Lecture 11:

Even More Curves

Last Time

- Basis Functions
- Quadratic Interpolation
- Cubics and Hermite Forms
- Cardinal Splines (Catmull-Rom)
- Bézier Curves
- Bézier Properties
- Geometric Constructions

Today

Review with **The Train**

- Why cardinals?
- Why Béziers?
- Why Arc-Length?

Going beyond **The Train**

- C(2) curves
- Other Cardinals
- Subdivision Curves

The Train!

- "Optional" This Year
- Only for Advanced Points

- 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 $s = .5$)

A technical detail of G(1)

Is $s = 0$ still C(1)? (maybe demo)

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?

- equal steps in u
- equal steps in distance

(demo)

How to implement Arc Length?

Hard to do analytically

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

Fancier Cardinals: TCB curves

- tension
- 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 Béziers) can't represent some shapes

Exact circles (and other conic sections)

Rational Polynomial Curves

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

$$f(u) = \frac{\sum a_i u^i}{\sum b_i u^i}$$

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

What about Interpolation?

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

(demo)

Natural Splines (Cubics)

- Models a "draftsmans spline" (stiff piece of metal)
- C(2)
- second derivative is zero at the ends

- requires solving a linear system to compute coefficients from points
- not local

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 $\mathbf{p_0}, \mathbf{p_1}, \cdots, \mathbf{p_n}$

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

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

Even with higher order...

We have points $\mathbf{p_0}, \mathbf{p_1}, \cdots, \mathbf{p_n}$

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

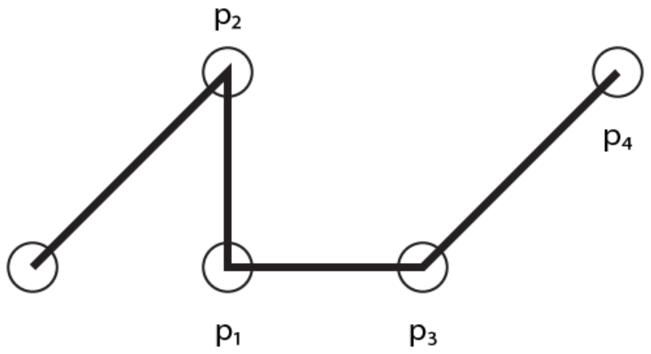
- (d=1) line segments $[\mathbf{p_0}, \mathbf{p_1}], [\mathbf{p_1}, \mathbf{p_2}], \cdots [\mathbf{p_{n-1}}, \mathbf{p_n}]$
- (d=2) quadratics $[\mathbf{p_0}, \mathbf{p_1}, \mathbf{p_2}], [\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}], \cdots [\mathbf{p_{n-2}}, \mathbf{p_{n-1}}, \mathbf{p_n}]$

