzLJqcp575fKpsSO&index=13
- Steve Seitz UW 5 minute videos (only first 2.5min)
- Can watch other splines videos on channel, but don't match the math as taught in this class.

7

Outline of Unit

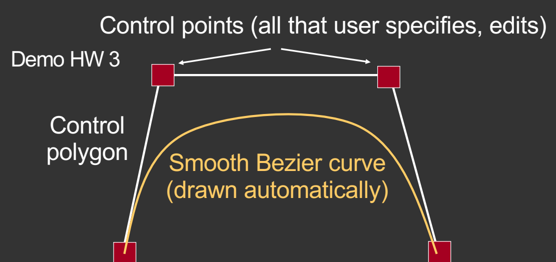
- *Bezier curves*
- deCasteljau algorithm, explicit form, matrix form
- Polar form labeling (next time)
- B-spline curves (next time)

- 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

8

Bezier Curve (with HW3 demo)

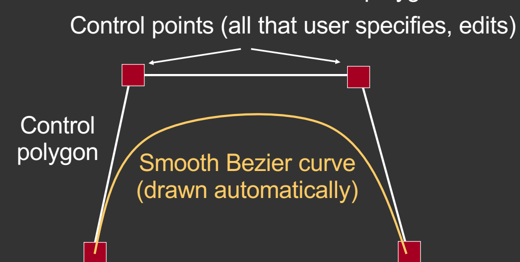
- *Motivation: Draw a smooth intuitive curve (or surface) given few key user-specified control points*



9

Bezier Curve: (Desirable) properties

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



10

Survey

- Anonymous, know how things are going
- Will try to use it to improve course

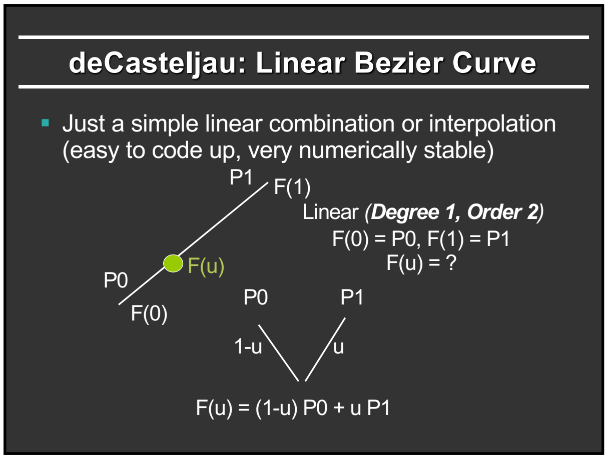
11

Issues for Bezier Curves

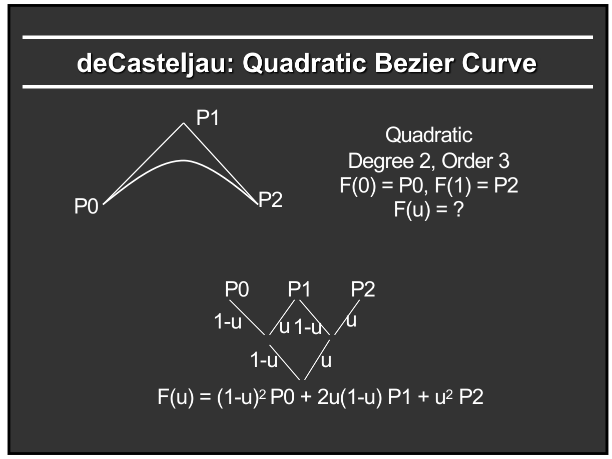
Main question: Given control points and constraints (interpolation, tangent), how to construct curve?

