CS-C3100 Computer Graphics

Bézier Curves and Splines

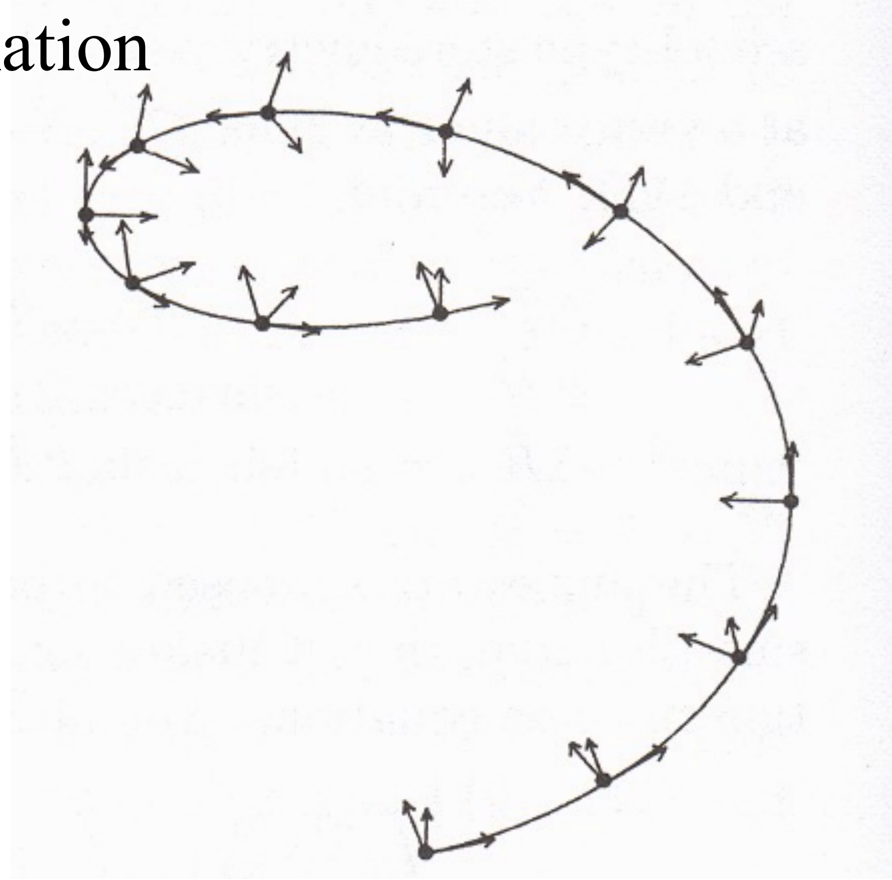
3.4 Differential properties of curves & cubic B-Splines

In These Slides

- Velocity, tangent, and curvature of smooth curves
- Orders of continuity
  - How smoothly curve segments join together
- Cubic B-Splines
  - C^2 (curvature continuous=very smooth) cubic splines

Differential properties of curves

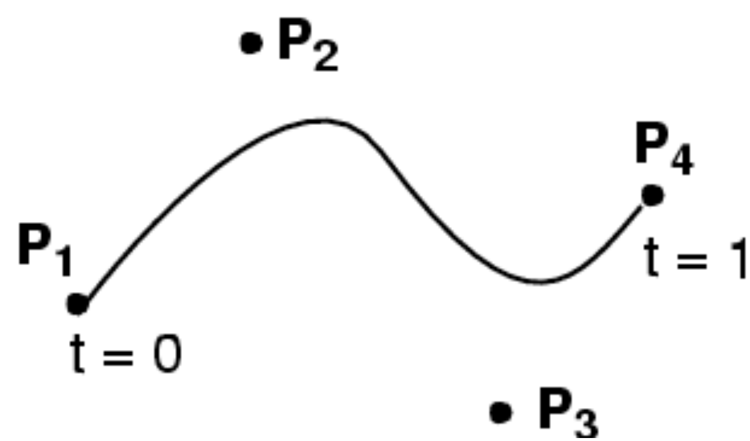
- Motivation
  - Compute normal for surfaces
  - Compute velocity for animation
  - Analyze smoothness



Velocity

- First derivative w.r.t. *t*
- Can you compute this for Bezier curves?

$$
\begin{aligned}
P(t) = \quad & (1-t)^3 && \mathbf{P}_1 \\
+ \quad & 3t(1-t)^2 && \mathbf{P}_2 \\
+ \quad & 3t^2(1-t) && \mathbf{P}_3 \\
+ \quad & t^3 && \mathbf{P}_4
\end{aligned}
$$

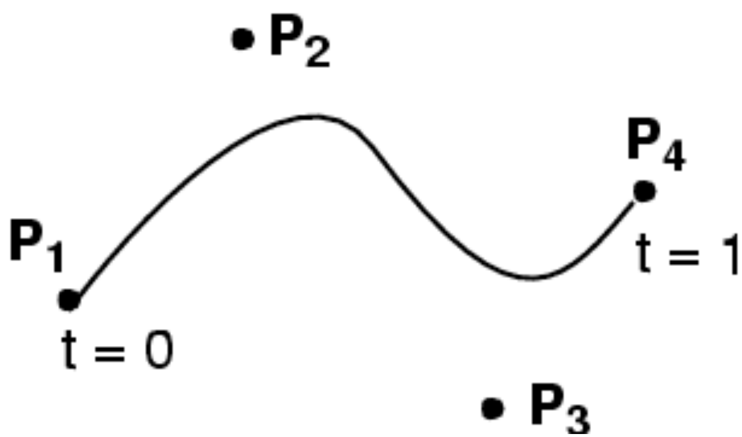


- You know how to differentiate polynomials...

Velocity

- First derivative w.r.t. *t*
- Can you compute this for Bezier curves?

$$P(t) = (1-t)^3 \mathbf{P}_1 + 3t(1-t)^2 \mathbf{P}_2 + 3t^2(1-t) \mathbf{P}_3 + t^3 \mathbf{P}_4$$



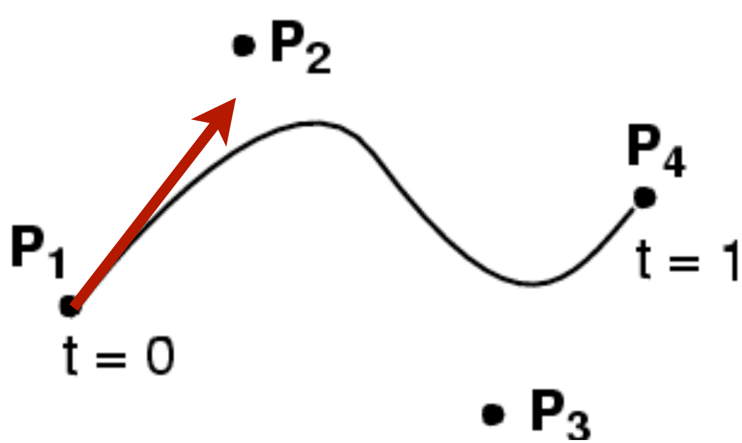
- $\mathbf{P'(t) = -3(1-t)^2 P_1 + [3(1-t)^2 - 6t(1-t)] P_2 + [6t(1-t) - 3t^2] P_3 + 3t^2 P_4}$

Sanity check: t=0; t=1

Can also write this using a matrix **B'** – try it out!

