

Lecture 11:

Even More Curves

Today

1. The main points of curves
2. The Train (to motivate fancier curve things)
3. Arc Length
4. TCB Cardinals
5. B-Splines ($C(2)$ curves)

Stuff You Need To Know About Curves

1. Continuity Conditions

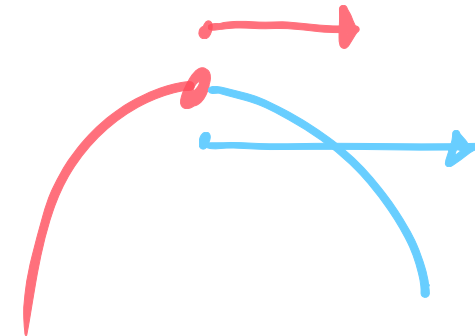
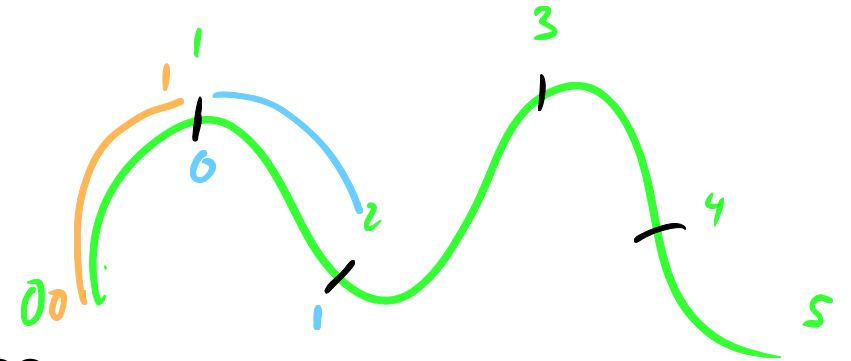
We build curves from pieces

One parameter for curve, or one parameter per piece

We care about continuity where the pieces meet

C continuity (derivatives)

G continuity (directions of derivatives)



Stuff You Need to Know About Curves

2. Polynomial Pieces

Canonical Form

$$f(u) = \underline{a_0} \underline{u^0} + \underline{a_1} \underline{u^1} + \underline{a_2} \underline{u^2} + \dots$$

Basis Function form

$$f(u) = \underline{p_1} \uparrow \underline{b_1(u)} + \underline{p_2} \uparrow \underline{b_2(u)} + \underline{p_3} \uparrow \underline{b_3(u)} + \dots$$

Stuff You Need to Know About Curves

3. Cubic Forms

Canonical Form

$$f(u) = \underline{a_0}u^0 + \underline{a_1}u^1 + \underline{a_2}u^2 + \underline{a_3}u^3$$

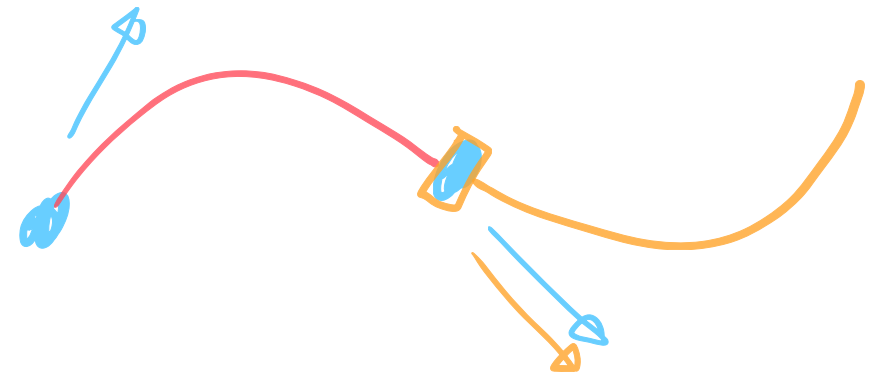
Hermite Form: beginning and end

$$p_0 = \underline{f(0)}$$

$$p_1 = \underline{f(1)}$$

$$p'_0 = \underline{f'(0)}$$

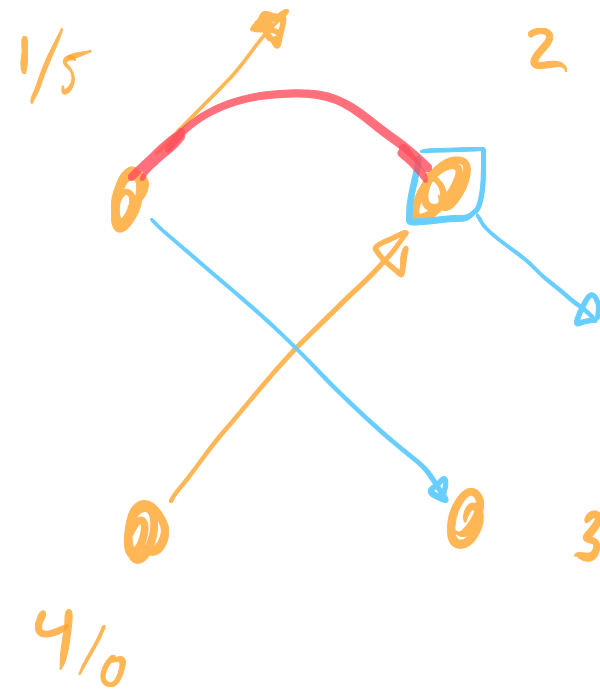
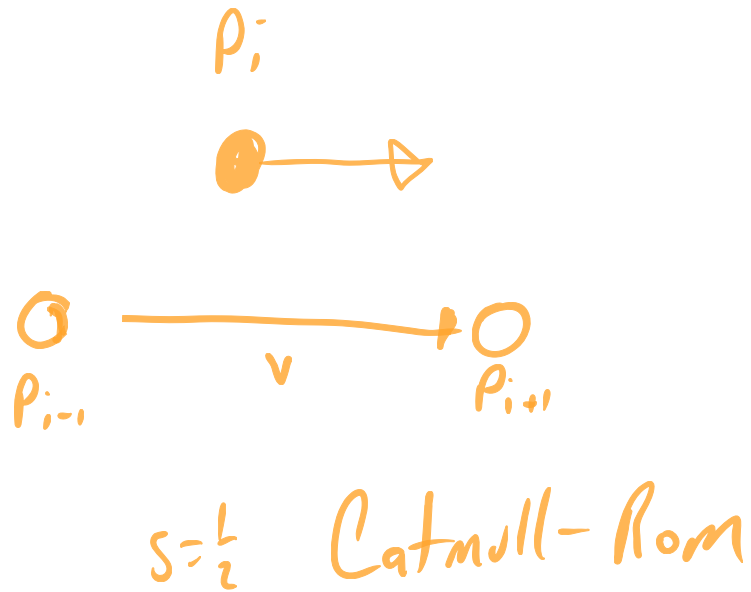
$$p'_1 = \underline{f'(1)}$$