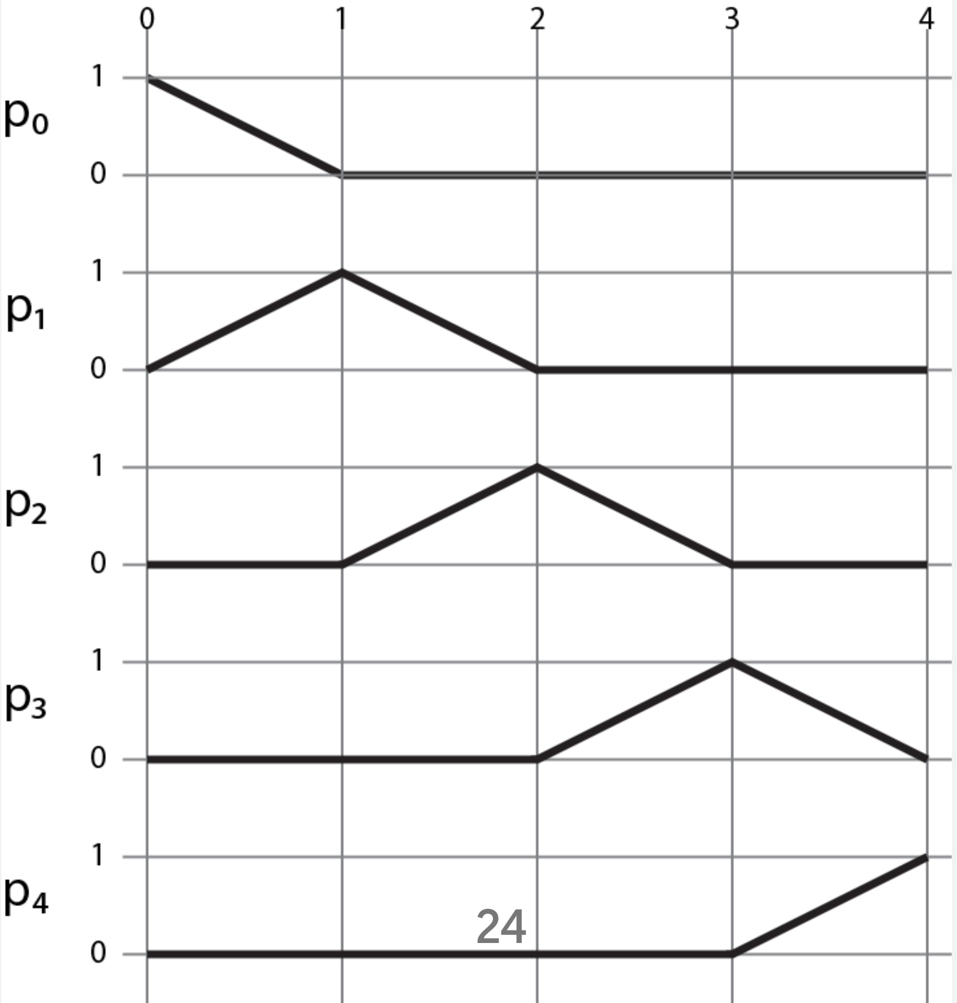
Each segment is $[\mathbf{p_i}, \mathbf{p_{i+1}}, \cdots, \mathbf{p_{i+d}}]$