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

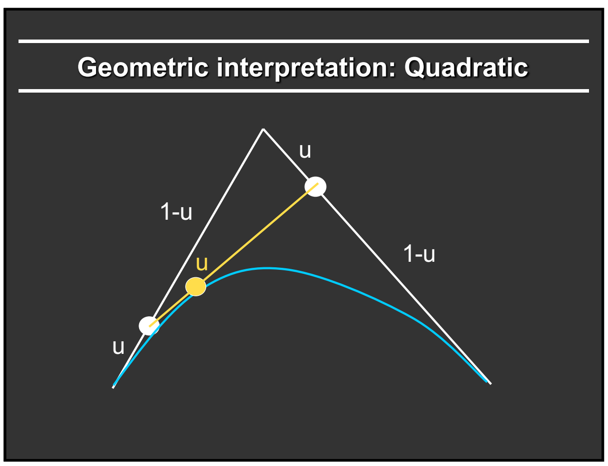
12



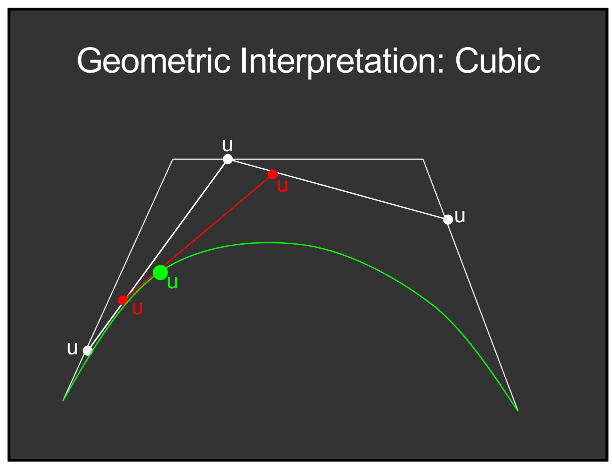
13



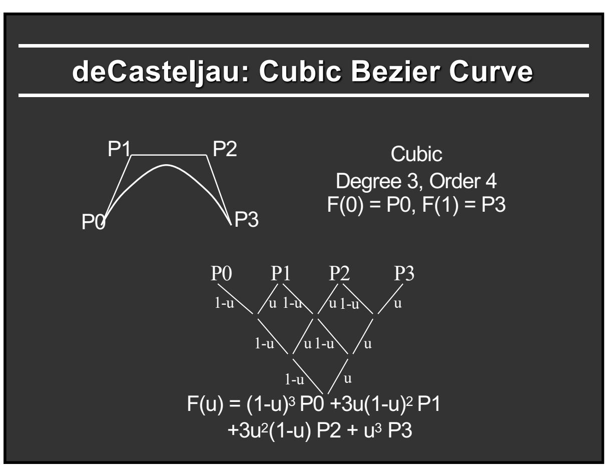
14



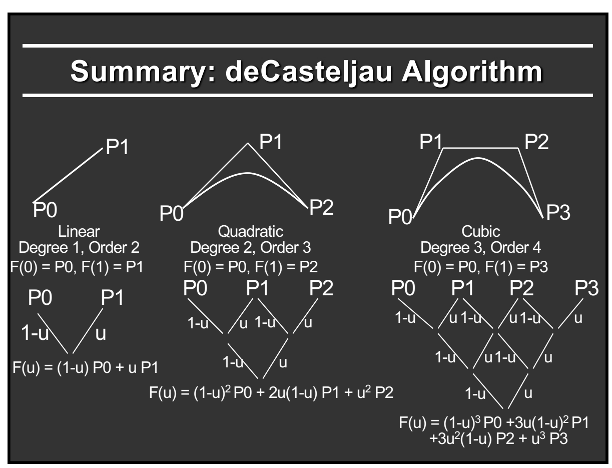
15



16



17



18

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

```
for (level = n ; level ≥ 0 ; level − −) {
  if (level == n) { // Initial control points
    ∀i : 0 ≤ i ≤ n : p_i^level = C_i ; continue ; }
  for (i = 0 ; i ≤ level ; i + +)
    p_i^level = (1 − u) * p_i^(level+1) + u * p_(i+1)^(level+1) ;
}
f(u) = p_0^0
```

- Can be optimized to do without auxiliary storage

19

Summary of HW3 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?

20

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

21

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?

22

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

23

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

$$B_k^n(u) = \frac{n!}{k!(n-k)!}(1-u)^{n-k} u^k$$

24

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

25

Cubic 4x4 Matrix (derive)

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

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

26

Cubic 4x4 Matrix (derive)

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

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

27

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*

28

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

29