Stuff You Need to Know About Curves

4. Cardinal Interpolation

$$p'_i = s(p_{i+1} - p_{i-1})$$



Thing we glossed over

5. Bezier Cubic (segment)

4 points

Interpolate endpoints

Tangents = 3 * vector between first (last) points

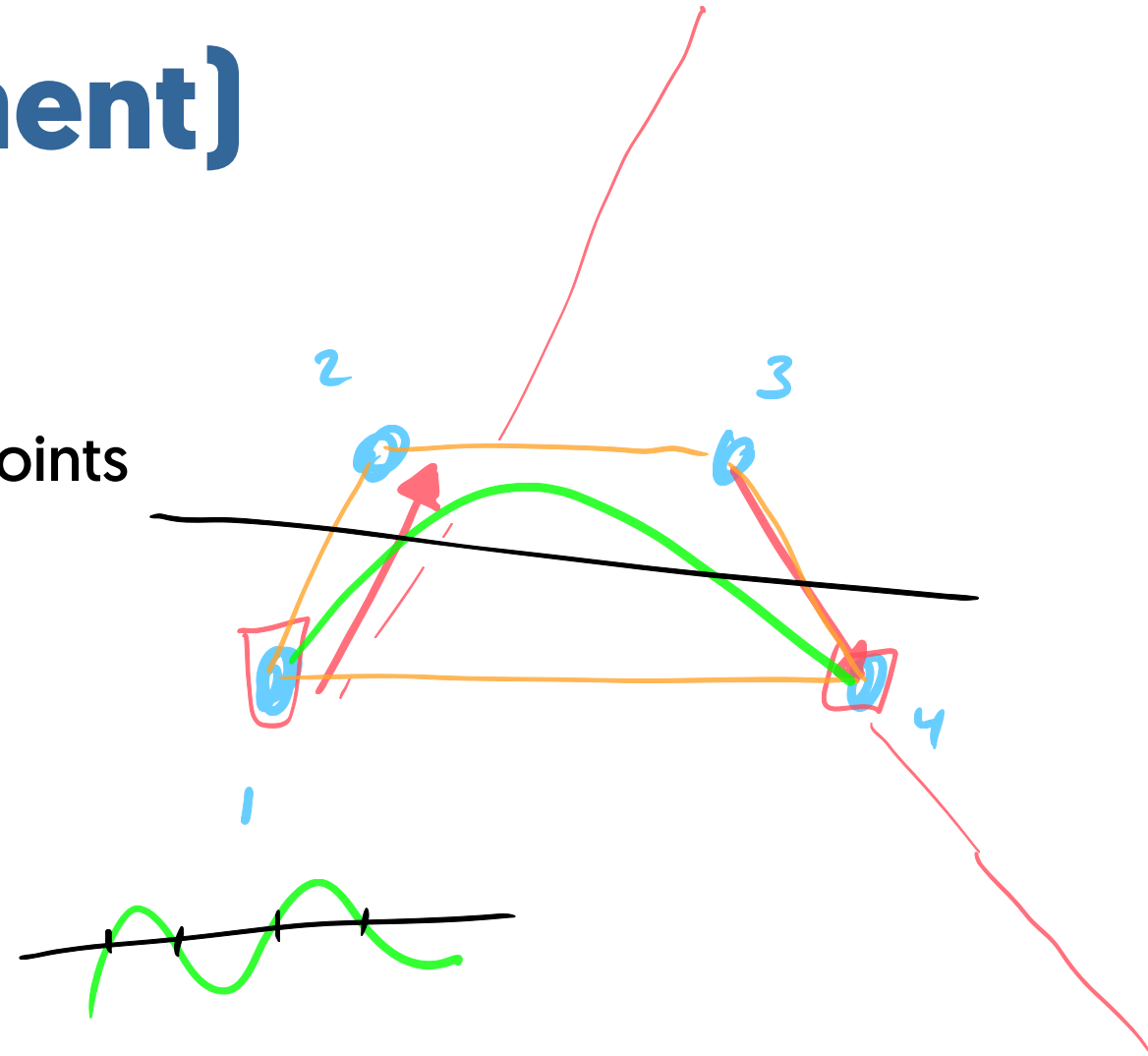
Convex Hull Property

Variation Diminishing Property

Symmetry

Locality

Cool Algorithms (Geometric)