Thinking in terms of Basis Functions

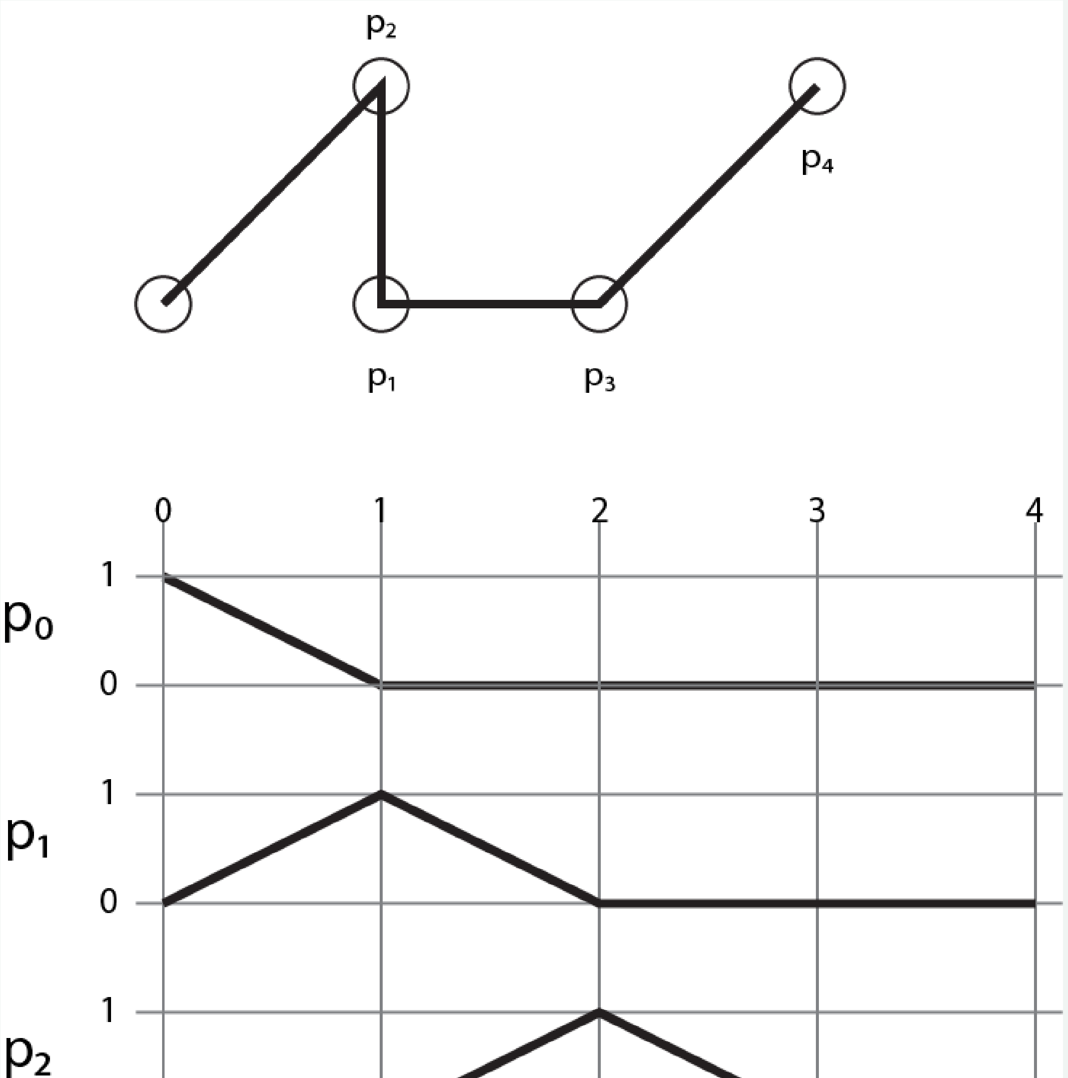
Consider connecting line segments

$$\mathbf{f}(u) = (1 - u)\,\mathbf{p_i} + u\,\mathbf{p_{i+1}}$$


p2
p4
p1
p3


0
1
2
3
4
p0
p1
p2
p3
p4
1
0

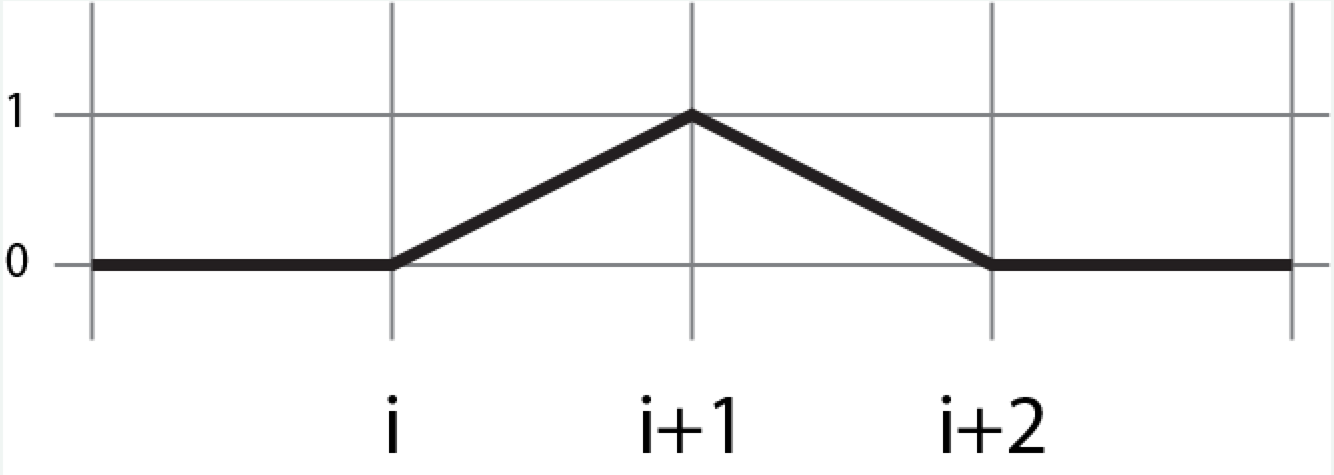
What about these Basis Functions?



- 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



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

On a chain of points? (lines)

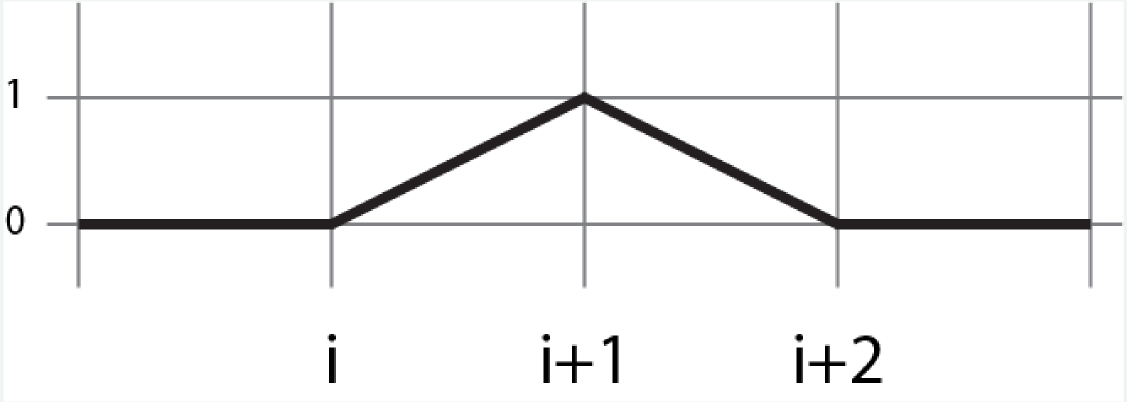
At any parameter value, 2 control points are active

