## CS 460

## Computer Graphics

**Professor Richard Eckert**

**March 24, 2004**

## B-Spline Polynomials

## B-Spline Polynomials

- Want local control
- Smoother curves
- B-spline curves:
  - Segmented approximating curve
  - 4 control points affect each segment
    - Local control
  - Level-2 continuity everywhere
    - Very smooth

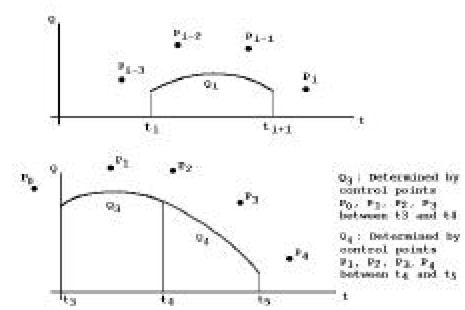
## Cubic B-Spline Polynomial Curves

- Approximate m+1 control points Pi (i=0,1,2,…,m)
  with a curve consisting of m-2 cubic polynomial curve <u>segments</u> $Q_i$ (i=3,4,...m), m>=3
- Each $Q_i$ defined in terms of:
  - parameter t: $t_i<=t<=t_{i+1}$
  - and by four of the m+1 control points

---

- Segment $Q_i$ determined by control points:
  $P_{i-3}$, $P_{i-2}$, $P_{i-1}$, $P_i$ between $t_i$ and $t_{i+1}$
- $Q_i$ begins at $t = t_i$ and ends at $t = t_{I+1,}$
- $Q_{i+1}$ joins $Q_i$ at $t_{i+1}$
- Join point called a knot.
- For example
  - First segment is $Q_3$, begins at $t_3$, ends at $t_4$
  - Is determined by control points P0,P1,P2,P3
- Each segment is affected by only 4 control points
- Each control pt affects at most 4 curve segments

## Uniform Cubic B-Spline Curves



## Uniform Cubic B-Splines

- A special case where we assume that:
- $t_{i+1} = t_i + 1$
- Polynomial equation for segment Qi:
- $Qi(t) = a*(t-ti)^3 + b*(t-ti)^2 + c*(t-ti) + d$, $ti <= t <= ti+1$
- Take independent variable as t-ti
  - Will vary from 0 to 1 for any interval

---

- Need to get polynomial coefficients (a,b,c,d)
  - from control points
- Find a "B-Spline Basis Matrix"
  - as for Bezier curves
  - but must do computation for each interval

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = M_{BS} * \begin{bmatrix} P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ P_i \end{bmatrix}$$

- $M_{BS}$ is the desired matrix

---

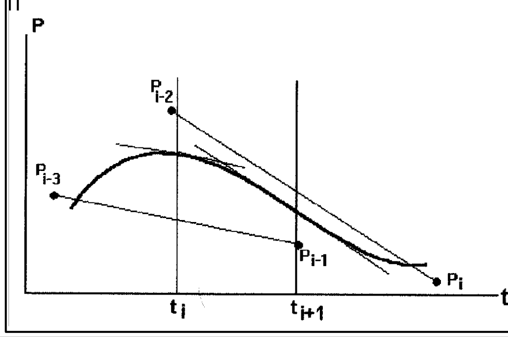
## B-Spline Continuity Conditions

- Conditions on 1st & 2nd derivatives:

1. $dQ_i/dt$ (at $t=t_i$) = slope of line segment joining $P_{i-3}$ and $P_{i-1}$
2. $dQ_i/dt$ (at $t=t_{i+1}$) = slope of line segment joining $P_{i-2}$ and $P_i$
3. $(dQ_i/dt)'$ $(t=t_i)$ = rate of change in slope at $t_i$ :
   ( slope of $[P_{i-3}-P_{i-2}]$ - slope of $[P_{i-2}-P_i]$ ) / ? t
4. $(dQi/dt)'$ $(t=t_{i+1})$ = rate of change in slope at $t_{i+1}$ :
   ( slope of $[P_{i-2}-P_{i-1}]$ - slope of $[P_{i-1}-P_i]$ ) / ? t

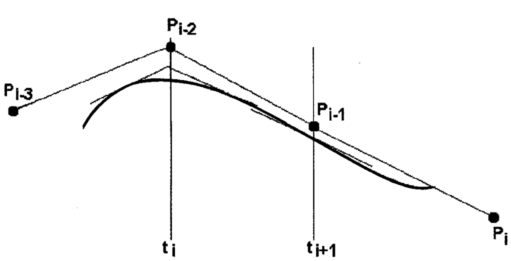
---

## Continuity Conditions 1 and 2




---

## Continuity Conditions 3 and 4




---

$Qi = a*(t-ti)^3 + b*(t-ti)^2 + c*(t-ti) + d$

$dQi/dt = 3*a*(t-ti)^2 + 2*b*(t-ti) + c$

$(dQi/dt)' = 6a*(t-ti) + 2*b$

- Condition 1: $c = ( P_{i-1} - P_{i-3})/2$
- Condition 2: $3*a + 2*b + c = (P_i - P_{i-2})/2$
- Condition 3: $2*b = ((P_{i-1} - P_{i-2}) - (P_{i-2} - P_{i-3})) / 1$
- Condition 4: $6*a + 2*b = ((P_i-P_{i-1})-(P_{i-1}-P_{i-2})) / 1$
- These four equations are not independent
  - Solving gives only a, b, c, but not d