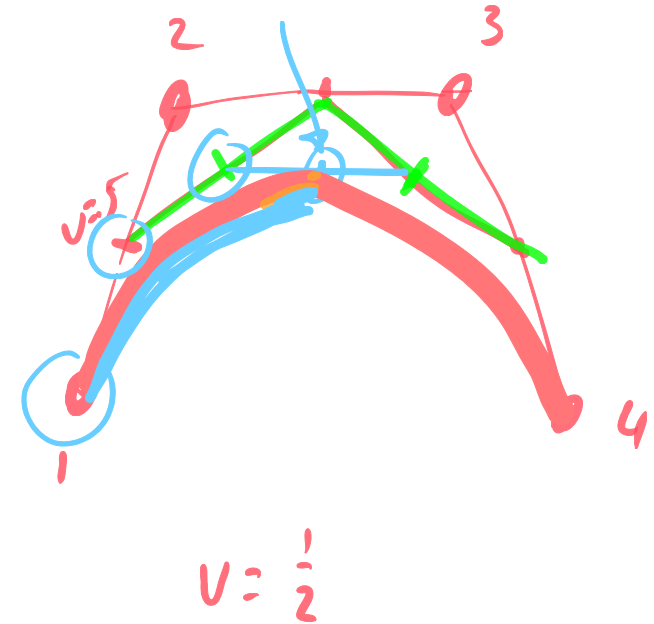
Things you need to be able to do

6. Geometric Algorithms for Beziars

DeCasteljau construction

1. compute values by repeated interpolation
2. subdivide curves

Useful for deriving properties and basis functions



Things you need to know about Curves

7. The Generality of Beziers

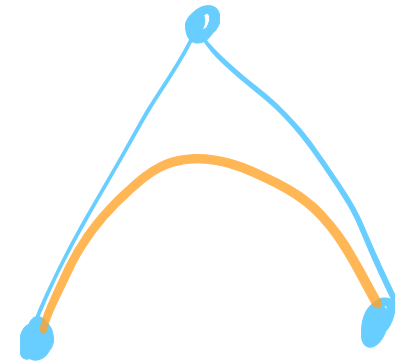
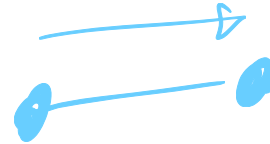
Works for any degree! [$d = \underline{\underline{n}} - 1$ for n points]

2 points (line segment)

3 points (quadratics)

4 points (cubics)

etc.



All properties hold

Scale vector (first points) by degree of curve

Things you should be able to do

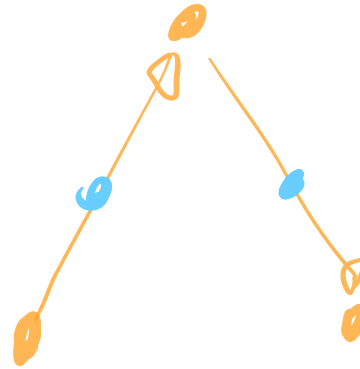
8. Convert Between Cubic Forms

APIs generally have Beziers (cubics)

Bezier to Hermite (for cubics)

Cardinal to Hermite

Beziers of different degrees



What can we do with this?

The Train!

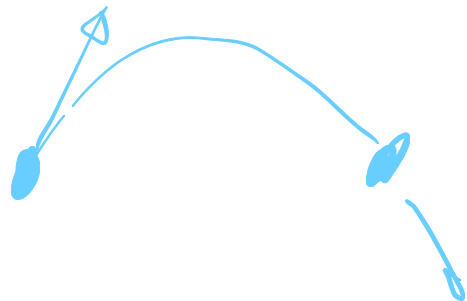
This has been part of CS559 since 1999 (the first time I taught it!)

- Good for showing curve ideas!
- 3D Demo (2014)
- 2D Demo (2019)

What's Good About Cardinals?

1. $C(1)$
2. Local
3. Interpolating

Sketching Cardinals (approximate $s = .5$)



How to draw a cardinal?

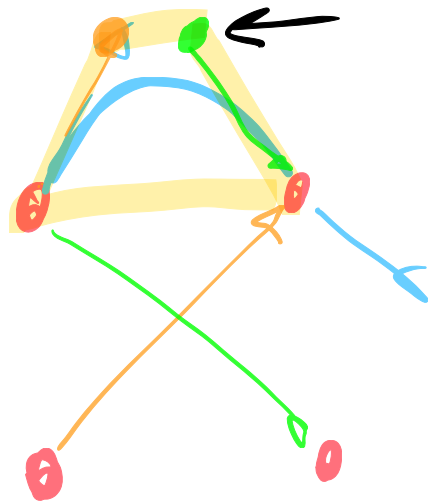
1. try a bunch of u values
2. convert to a Bézier (draw with the API)



How far does the cardinal go?

It goes outside of its points.

Convert to Bézier!

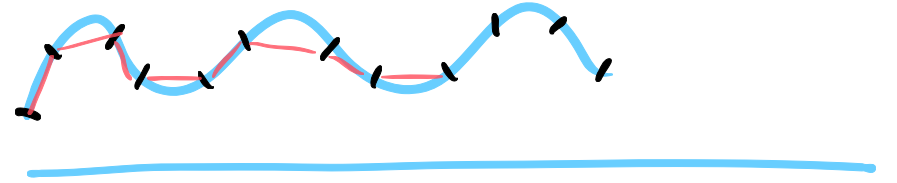


Arc Length

If you straighten out the curve - how long is it

Integral of the magnitude of the tangent

Approximate curve with straight lines

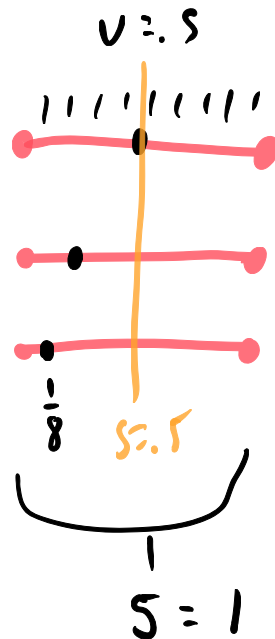


Arc Length Parameterization?

- equal steps in u ← natural parameterization
- equal steps in distance ← arc length

consider:

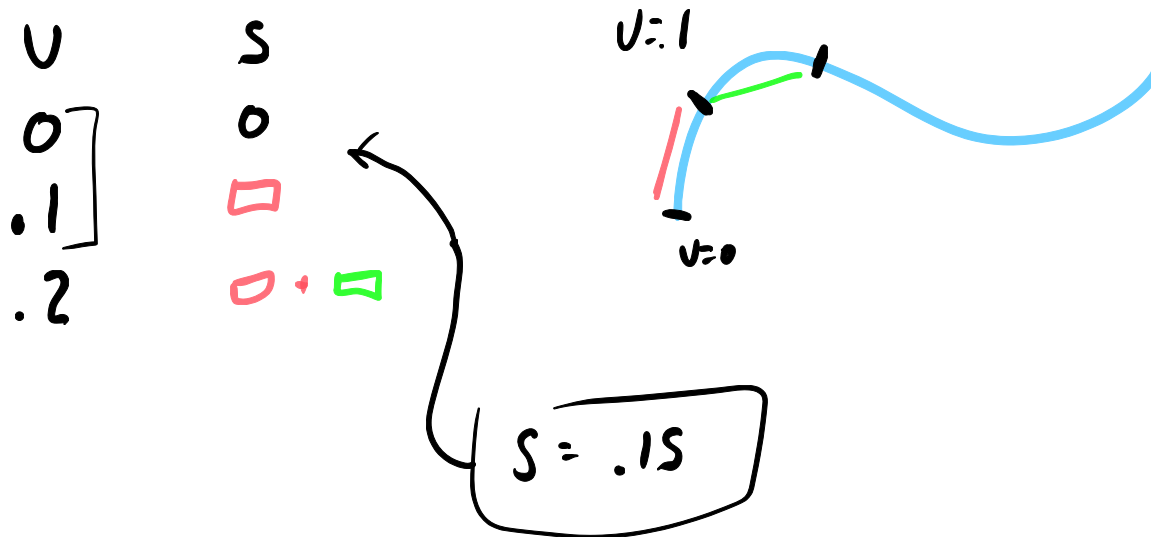
- $f(u) = u$
- $f(u) = u^2$
- $f(u) = u^3$



How to implement Arc Length Parameterization?

Hard to do analytically

Approximate *numerically* (make a table, solve, ...)



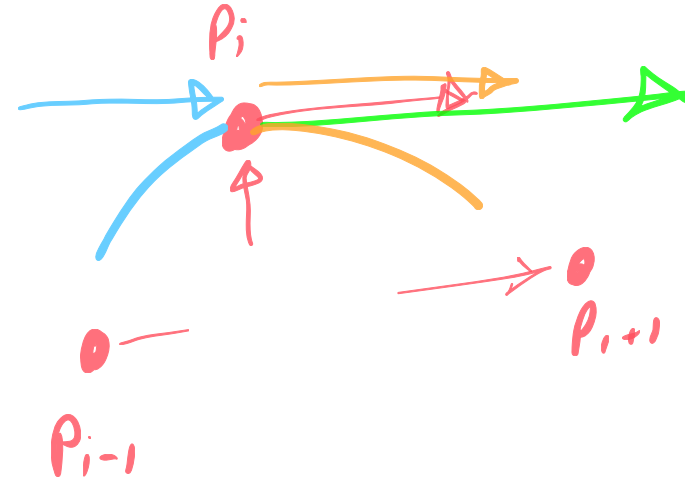
Fancier Cardinals: TCB curves

- tension $c\text{-Rem} = .5$

- continuity 

- bias

Used by animators to get more control

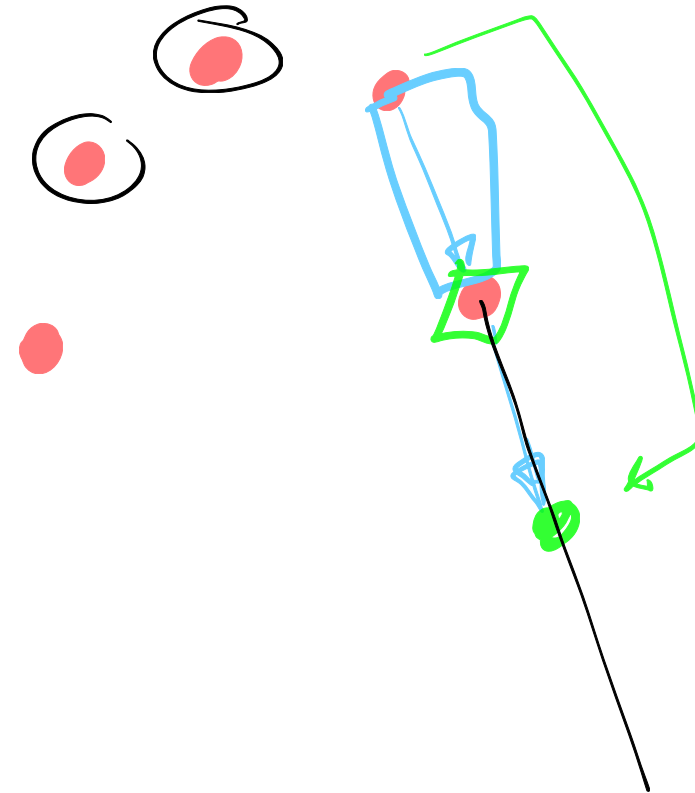


What's Good About Béziers?

1. Interpolate end-points, stay in Convex Hull
2. Tangents at ends based on points
3. Good algorithms (geometric and algebraic)
4. Variation-Diminishing (limited wiggles), Affine Invariance
5. General (any degree)
6. And other properties...