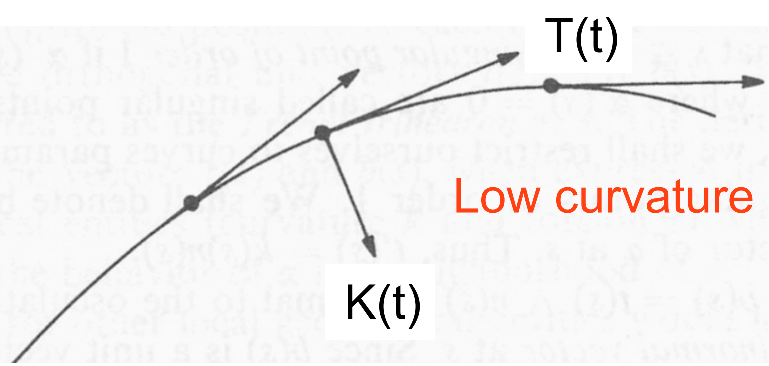
Tangent

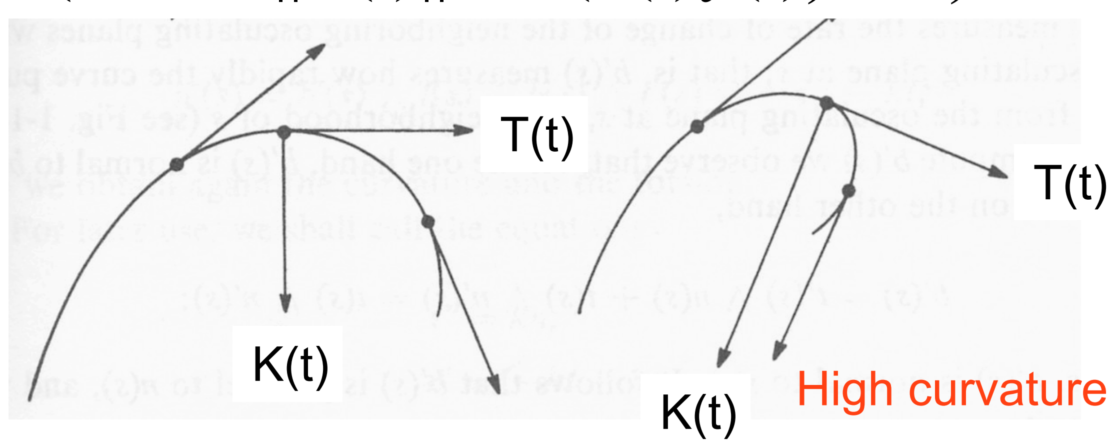
- The tangent to the curve P(t) can be defined as $T(t) = P'(t)/||P'(t)||$
  - normalized velocity, $||T(t)|| = 1$
- This provides us with one orientation for swept surfaces in a little while



Curvature

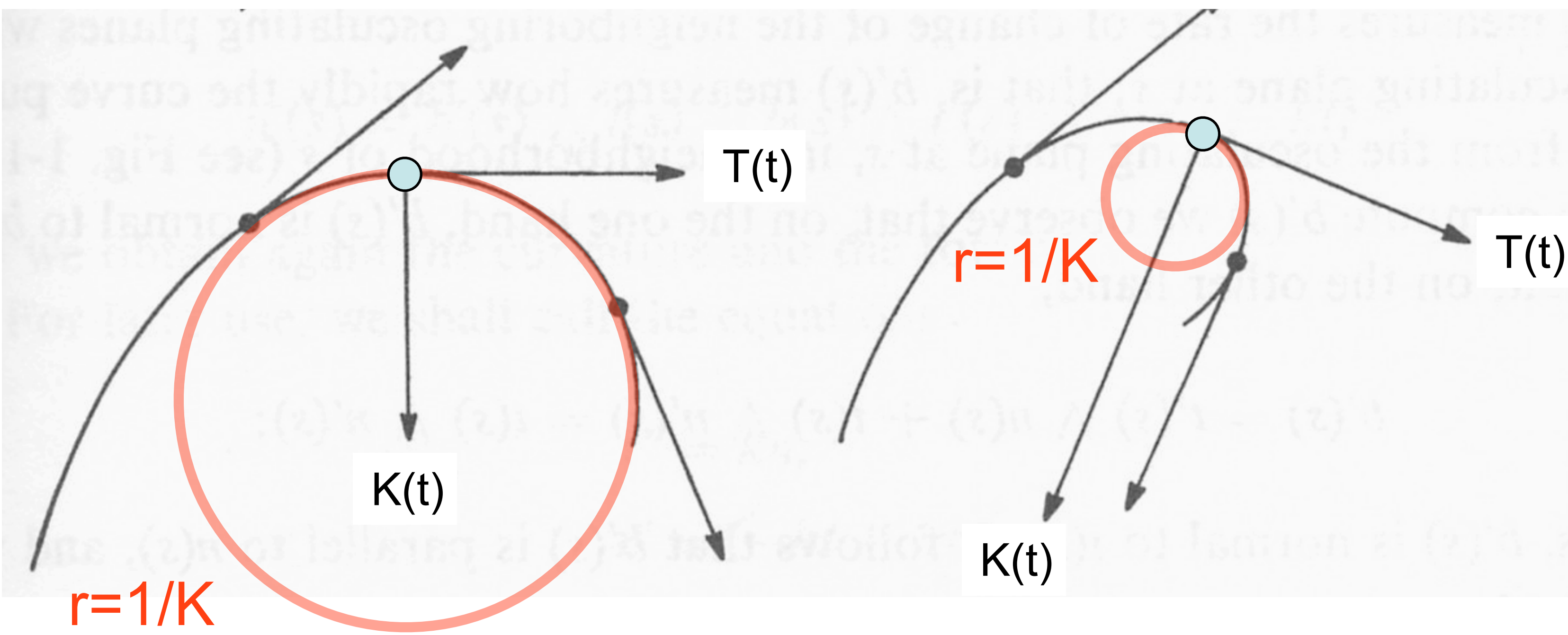
- Derivative of **unit** tangent (*not* just the 1st deriv.!)
  - K(t)=T'(t)
  - Magnitude ||K(t)|| is constant for a circle
  - Zero for a straight line
- Always orthogonal to tangent, ie. $K \cdot T = 0$
  - Can you prove this? (Hints: ||T(t)||=1, (x(t)y(t))'=?)





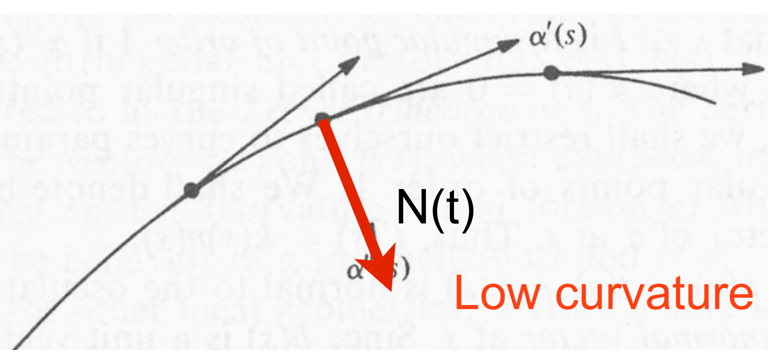
Geometric Interpretation

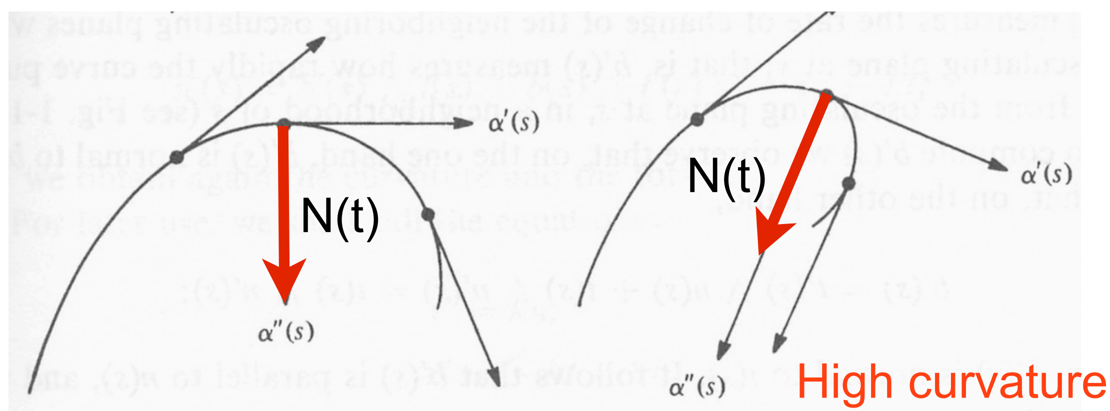
- K is zero for a line, constant for circle
  - What constant? 1/r
- 1/||K(t)|| is the radius of the circle that touches P(t) at *t* and has the same curvature as the curve