## Solution

$a = (1/6) * (-P_{i-3} + 3*P_{i-2} - 3*P_{i-1} + P_i)$
$b = (1/6) * (3*P_{i-3} - 6*P_{i-2} + 3*P_{i-1})$
$c = (1/6) * (-3*P_{i-3} + 3*P_{i-1})$

## Need another condition to get d

- Choose the following condition:
  $Q\text{ (at t=ti)} = (1/6) * (P_{i-3} + 4*P_{i-2} + P_{i-1})$
  - i.e., control point at ti ($P_{i-2}$) pulls 4 times as hard at t=ti as control points on either side of ti
- Substitute in polynomial equation -->
  $d = (1/6) * (P_{i-3} + 4*P_{i-2} + P_{i-1})$

## Uniform Cubic B-Spline Coefficient Matrix Equation

$$\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = (1/6) * \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} * \begin{bmatrix} P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ P_i \end{bmatrix}$$

## Could also be written in terms of blending functions

$$Q_i(t) = \sum_{j=0}^{3} B_{i-j,4}(t) * P_{i-j}$$

$B_{i-3,4} = 1/6 * (1-t)^3$
$B_{i-2,4} = 1/6 * (3t^3 - 6t^2 + 4)$
$B_{i-1,4} = 1/6 * (-3t^3 + 3t^2 - 3t + 1)$
$B_{i,4} = 1/6 * t^3$

See Foley & Van Dam

## Plotting Uniform Cubic B-Splines

- Given m+1 control points P0,P1,P2,...Pm
  - (Recall that each has an x and y coordinate)
    - i.e., P0 --> x0 and y0, etc.
- The following is a "brute force" algorithm to plot the curve
  - delta is a very small increment (e.g., 0.05)

## Brute Force Algorithm

```
For (i=3 to m)
    Compute ax,bx,cx,dx and ay,by,cy,dy
        from control points i-3, i-2, i-1, i
    For (t=0; t<=1; t+=delta)
      x = ax*t^3 + bx*t^2 + cx*t + dx
      y = ay*t^3 + by*t^2 + cy*t + dy
      If (t==0)
        MoveTo(x,y)
      Else
        LineTo(x,y)
```

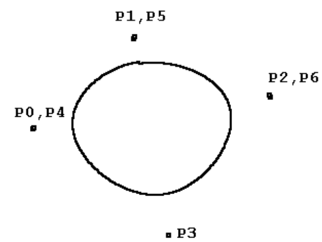
- To increase performance, use forward differences

## Closed Cubic B-Splines

Make last 3 control points coincide with 1st 3

0 <--> m-2, 1 <--> m-1, 2 <--> m

Example: m=6



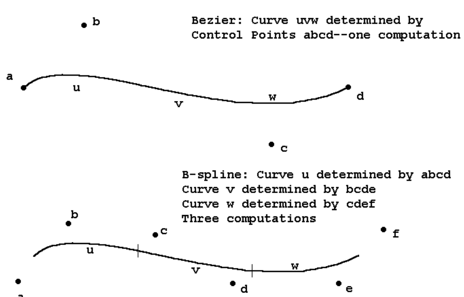

## Forcing Interpolation

Reproduce a control point three times

## Properties of Uniform B-Splines

1. Local Control
   – Each segment determined by only 4 control points
2. Approximates control points; doesn't interpolate
   (However it will interpolate triplicated control points)
3. Lies inside convex hull of control points
   – Each segment lies inside convex hull of its 4 control points
4. Invariant under affine transformations
5. Very smooth
   – Level-2 continuity everywhere
6. More computations required than for "equivalent" Bezier curve

## Bezier vs. B-Spline Curves




## Non-uniform Cubic B-Splines

- Greater variety of curve shapes
- Can have cusps and discontinuities
- Intervals between successive knots varies
- Knot values must be specified
  $t_0, t_1, t_2, t_3, t_4, \ldots, t_{m-2}$

NON-UNIFORM CUBIC B-SPLINES

Variable size intervals between successive knot values

Must specify knot values --> the knot vector,
a non-decreasing sequence
e.g., (0,0,0,0,1,1,2,3,4,4,....)