What's not good about Béziers

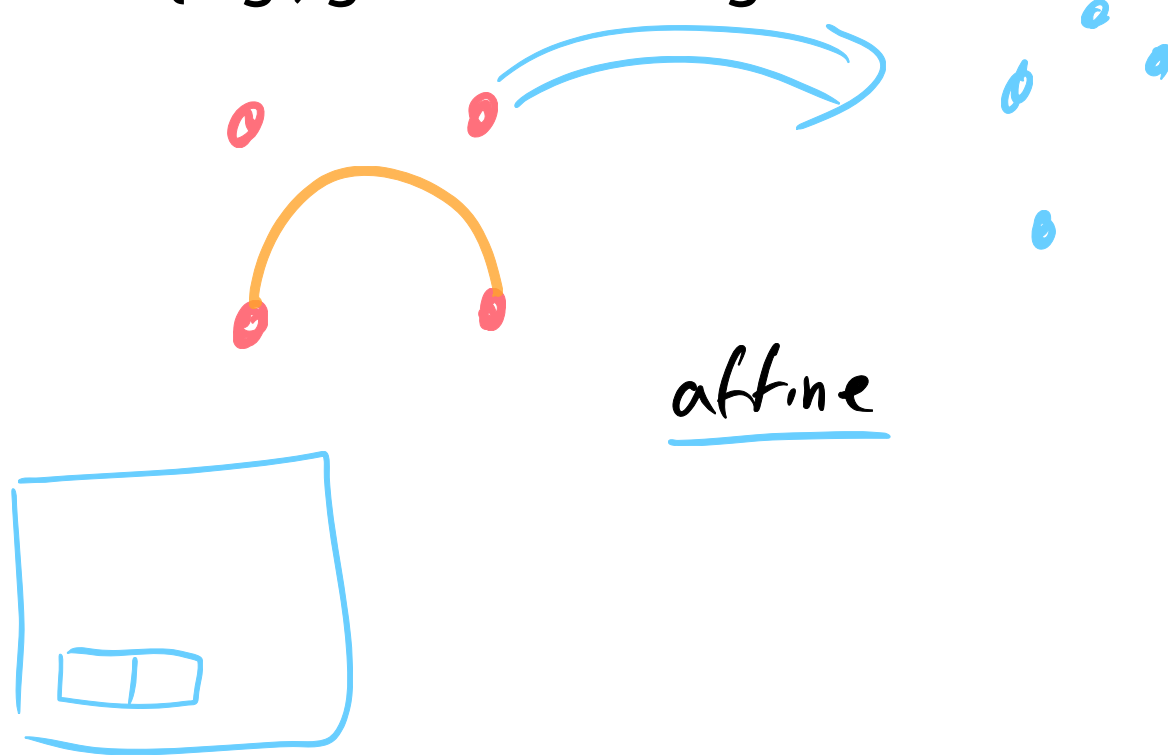
1. Not projective invariant
2. Polynomial can't represent conics
 - no exact circles!
3. Hard to get better than C(1)/G(1)
4. Sometimes we want **interpolation**



Projective Invariance

Béziers are **affine invariant**

Projection (e.g., general homogeneous coords) requires division



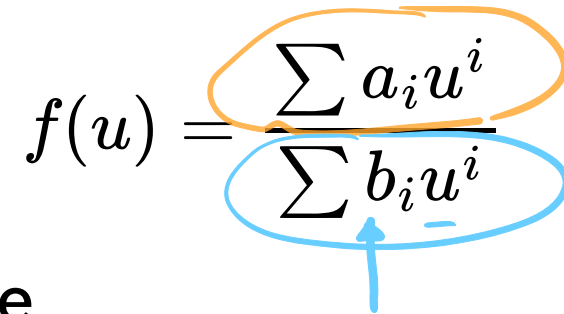
Limited Shapes

Polynomials (like Béziars) can't represent some shapes

Exact circles (and other conic sections)

Rational Polynomial Curves

Represent a curve as the **ratio** of two polynomials

$$f(u) = \frac{\sum a_i u^i}{\sum b_i u^i}$$
The equation $f(u) = \frac{\sum a_i u^i}{\sum b_i u^i}$ is shown. The numerator $\sum a_i u^i$ is circled in orange. The denominator $\sum b_i u^i$ is circled in blue. A blue arrow points upwards from the denominator towards the numerator.

Allows for projective invariance

Allows for more shapes (circles, conic sections, ...)

Very complicated - and usually not used in Computer Graphics

Used in mechanical design where exact shapes are required

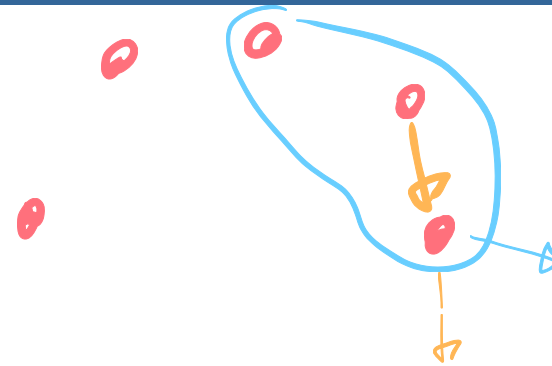
Better than C(1)/G(1)

requires aligning multiple points

Note:

If we want a very smooth curve, we can make a high degree Bézier

But getting continuity between segments can be tricky



What about Interpolation?

1. Cardinal Splines $C(1)$
2. High-Order Polynomials (smooth)
3. Natural Splines $C(2)$

B-Splines

This should be a big topic - but we will scratch the surface

A general way to think about piecewise polynomials.

A lot of history here at Wisconsin

Algebraic and Geometric Constructions

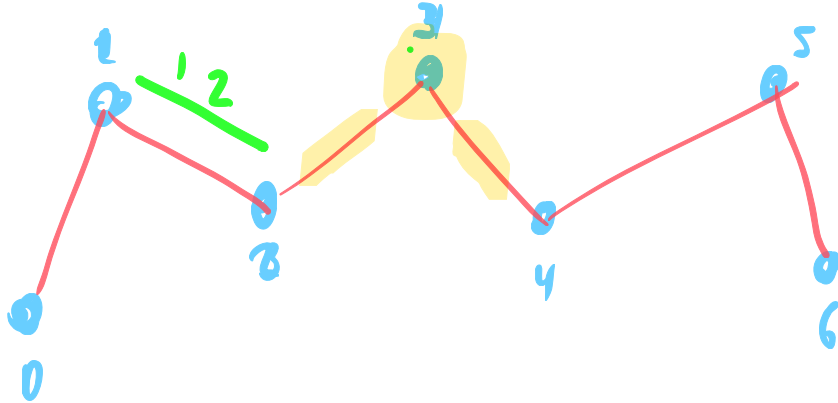
Incredibly General

Thinking in terms of chains of points...

We have points p_0, p_1, \dots, p_n

Each point is the beginning of a segment (except at end)

- ($d=1$) line segments $[p_0, p_1], [p_1, p_2], \dots [p_{n-1}, p_n]$



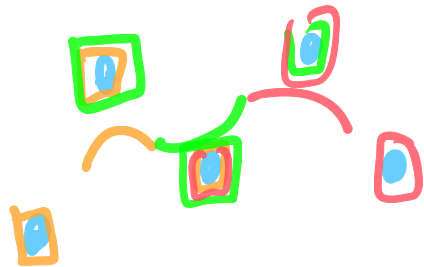
Even with higher order...