Curve Normal

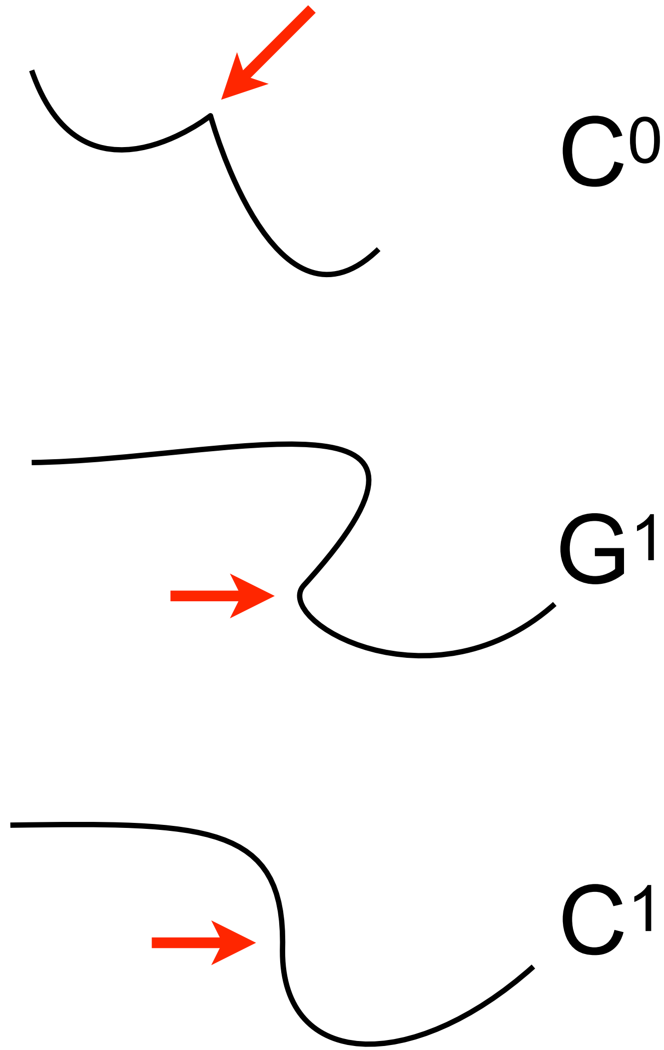
- Normalized curvature: T'(t)/||T'(t)||





Orders of Continuity

- C^0 = continuous
  - The seam can be a sharp kink
- G^1 = geometric continuity
  - Tangents **point to the same direction** at the seam
- C^1 = parametric continuity
  - Tangents **are the same** at the seam, implies G^1
- C^2 = curvature continuity
  - Tangents and their derivatives are the same

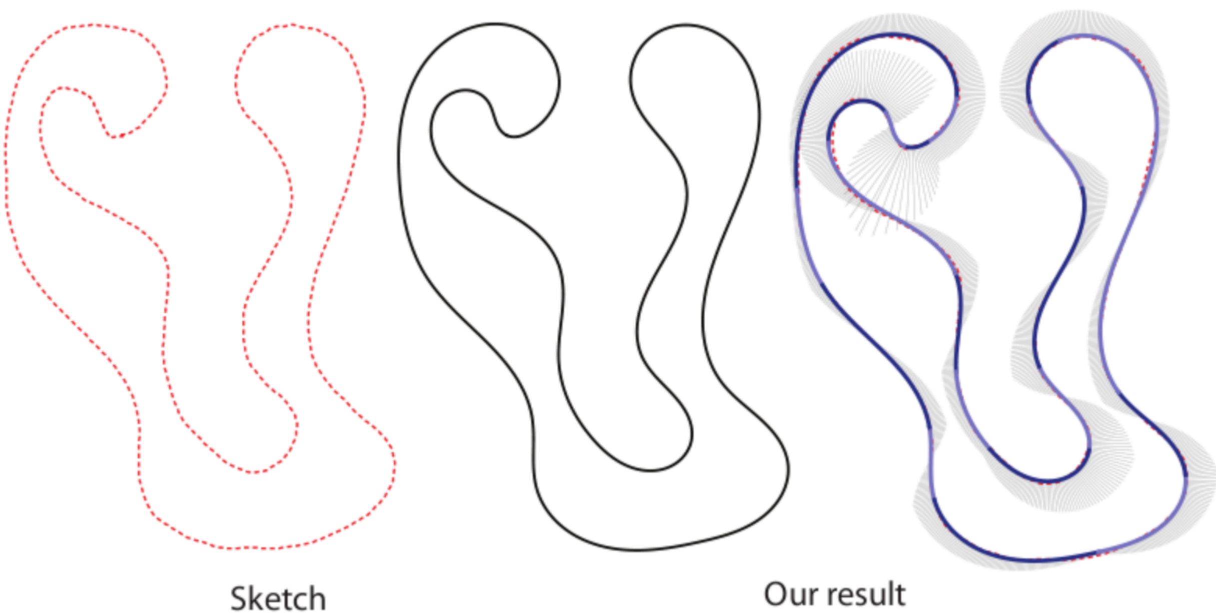


(Even nicer: Clothoid Splines)

- Curves with piecewise linear curvature
- See our paper from 2010 – this is now the basis of Illustrator's freehand drawing tool!

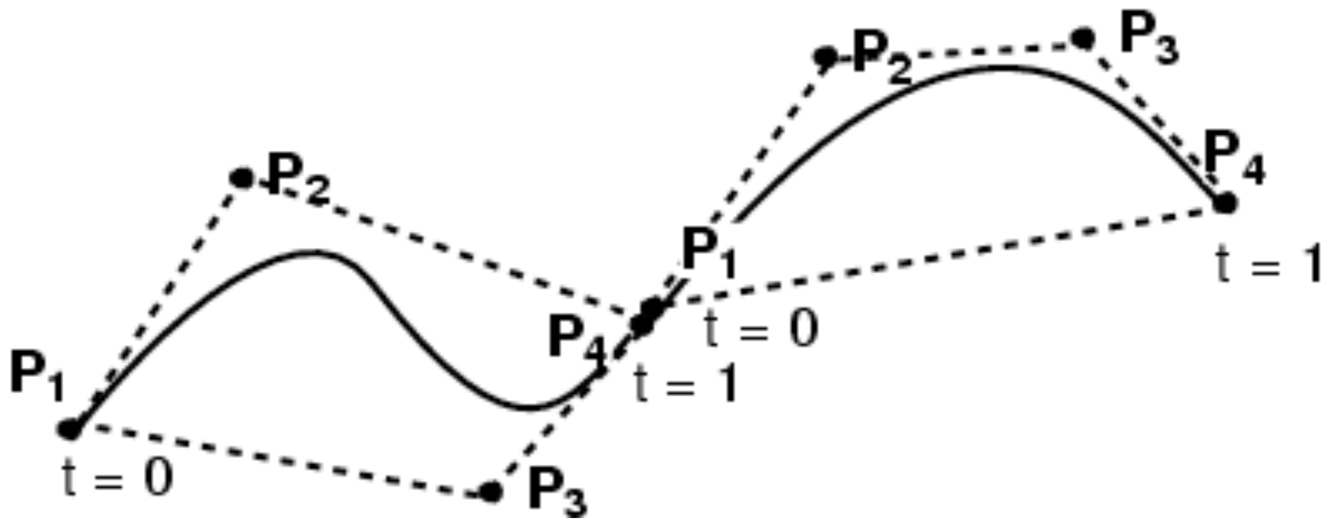
Sketching Clothoid Splines Using Shortest Paths

