Outline

- Drawing of 2D Curves
  - De Casteljau algorithm
  - Subdivision algorithm
  - Drawing parametric curves
- Introduction and discussion of homework #3

The de Casteljau Algorithm

- How do define a smooth curve that approximates a sequence 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)
  - \( x(t) = a + (b-a)t \)

Historic Analogy: The Universe of Aristotle and Ptolemy

- Earth is in the center
- Planetary motion described as Epicycles
- For some planets Epicycles had to be placed on Epicycles
- In the similar manner we will have linear interpolation of the linear interpolating points

Properties of Piecewise Linear Interpolations

- Given
  - continuous curve, C
  - piecewise linear interpolant of C, PLC
  - and an arbitrary plane, P
- Then:
  The number of crossings of P by PLC is no greater than those of C
Linear Interpolation: Example 1

- Constructing a parabola using three control points
- From analytic geometry

\[
\text{ratio}(b_0, b_1, b_2) = \text{ratio}(b_1, b_2, b_3) = \text{ratio}(b_3, b_0, b_1) = t
\]

Linear Interpolation: Example 2

- Constructing a Bézier curve with four control points
- Point \( b_i(t) \) is obtained by repeated linear interpolation
- Shown: cubic case, \( n=3 \) and \( r=1/3 \)

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 \( p_0, p_1 \)

\[
p(u) = (1 - u)p_0 + up_1 \quad \text{for} \quad 0 \leq u \leq 1.
\]

The de Casteljau Algorithm

- Generalizing to three points
  - Interpolate \( p_0p_1 \) and \( p_1p_2 \)
  - Interpolate along the resulting points

\[
p_{01}(u) = (1 - u)p_0 + up_1
\]

\[
p_{11}(u) = (1 - u)p_1 + up_2
\]

\[
p(u) = (1 - u)p_{01}(u) + up_{11}(u)
\]

The de Casteljau Algorithm

- The complete solution from the algorithm for three iterations:

\[
p_0, p_1, p_01, p_11, p(u)
\]

Final Value

The de Casteljau Algorithm

- The solution after four iterations:
The de Casteljau Algorithm

- Input: $p_0, p_1, p_2, \ldots, p_n \in \mathbb{R}^3$, $t \in \mathbb{R}$
- Iteratively set:
  
  $p_{i,r}(t) = (1-t)p_{i(r-1)}(t) + t p_{i(r-1)}(t)$  
  
  $r = 1, \ldots, n$
  $i = 0, \ldots, n - r$

Then $p_{i,0}(t)$ 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: Wave 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

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

Bézier Curve Subdivision

- Subdivision allows display of curves at different 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, avec 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

Bézier Curve Subdivision

- Observe subdivision:
  - does not affect the shape of the curve
  - partitions one curve into several curved pieces with (collectively) the same shape
Subdivision: Arc Segment

Subdivision: Cubic Curve

Subdivision: Wave Curve

Subdivision: Loop Curve

Bézier Curve: Degree Elevation

- Given a control polygon
- Generate additional control points
- Keep the curve the same
- In the limit, this converges to the curve defined by the original control polygon

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 with straight line
- Curve Flatness Test:
  - based on the convex hull
  - if $d_2$ and $d_3$ 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

$$d(P, L) = \frac{(y_0 - y_1)x + (x_1 - x_0)y + (x_0y_1 - x_1y_0)}{\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}}$$

Programming assignment 3

- Output B/W XPM, input PostScript like file.
- Implement viewports.
- Use Weiler-Atherton intersection for polygon clipping.
- Implement scanline polygon filling. (You cannot use flood filling algorithms)

Outline

- Clipping of 2D Curves
  - Newtonian Method
  - Bezier Clipping
- Introduction and discussion of homework #3

Ray Curve Intersection

- Parametric Bezier curve
  $$p(t) = \sum B_i(t) p_i$$
- Use implicit ray (line) equation
  $$\langle x, y \rangle = ax + by + c$$
  - normalized: $a^2 + b^2 = 1$
- Solve for intersection: $$\langle p(t) \rangle = 0$$
  $$\langle p(t) \rangle = ax(t) + by(t) + c$$
  $$= a \sum B_i(t) x_i + b \sum B_i(t) y_i + c$$
  $$= \sum B_i(t) (ax_i + by_i + c)$$
- Solve: $$d(t) = \sum B_i(t) d_i = 0$$
- Distance to ray: $$d = ax + by + c$$
Explicitize the Curve

- Solve: $d(t) = \sum B_i(t) \ d_i = 0$
- Define new (explicit) Bezier curve
  
  $d(t) = \sum B_i(t) \ d_i$

  $\textbf{d}_0 = (0, \ d_0)$, $\textbf{d}_1 = (1/3, \ d_1)$,
  $\textbf{d}_2 = (2/3, \ d_2)$, $\textbf{d}_3 = (1, \ d_3)$

- Bezier curve $d(t)$ intersects $t$-axis at same parameter $t$ that Bezier curve $p(t)$ intersects ray!

---

Bezier Clipping

- Curve $d(t)$ bounded by convex hull of its control points $\textbf{d}_i$
- Find intersection of convex hull with $t$ axis: $t_0, t_1$
- If curve $p(t)$ intersects ray then intersection occurs at $t$ in $[t_0, t_1]$ $p_0$, $p_1$
- Apply de Casteljau twice to subdivide $p(t)$ into $p'(t)$ such that $p_0' = p_0(t_0)$ and $p_3' = p(t_1)$
- Do it all over again until the curve