We have points $\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_n$

Each point is the beginning of a segment (except at end)

- (d=1) line segments $[\mathbf{p}_0, \mathbf{p}_1], [\mathbf{p}_1, \mathbf{p}_2], \dots, [\mathbf{p}_{n-1}, \mathbf{p}_n]$
- (d=2) quadratics $[\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2], [\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3], \dots, [\mathbf{p}_{n-2}, \mathbf{p}_{n-1}, \mathbf{p}_n]$

Each segment is $[\mathbf{p}_i, \mathbf{p}_{i+1}, \dots, \mathbf{p}_{i+d}]$



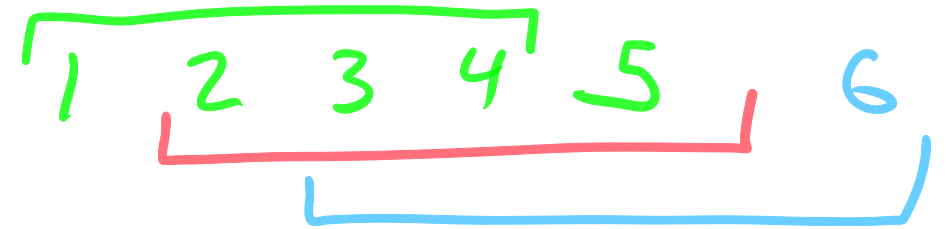
The Simplest B-Splines

- Uniform
- Low order Polynomial (d=2 or 3 in graphics)
- Chains of points are chains of polynomials

number of points

Cubic B-Splines

- 4 points influence a curve segment
- neighboring segments share 3 points
- each segment is a cubic polynomial



[demo]

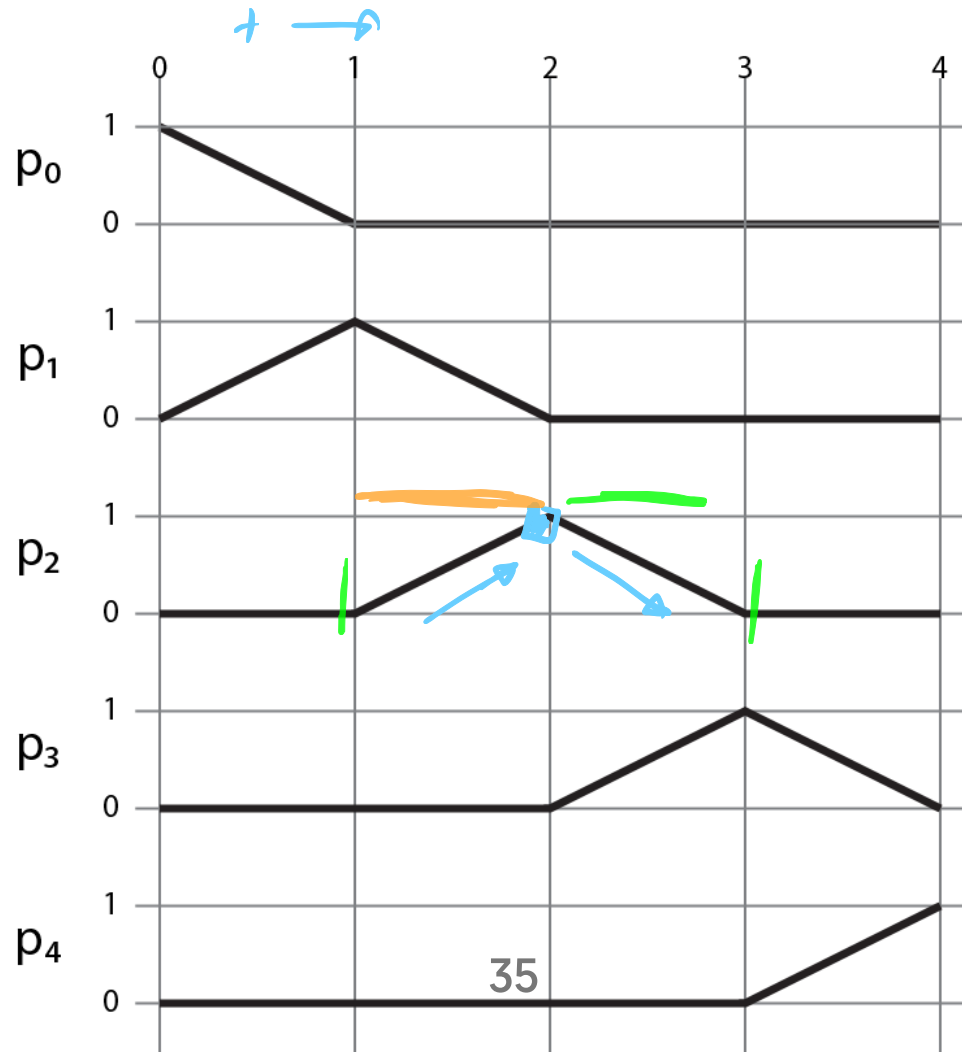
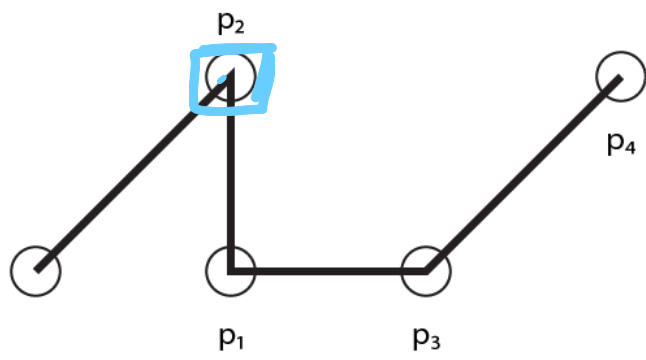
Why B-Splines?