Ilya Baran Jaakko Lehtinen Jovan Popović



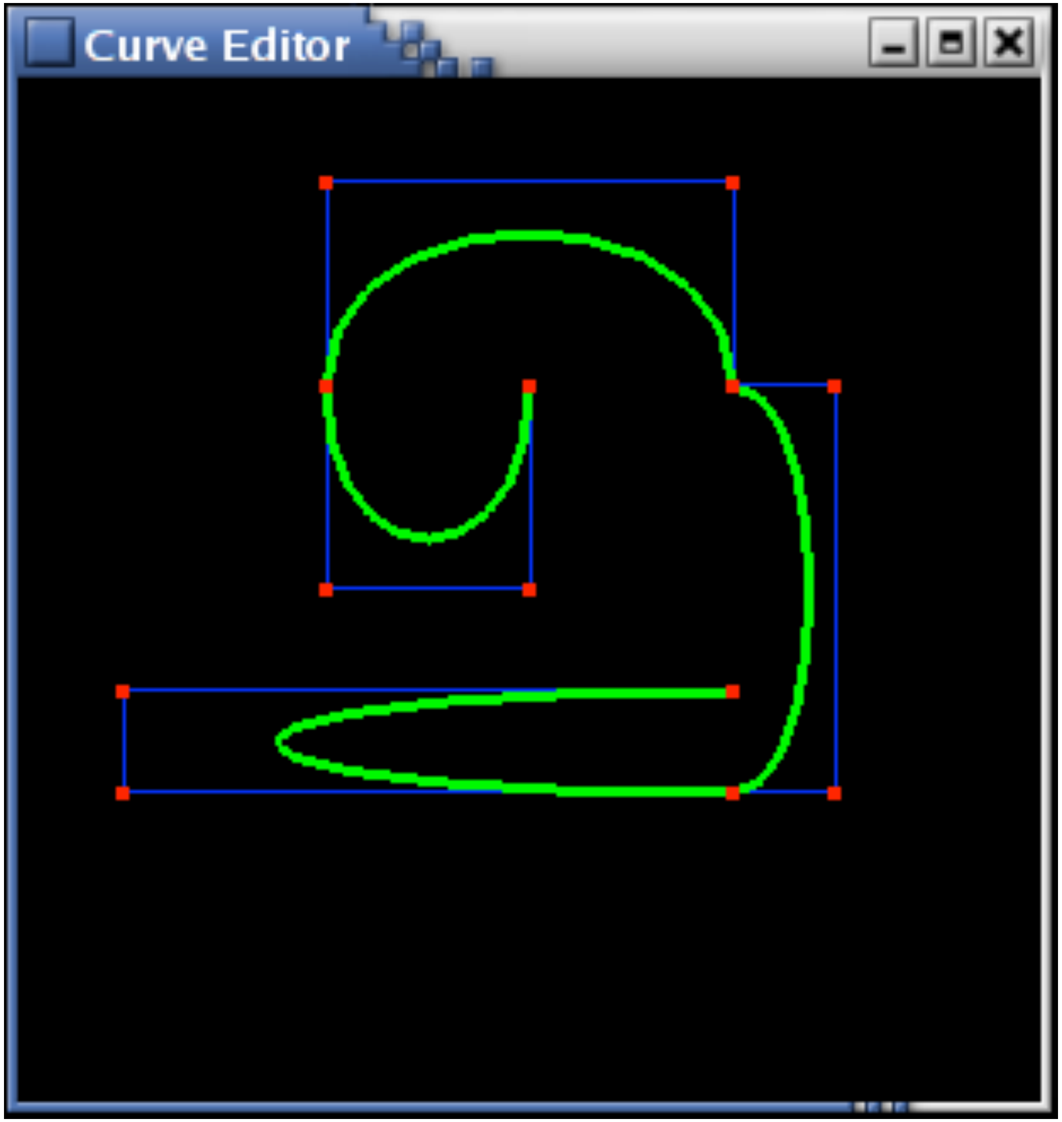
We obtain high-quality piecewise-clothoid curves from hand-drawn sketches.

Connecting Cubic Bézier Curves



- How can we guarantee C^0 continuity?
- How can we guarantee G^1 continuity?
- How can we guarantee C^1 continuity?
- C^2 and above gets difficult

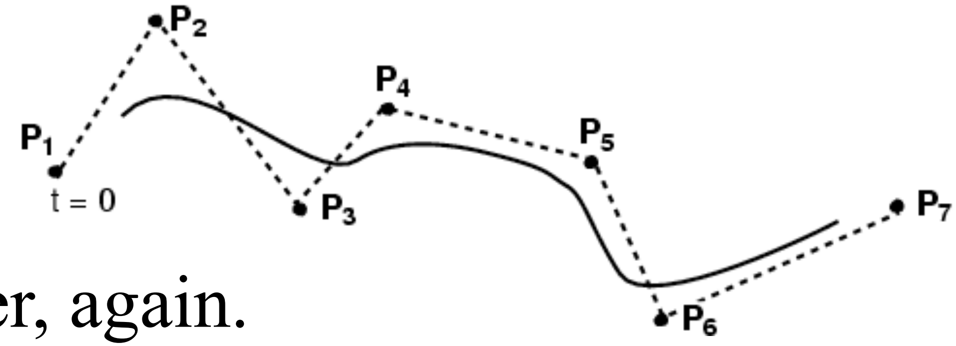
Connecting Cubic Bézier Curves



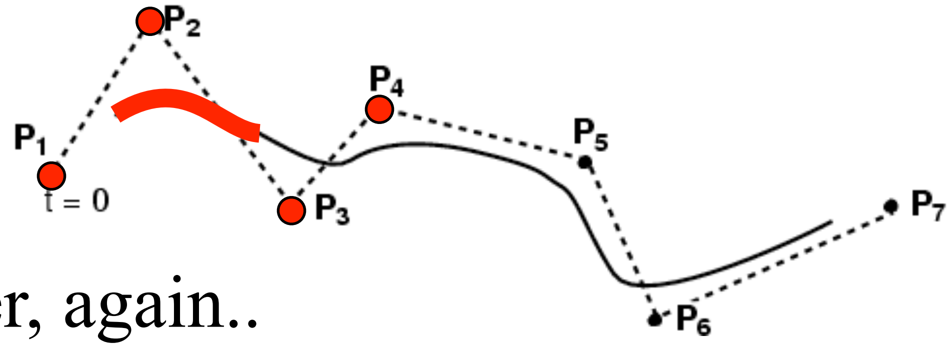
- Where is this curve
  - C^0 continuous?
  - G^1 continuous?
  - C^1 continuous?
- What's the relationship between:
  - the # of control points, and the # of cubic Bézier subcurves?

Cubic B-Splines

- $\geq$ 4 control points
- Locally cubic
  - Cubics chained together, again.

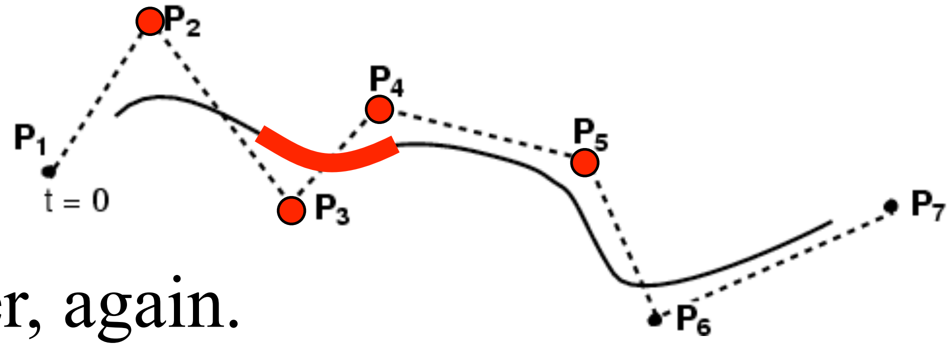


Cubic B-Splines



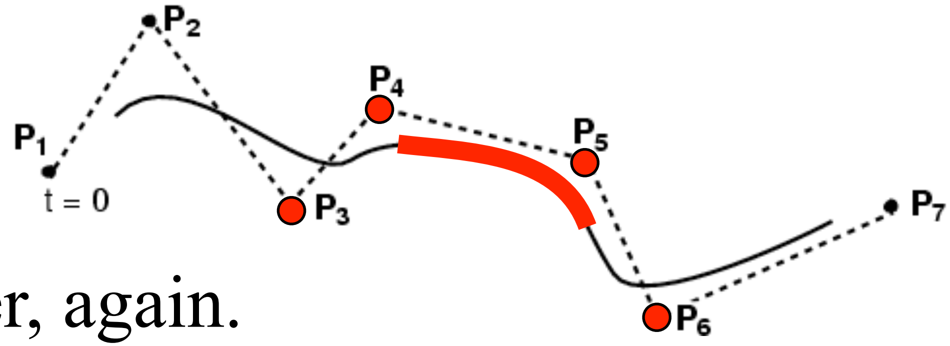
- $\geq$ 4 control points
- Locally cubic
  - Cubics chained together, again..
  - BUT with a sliding window of 4 control points!

Cubic B-Splines



- $\geq$ 4 control points
- Locally cubic
  - Cubics chained together, again.
  - BUT with a sliding window of 4 control points!

