CS 430
Computer Graphics

Curve Drawing Algorithms
Week 4, Lecture 8
David Breen, William Regli and Maxim Peysakhov
Department of Computer Science
Drexel University

Outline

• Drawing of 2D Curves
  – De Casteljau algorithm
  – Subdivision algorithm
  – Drawing parametric curves

The de Casteljau Algorithm

• How to compute a sequence of points that approximates a smooth curve given a set of control points?
• Developed by Paul de Casteljau at Citroën in the late 1950s
• Idea: recursively subdivide the curve and add points to refine the number of control points

Recall: Linear Interpolation

• Simple example
  – interpolating along the line between two points
  – (really an affine combination of points $a$ and $b$)
  – $x(t) = a + (b-a)t$

Properties of Piecewise Linear Interpolations

• Given
  – continuous curve, $C$
  – piecewise linear interpolant (PLI) of $C$
  – and an arbitrary plane, $P$
• Then:
The number of crossings of $P$ by PLI is no greater than those of $C$

Linear Interpolation: Example 1

• Constructing a parabola using three control points
• From analytic geometry
  \[
  \text{ratio}(u, v, w) = (v - u)/(w - u) \\
  \text{ratio}(b_0, b_1, b_2) = \text{ratio}(b_1', b_2', b_1') = \text{ratio}(b_2', b_1', b_2') = t 
  \]
The de Casteljau Algorithm

Basic case, with two points:
- Plotting a curve via repeated linear interpolation
  - Given \( P_0, P_1, \ldots \) a sequence of control points
  - Simple case: Mapping a parameter \( u \) to the line \( P_0 P_1 \)
  \[ p(u) = (1-u)P_0 + uP_1 \text{ for } 0 \leq u \leq 1. \]

The de Casteljau Algorithm

Generalizing to three points
- Interpolate along the line \( P_0P_1 \)
  \( p_0(u) = (1-u)P_0 + uP_1 \)
- Interpolate along the resulting points
  \( p_{10}(u) = (1-u)p_0 + up_1 \)
  \( p_{11}(u) = (1-u)p_2 + up_2 \)

The de Casteljau Algorithm

The complete solution from the algorithm for three iterations:
\[ p_{10}(u) = (1-u)p_0 + up_1 \]
\[ p_{11}(u) = (1-u)p_2 + up_2 \]
\[ p(u) = (1-u)p_{10}(u) + up_{11}(u) \]

The de Casteljau Algorithm

The solution after four iterations:

The de Casteljau Algorithm

Example Results

- Input: \( p_0, p_1, \ldots, p_n \in \mathbb{R}^3, t \in \mathbb{R} \)
- Iteratively set:
  \[ p_r(t) = \begin{cases} p_{r-1}(t) + t p_{r+1}(t) & \text{for } r = 1, \ldots, n \end{cases} \]
  \[ p_{10}(t) = p_i \text{ for } i = 0, \ldots, n-r \]
- Then \( p_r(t) \) is the point with parameter value \( t \) on the Bézier curve defined by the \( p_i \)'s
The de Casteljau Algorithm: Example Results

- A degree 6 curve
- 60 points computed on the curve
  - the black points
- Intermediate control points
  - the gray points

De Casteljau: Arc Segment Animation

De Casteljau: Cubic Curve Animation

De Casteljau: Loop Curve Animation

The de Casteljau Algorithm: Some Observations

- Interpolation along the curve is based only on $u$
- Drawing the curve’s pixels requires iterating over $u$ at sufficient refinement
- What is the right increment?
- It’s not constant!

Subdivision

- Common in many areas of graphics, CAD, CAGD, vision
- Basic idea
  - primitives def’d by control polygons
  - set of control points is not unique
  - more than one way to compute a curve
  - subdivision refines representation of an object by introducing more control points
- Allows for local modification
- Subdivide to pixel resolution
Bézier Curve Subdivision

- Subdivision allows display of curves at different/adaptive levels of resolution
- Rendering systems (OpenGL, ActiveX, etc) only display polygons or lines
- Subdivision generates the lines/facets that approximate the curve/surface
  – output of subdivision sent to renderer

Bézier Curve Subdivision, with de Casteljau

- Calculate the value of \( x(u) \) at \( u = 1/2 \)
- This creates a new control point for subdividing the curve
- Use the two new edges to form control polygon for two new Bézier curves

Drawing Parametric Curves

Two basic ways:
- **Iterative evaluation** of \( x(t), y(t), z(t) \) for incrementally spaced values of \( t \)
  – can’t easily control segment lengths and error
- **Recursive Subdivision** via de Casteljau, that stops when control points get sufficiently close to the curve
  – i.e. when the curve is nearly a straight line
- Use Bresenham to draw each line segment

Drawing Parametric Curves via Recursive Subdivision

- Idea: stop subdivision when segment is flat enough to be drawn w/ straight line
- **Curve Flatness Test:**
  – based on the convex hull
  – if \( d_1 \) and \( d_2 \) are both less than some \( \varepsilon \)
    then the curve is declared flat

FYI: Computing the Distance from a Point to a Line

- Line is defined with two points
- Basic idea:
  – Project point \( P \) onto the line
  – Find the location of the projection
  \[
  d(P,L) = \frac{(y_2 - y_1)x + (x_2 - x_1)y + (x_1y_2 - x_2y_1)}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}
  \]
Drawing Parametric Curves via Recursive Subdivision

The Algorithm:

• DrawCurveRecSub(curve, e)
  – If straight(curve, e) then DrawLine(curve)
  – Else
    • SubdivideCurve(curve, LeftCurve, RightCurve)
    • DrawCurveRecSub(LeftCurve, e)
    • DrawCurveRecSub(RightCurve, e)
Beziers Curve Drawing

- Given control points, you can either:
  - Iterate through \( t \) and evaluate the formula.
  - Iterate through \( t \) and use the de Casteljau Algorithm.
    - Successive interpolation of control polygon edges.
    - Recursively subdivide the Casteljau polygons until they are approximately flat.
  - Generate more control points with degree elevation until the control polygon approximates the curve.

Programming Assignment 3

- Input PostScript-like file.
- Output B/W PBM.
- Implement viewports.
- Use HW2 for polygon clipping.
- Implement scanline polygon filling. (You cannot use flood filling algorithms)

Extra Credit Assignment

- Input Curveto PostScript-like file
  - Read in control points for a number of curves.
- Support HW3 options.
- Interpret points as the control points of a cubic Bezier curve.
- Evaluate curve to produce an approximating polyline.
- Draw lines into frame buffer.
- Output B/W PBM.