One in each "phase"

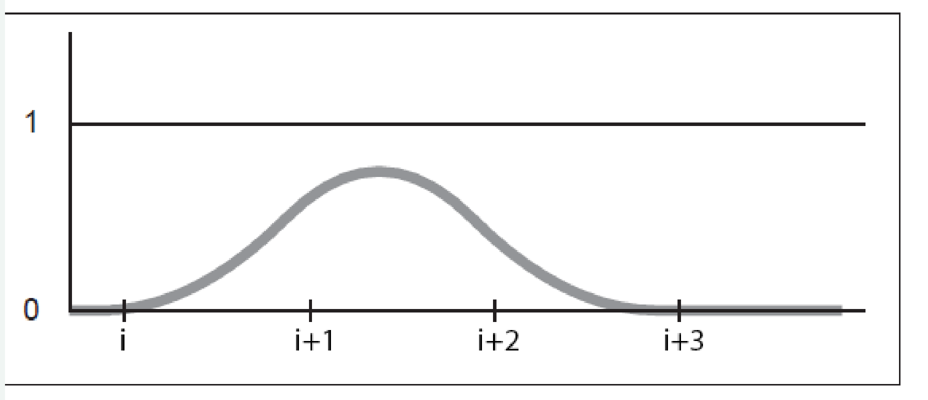
Higher-degree basis functions

Only linear interpolates (like Beziers)

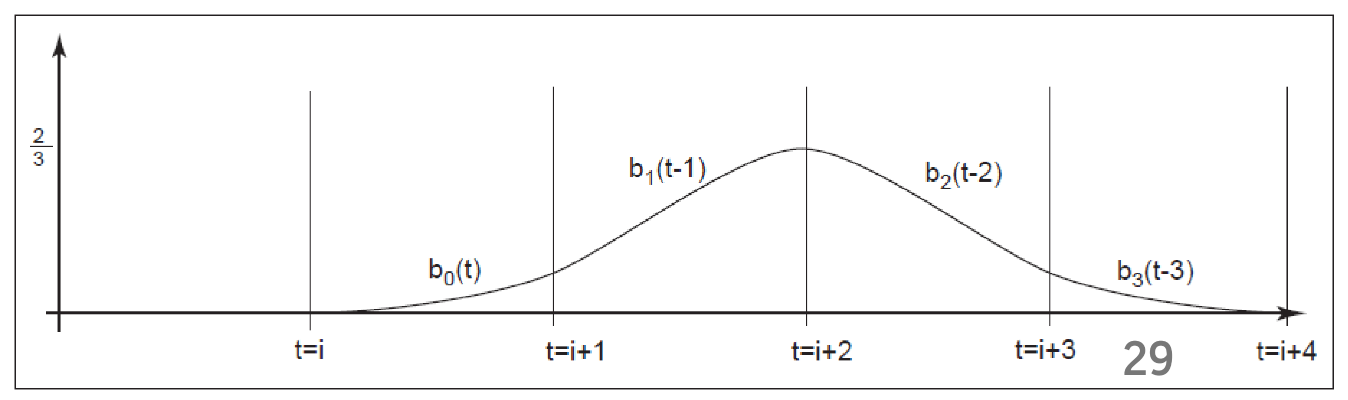
Degree d has $d + 1$ segments, meet with $C(d - 1)$ continuity



Linear



Quadratic



Cubic

The Simplest B-Splines

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

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)

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

What to do with these?

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

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

Look up the blending functions if you need them

Train Demo

See how nice B-Splines Are?

The Train Assigment

Are Curves different in 3d?

Curves are not Surfaces!

Dimensions are independent (just 3 numbers per vector)

Equations are the same

Tangent vector - normal plane (perpendicular to vector)

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!