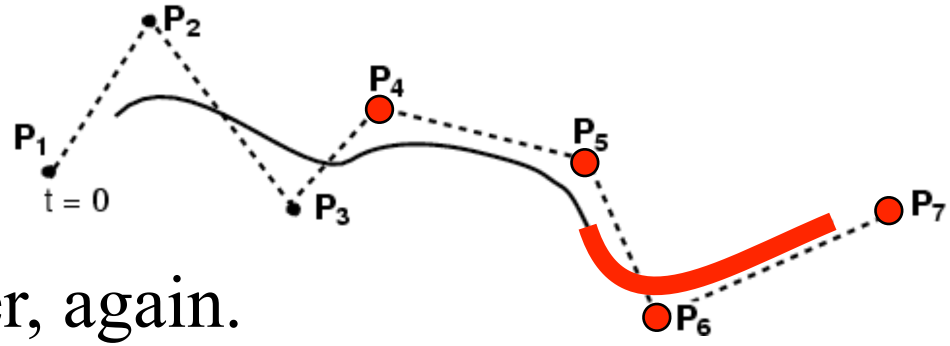
Cubic B-Splines



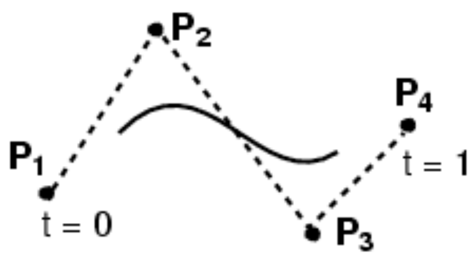
- ≥ 4 control points
- Locally cubic
  - Cubics chained together, again.
  - BUT with a sliding window of 4 control points!

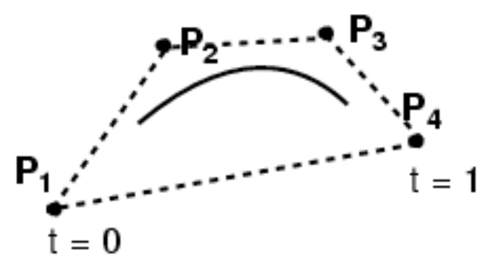
Cubic B-Splines

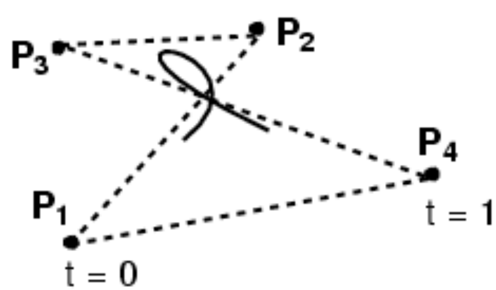
- $\geq$ 4 control points
- Locally cubic
  - Cubics chained together, again.



- Curve is not constrained to pass through any control points

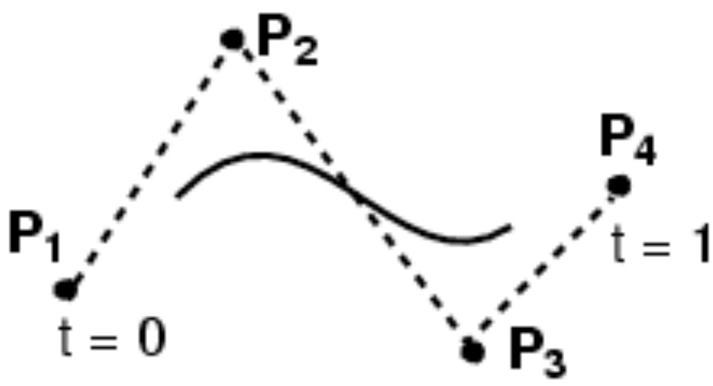


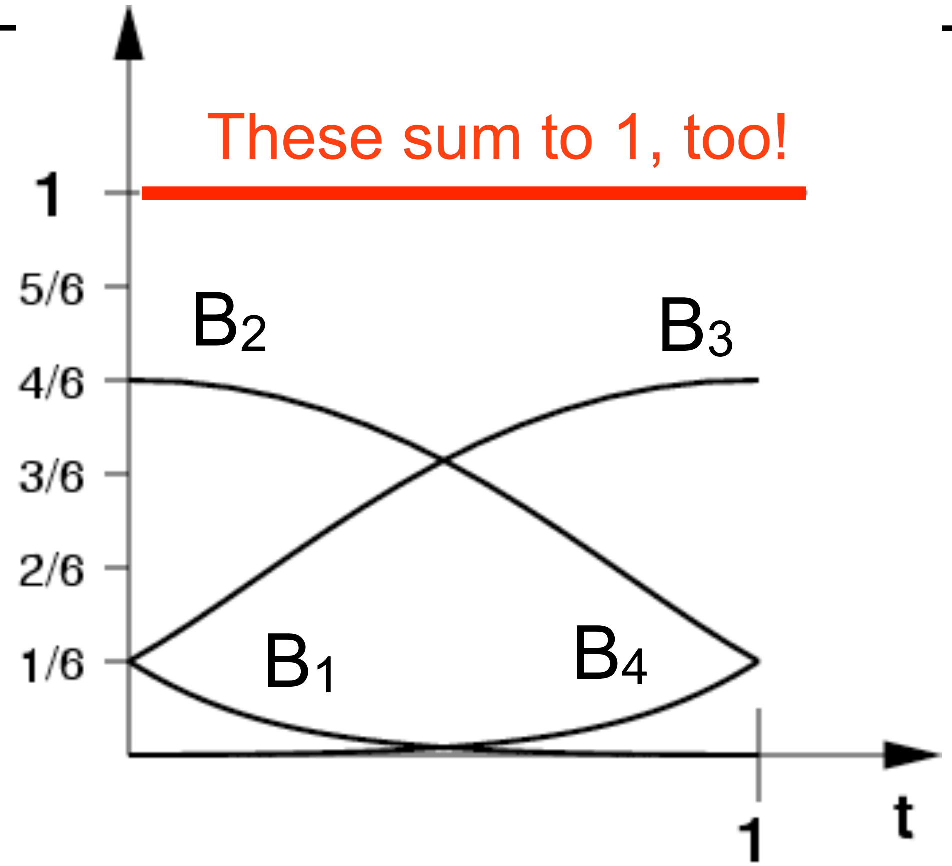




A BSpline curve is also bounded by the convex hull of its control points.