- Curves of any degree d
- Locality: each control point influences $d + 1$ segments
- Continuity: curve is $C(d - 1)$
- Stays with the Convex Hull
- Symmetric
- Affine Invariant
- Variation Diminishing
- not interpolating (except $d=1$)

Thinking in terms of Basis Functions

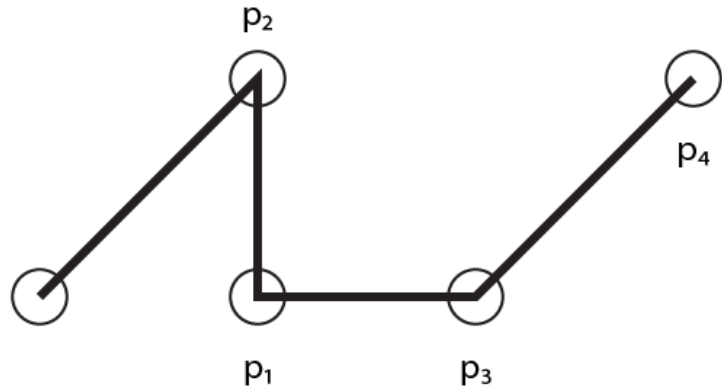
Consider connecting line segments

$$\mathbf{f}(u) = \underbrace{(1 - u)}_{b_1} \mathbf{p}_i + \underbrace{u}_{b_2} \mathbf{p}_{i+1}$$



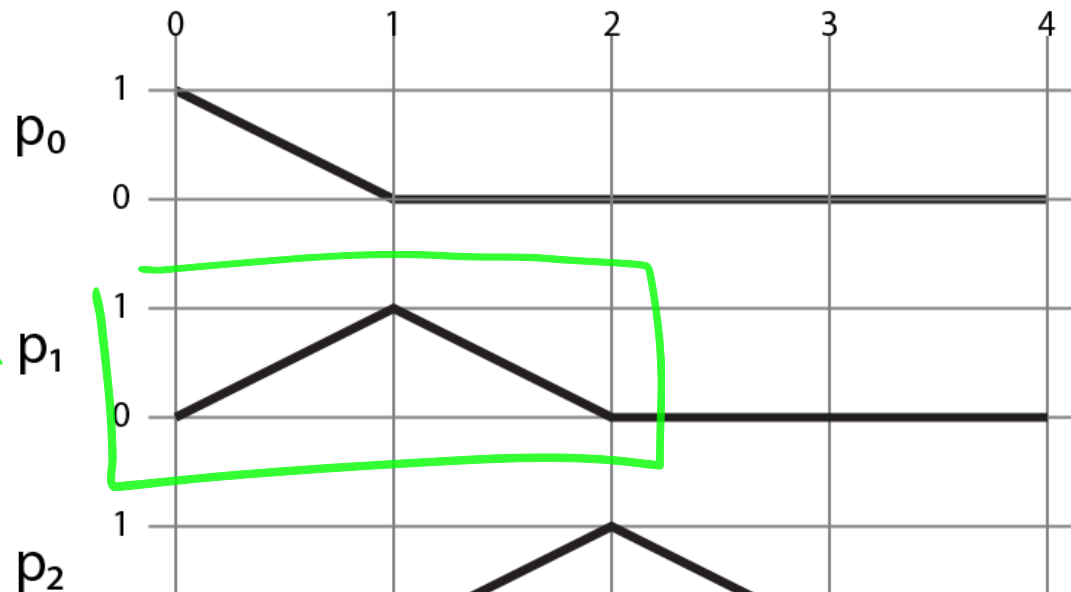
35

What about these Basis Functions?

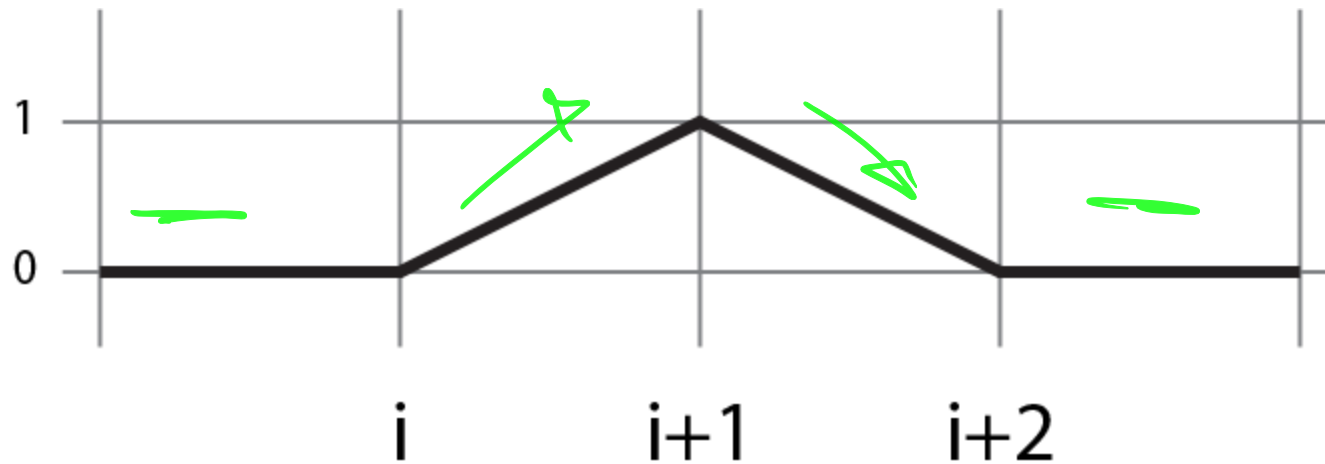


- They are piecewise polynomials
- They have the same continuity
- They repeat (except ends)

- If you have one, you'd have them all



The Linear Basis Functions



$$b_i(t) = \begin{cases} \text{if } t < i \text{ then } 0 \\ \text{elif } t < i + 1 \text{ then } t - i \\ \text{elif } t < i + 2 \text{ then } i + 2 - t \\ \text{else } 0 \end{cases}$$

On a chain of points? (lines)