Can have multiple knots

The curve segment $Q_i$ is determined by control points: $P_{i-3}$, $P_{i-2}$, $P_{i-1}$, $P_i$

and by blending functions: $B_{i-3,4}(t)$, $B_{i-2,4}(t)$, $B_{i-1,4}(t)$, $B_{i,4}(t)$

[4 = the order (degree-3 plus 1) of the polynomials]

is given by:

$$Q_i(t) = P_{i-3} B_{i-3,4}(t) + P_{i-2} B_{i-2,4}(t) + P_{i-1} B_{i-1,4}(t) + P_i B_{i,4}(t)$$

$3 <= i <= m$, $t_i <= t < t_{i+1}$ defined between $t_3$ and $t_{m+1}$

If $t_i = t_{i+1}$ then the curve segment $Q_i$ degenerates to a point.

The Blending functions B(t) are defined recursively:

$$B_{i,1}(t) = \begin{cases} 1, & t_i <= t < t_{i+1} \\ 0, & \text{otherwise} \end{cases}$$

$$B_{i,2}(t) = \frac{t - t_i}{t_{i+1} - t_i} B_{i,1}(t) + \frac{t_{i+2} - t}{t_{i+2} - t_{i+1}} B_{i+1,1}(t)$$

$$B_{i,3}(t) = \frac{t - t_i}{t_{i+2} - t_i} B_{i,2}(t) + \frac{t_{i+3} - t}{t_{i+3} - t_{i+1}} B_{i+1,2}(t)$$

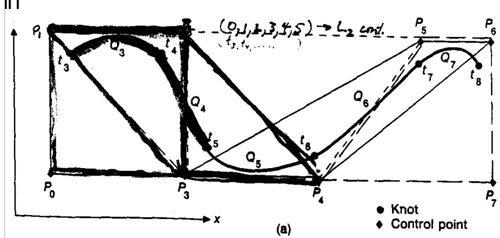
$$B_{i,4}(t) = \frac{t - t_i}{t_{i+3} - t_i} B_{i,3}(t) + \frac{t_{i+4} - t}{t_{i+4} - t_{i+1}} B_{i+1,3}(t)$$

In these equations, 0/0 is defined to be equal to 0.

## Case A (Level-2 Continuity)

- Knot vector: (0,1,2,3,4,5,...)
  - Just our friend the uniform B-spline
- Q3 determined by P0, P1, P2, P3
- Q4 determined by P1, P2, P3, P4
- Q3 and Q4 share control points P1,P2,P3
  - Three constraints ==> L0, L1, L2 continuity

## Case A (Level-2 Continuity)

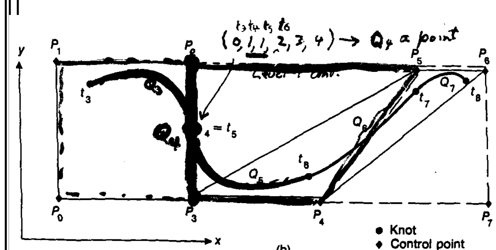


(a)

## Case B (Level-1 Continuity)

- Knot vector: (0,1,1,2,3,4,...)
- Segment Q4 becomes a point
  - (since t4 = t5)
- Q3 determined by P0, P1, P2, P3
- Q5 determined by P2, P3, P4, P5
- So Q4 must lie on line connecting P2 & P3
- Q3 and Q5 share control points P2 & P3
  - Two constraints ==> L0, L1 continuity

## Case B (Level-1 Continuity)

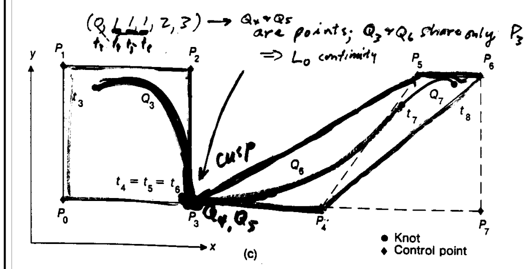


(b)

## Case C (Level-0 continuity)

- Knot vector: (0,1,1,1,2,3,...)
- Q4 and Q5 become points
  - (since t4=t5=t6)
- Q3 determined by P0, P1, P2, P3
- Q6 determined by P3, P4, P5, P6
- So Q4/Q5 must lie on control Point P3
  - (interpolates it)
- Q3 and Q6 share control point P3
  - One constraint ==> L0 continuity

## Case C (Level-0 continuity)




## Case D (No Continuity)

- Knot vector: (0,1,1,1,1,2,...)
- Q4, Q5, Q6 become points
  - (since t4=t5=t6=t7)
- Q3 determined by P0, P1, P2, P3
- Q7 determined by P4, P5, P6, P7
- There is no overlap
- Q3 and Q7 share no control points
  - No constraints ==> discontinuity

## Case D (No Continuity)