Cubic B-Splines: Basis

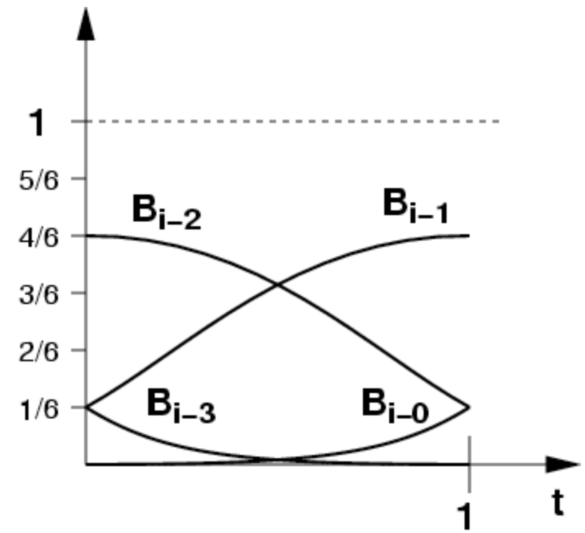




$$B_1(t) = \frac{1}{6}(1-t)^3 \qquad B_2(t) = \frac{1}{6}(3t^3 - 6t^2 + 4)$$

$$B_3(t) = \frac{1}{6}(-3t^3 + 3t^2 + 3t + 1) \qquad B_4(t) = \frac{1}{6}t^3$$

Cubic B-Splines: Basis



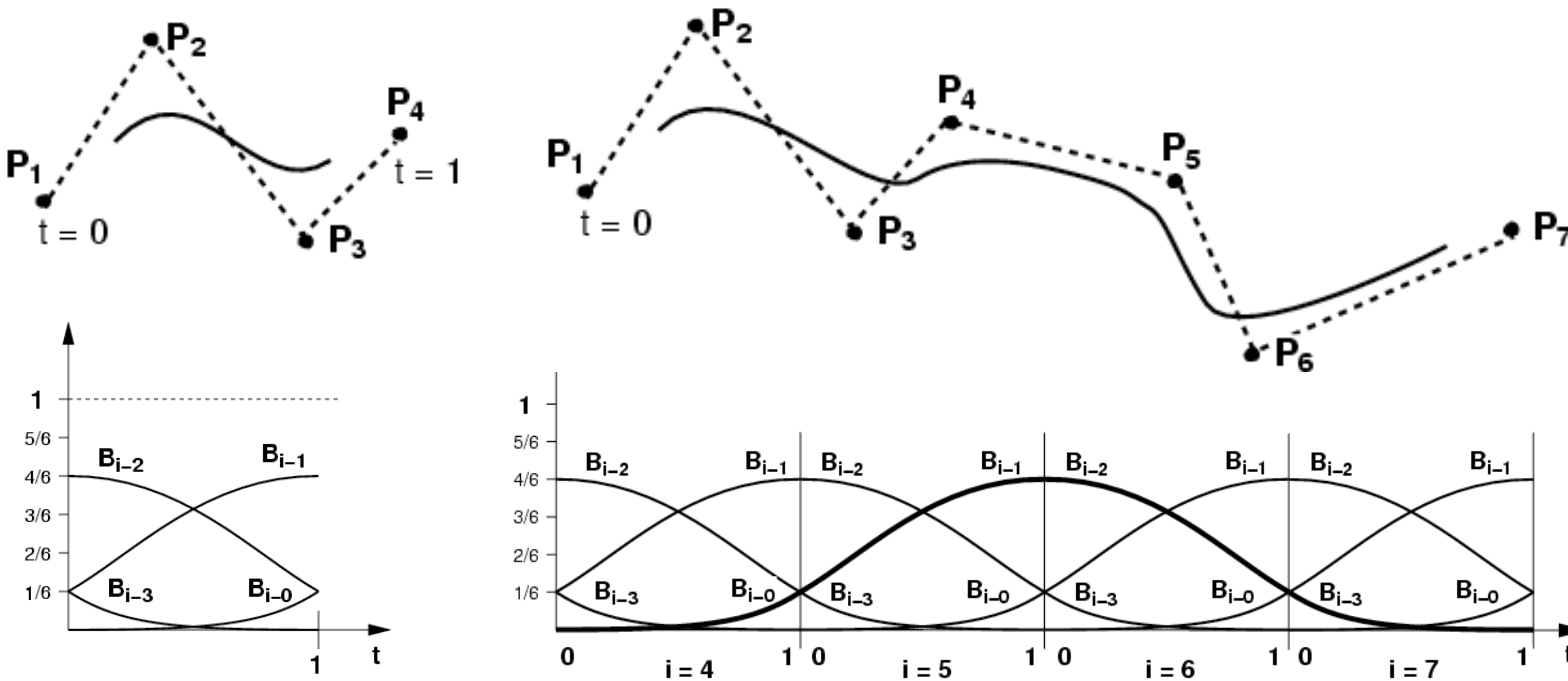
$$Q(t) = \frac{(1-t)^3}{6} P_{i-3} + \frac{3t^3 - 6t^2 + 4}{6} P_{i-2} + \frac{-3t^3 + 3t^2 + 3t + 1}{6} P_{i-1} + \frac{t^3}{6} P_i$$

$$Q(t) = \mathbf{GBT(t)}$$

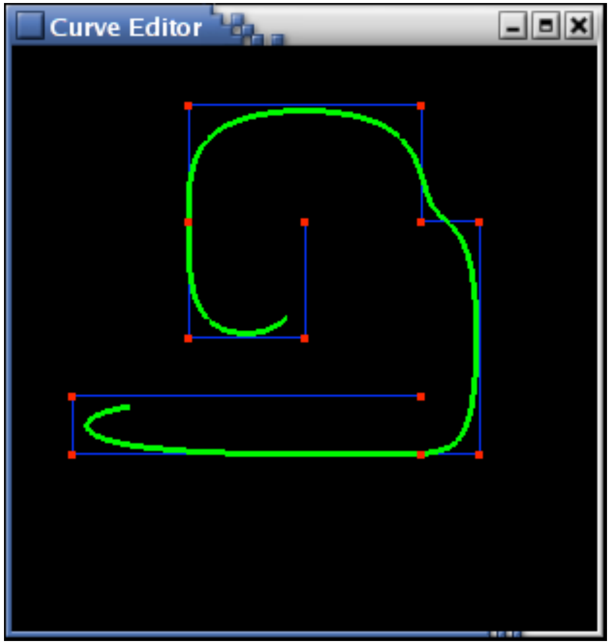
$$\boxed{B_{B-Spline} = \frac{1}{6} \begin{pmatrix} 1 & -3 & 3 & -1 \\ 4 & 0 & -6 & 3 \\ 1 & 3 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}}$$

Cubic B-Splines

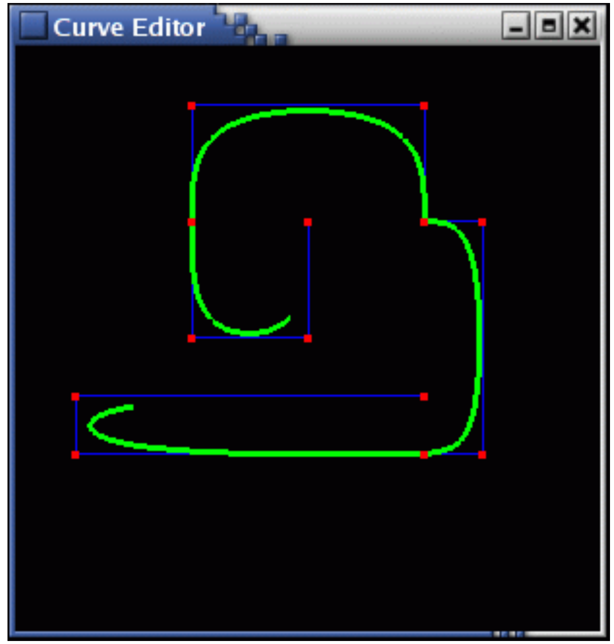
- Local control (windowing)
- Automatically C^2, and no need to match tangents!