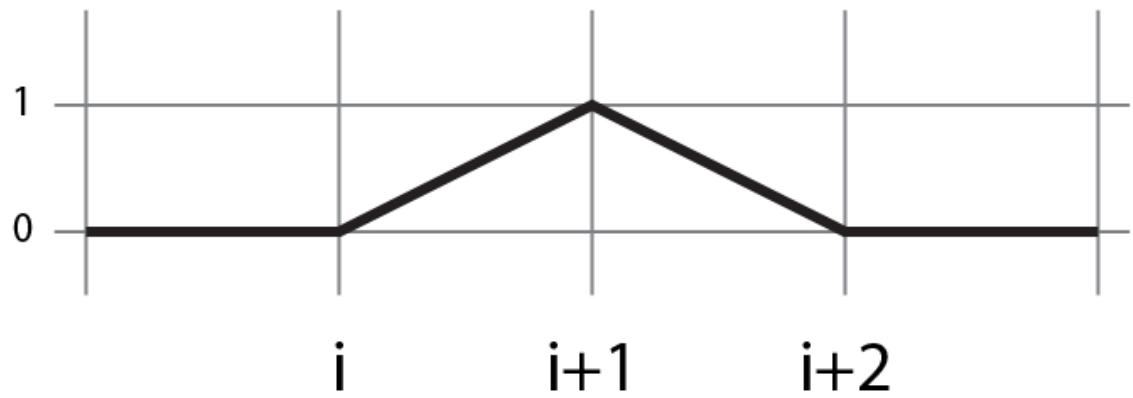
At any parameter value, 2 control points are active

One in each "phase"

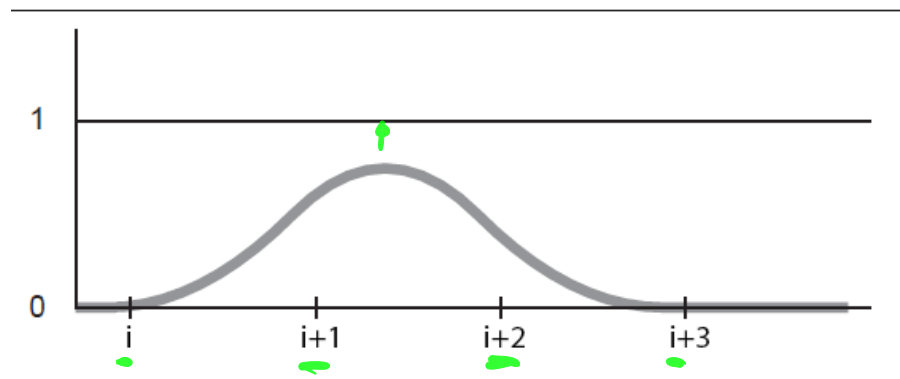
Higher-degree basis functions

Only linear interpolates (like Beziers)

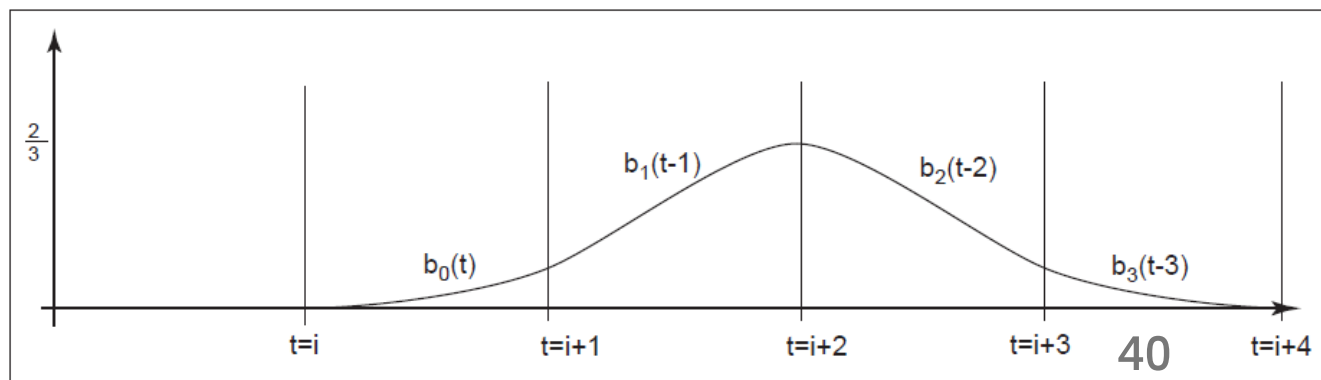
Degree d has $d + 1$ segments, meet with $C(d - 1)$ continuity



Linear



Quadratic



Cubic

Geometric Intuition for B-Splines

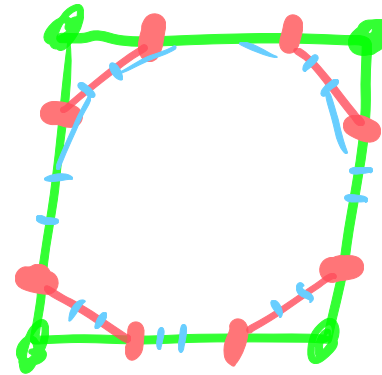
or

A More Interesting Subdivision Curve

A Geometric Approach: Chakin Corner Cutting

Subdivide each line segment:

- Mark $1/4$ and $3/4$
- Cut it there



Each iteration gets smoother

- In the limit it will be smooth

Notice:

- does not interpolate its "control points"

What to learn from that?

- Practical? (might need a lot of iterations)
- Subdivision: in the limit, it is a piecewise quadratic
- It converges to a B-Spline
- Approximating curve: it does not interpolate its (initial) points

In practice...

1. Know that it converges to piecewise quadratics
2. Look up the basis functions

What to do with these?

Use cubic B-Splines if you want $C(2)$ curves

Even though they do not interpolate, they "behave nicely"

Look up the blending functions if you need them

Curve Summary

- Use **parametric** curves
- Use **piecewise** representations, connect with **continuity**
- Use **polynomial** segments, usually **quadratics** or **cubics**
- Use **Hemite** or **Cardinal** interpolation (if you need interpolation)
- Use **Bézier** curves for good control (and API compatibility)
- Use **B-Splines** to get very smooth curves

Get ready for 3D!