B-Spline Curve Control Points

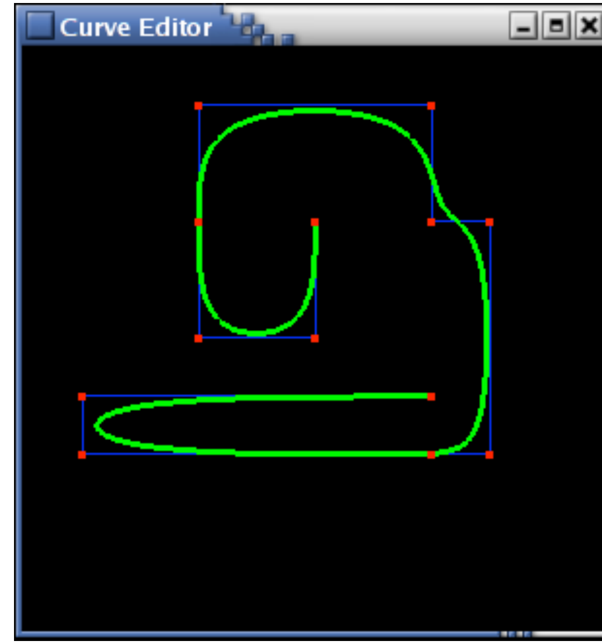


Default BSpline



BSpline with derivative discontinuity

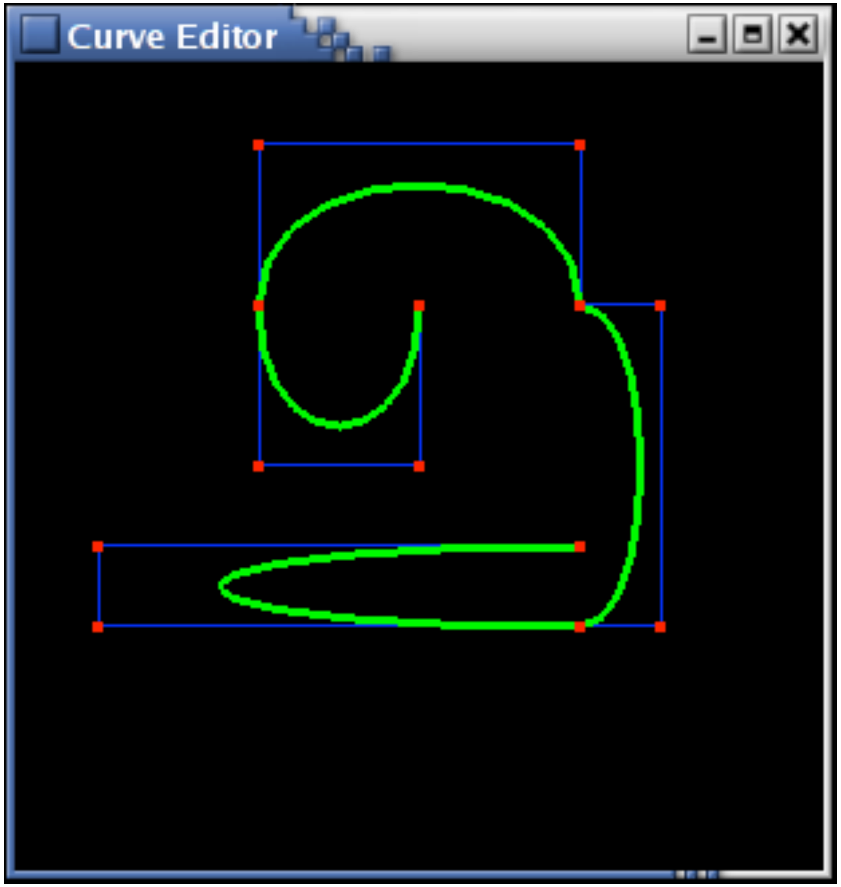
Repeat interior control point



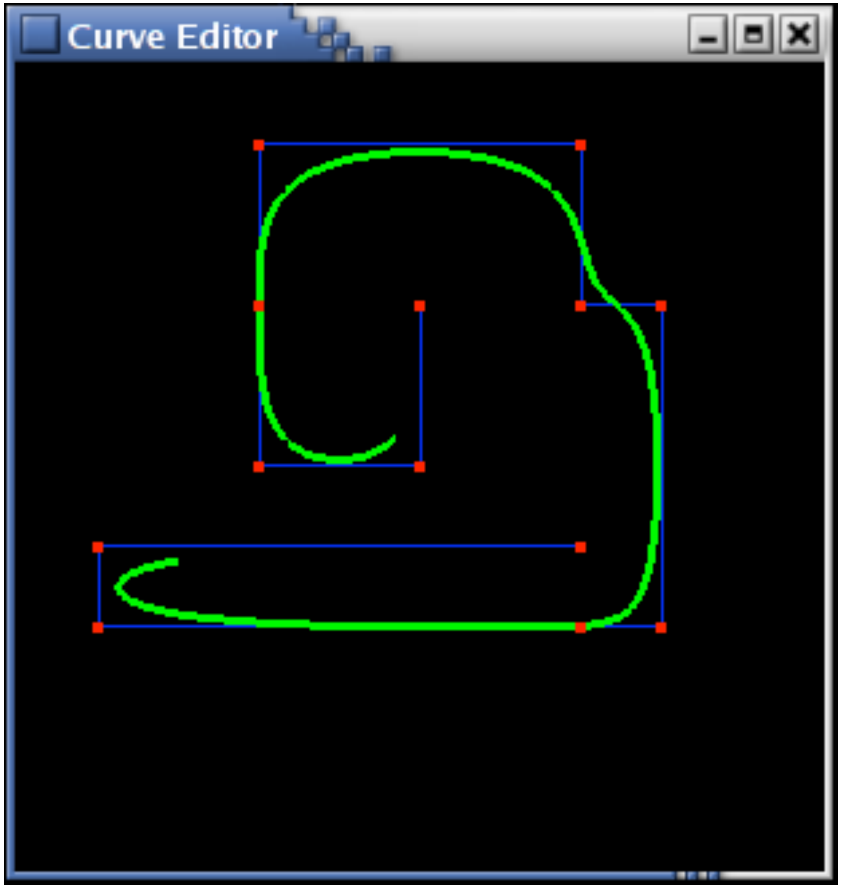
BSpline which passes through end points

Repeat end points

Bézier ≠ B-Spline



Bézier



BSpline

But both are cubics, so one can be converted into the other!

Converting between Bézier & BSpline

- Simple with the basis matrices!
  – Note that this only works for a single segment of 4 control points

$$B_{Bezier} = \begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- $P(t) = \mathbf{G}\ \mathbf{B}_1\ \mathbf{T}(t) =$
  $\mathbf{G}\ \mathbf{B}_1\ (\mathbf{B}_2^{-1}\mathbf{B}_2)\ \mathbf{T}(t) =$
  $(\mathbf{G}\ \mathbf{B}_1\ \mathbf{B}_2^{-1})\ \mathbf{B}_2\ \mathbf{T}(t)$

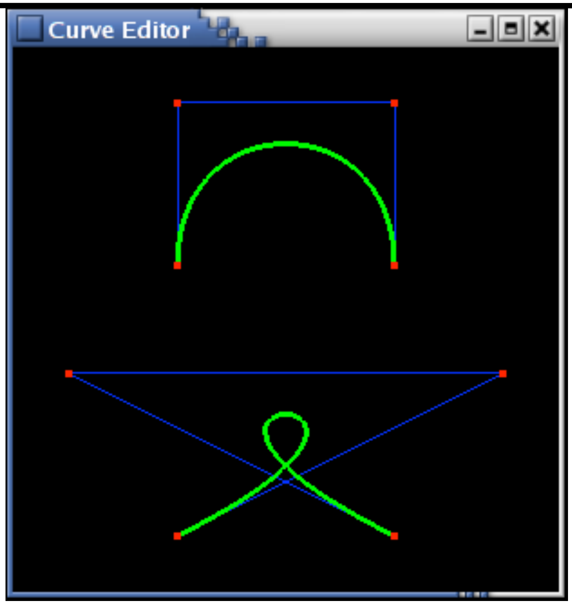
$$B_{B-Spline} = \frac{1}{6}\begin{pmatrix} 1 & -3 & 3 & -1 \\ 4 & 0 & -6 & 3 \\ 1 & 3 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

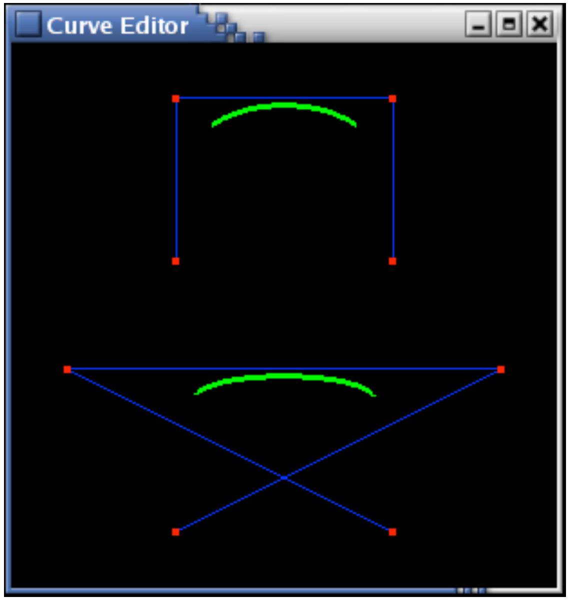
- $\mathbf{G}\ \mathbf{B}_1\ \mathbf{B}_2^{-1}$ are the control points for the segment in new basis.

$$Q(t) = \mathbf{GBT(t)} = \text{Geometry } \mathbf{G} \cdot \text{Spline Basis } \mathbf{B} \cdot \text{Power Basis } \mathbf{T(t)}$$

Converting between Bézier & BSpline

original control points as Bézier

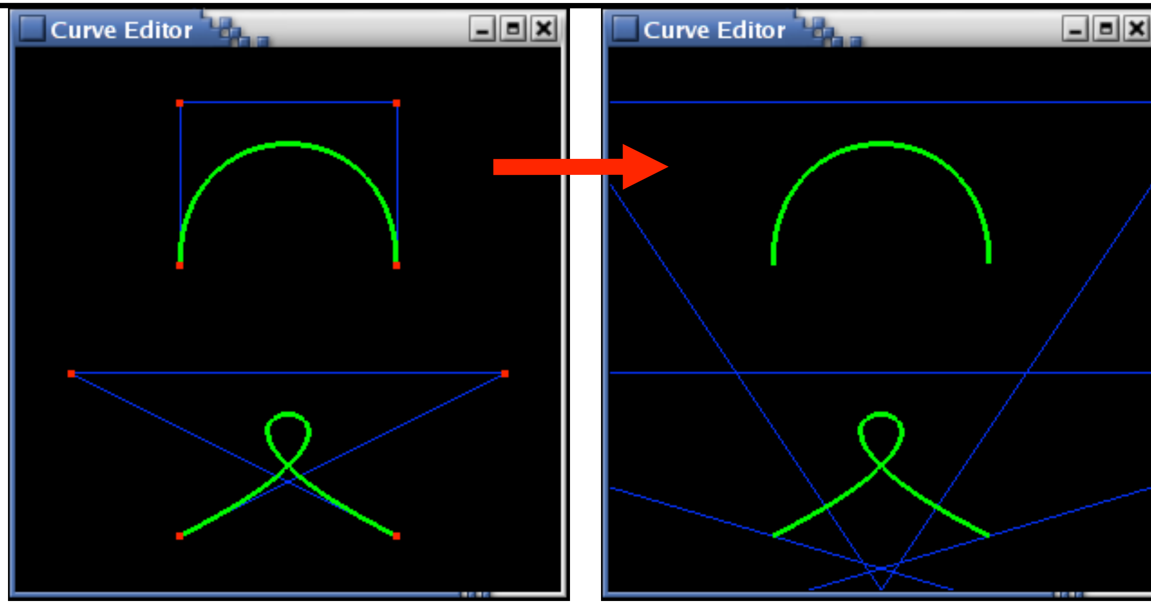




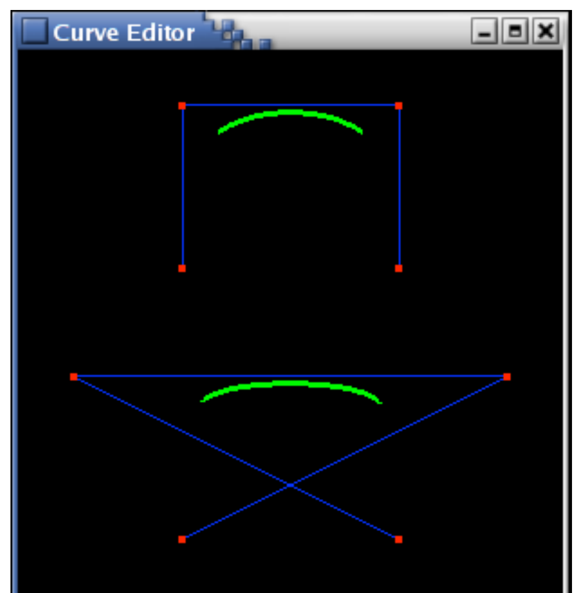
original control points as BSpline

Converting between Bézier & BSpline

original control points as Bézier



new BSpline control points to match Bézier

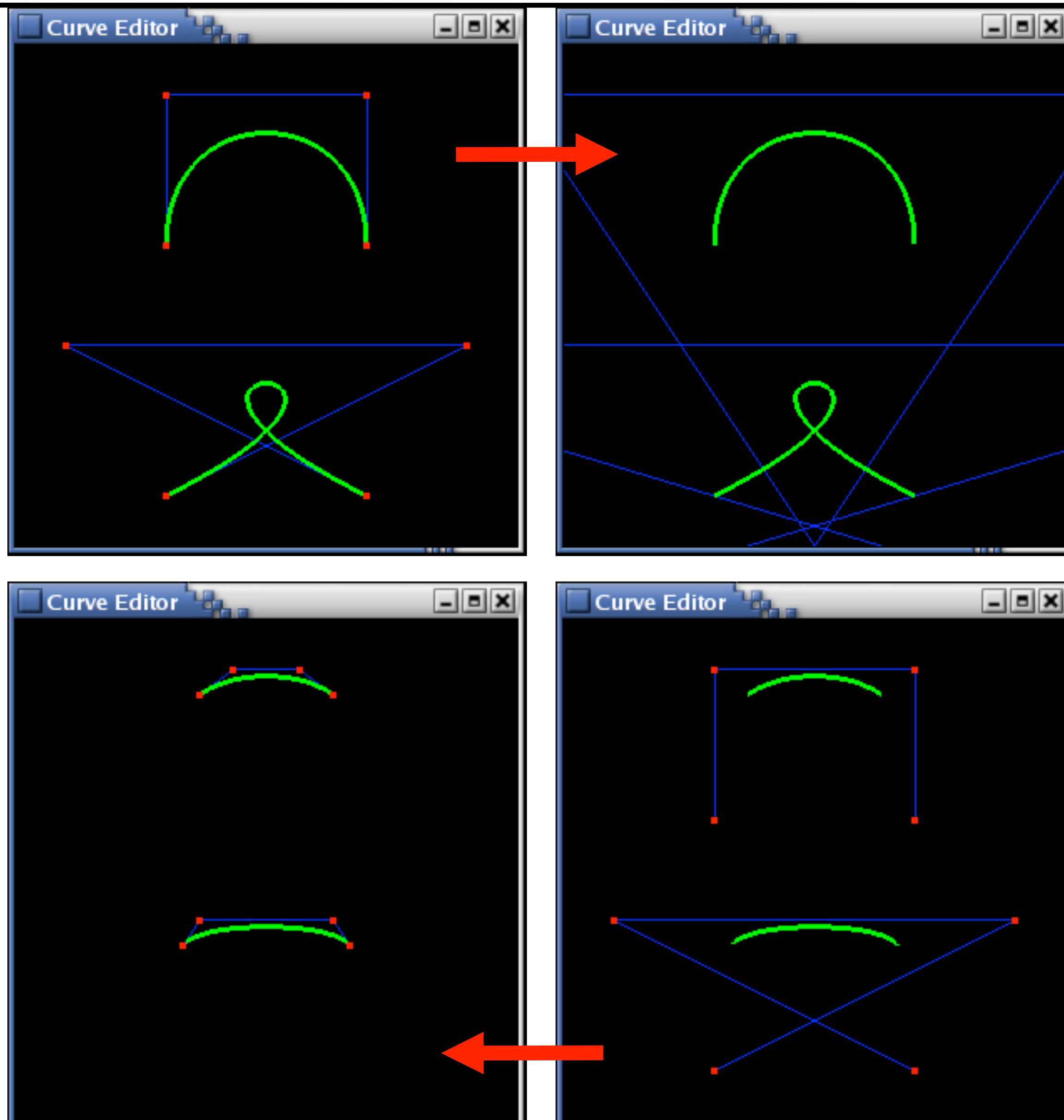


original control points as BSpline

Converting between Bézier & BSpline

original control points as Bézier

new BSpline control points to match Bézier

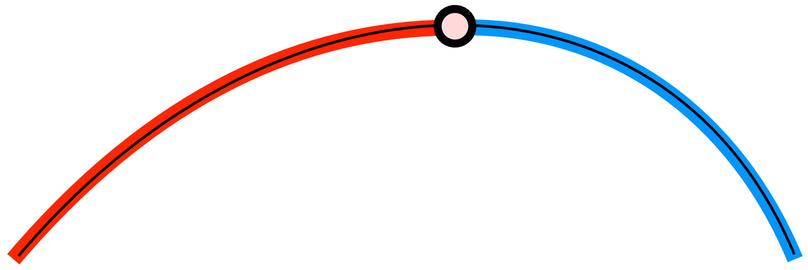


new Bézier control points to match BSpline

original control points as BSpline

Why Bother with B-Splines?

- Automatic C^2 is nice!
- Also, B-Splines can be split into segments of non-uniform length without affecting the global parametrization.
  - “Non-uniform B-Splines”
  - We’ll not do this, but just so you know.



NURBS (Generalized B-Splines)

- Rational cubics
  - Use homogeneous coordinates, just add *w* !
    - Provides a "tension" parameter to control points

- NURBS: Non-Uniform Rational B-Spline
  - **non-uniform** = different spacing between the blending functions, a.k.a. "knots"
  - **rational** = ratio of cubic polynomials (instead of just cubic)
    - implemented by adding the homogeneous coordinate *w* into the control points.