CS4620/5620: Lecture 27

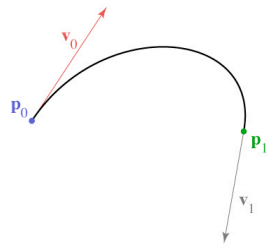
Splines

Announcements

- PPA2
  - Due a bit later (Friday)

Hermite splines

- Less trivial example
- Form of curve: piecewise cubic
- Constraints: endpoints and tangents (derivatives)



Hermite splines

- Matrix form is much simpler

$$\mathbf{p}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{v}_0 \\ \mathbf{v}_1 \end{bmatrix}$$

  - coefficients = rows
  - basis functions = columns

Matrix form of spline

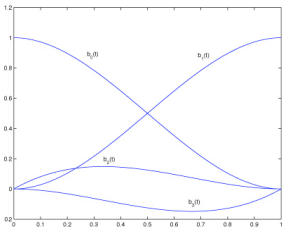
$$\mathbf{p}(t) = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d}$$

$$\begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

$$\mathbf{p}(t) = b_0(t)\mathbf{p}_0 + b_1(t)\mathbf{p}_1 + b_2(t)\mathbf{p}_2 + b_3(t)\mathbf{p}_3$$

Hermite splines

- Hermite basis functions

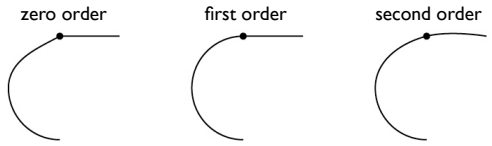


Longer Hermite splines

- Can only do so much with one Hermite spline
- Can use these splines as segments of a longer curve
  - curve from t = 0 to t = 1 defined by first segment
  - curve from t = 1 to t = 2 defined by second segment
- To avoid discontinuity, match derivatives at junctions
  - this produces a C^1 curve

Continuity

- Smoothness can be described by degree of continuity
  - zero-order (G^0): position matches from both sides
  - first-order (G^1): tangent also matches from both sides
  - second-order (G^2): curvature also matches from both sides
  - G^n vs. C^n



Continuity

$$\mathbf{p}^{(n)}(t) = \frac{d^n \mathbf{p}(t)}{dt^n}$$

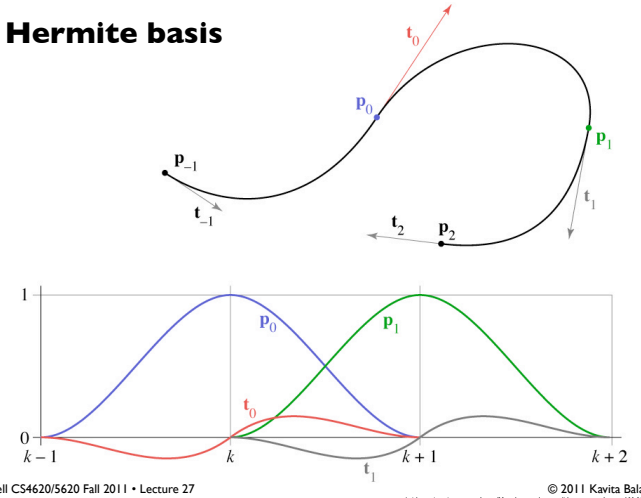
- A curve is said to be C^n continuous if **p**(t) is continuous, and all derivatives of **p**(t) up to and including degree n have the same direction and magnitude:

$$\lim_{x \to t_-} \mathbf{p}^{(m)}(x) = \lim_{x \to t_+} \mathbf{p}^{(m)}(x), \quad m = 0 \ldots n$$

- G^n continuity is like C^n but only requires the derivatives to have the same direction:

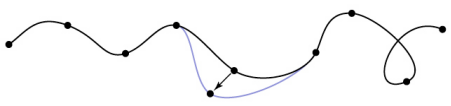
$$\lim_{x \to t_-} \mathbf{p}^{(n)}(x) = k \lim_{x \to t_+} \mathbf{p}^{(n)}(x), \quad \text{for some } k > 0$$

Hermite basis



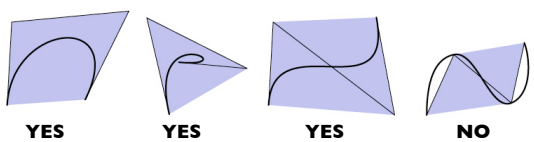
Control

- Local control
  - changing control point only affects a limited part of spline
  - without this, splines are very difficult to use



Control

- Convex hull property
  - convex hull = smallest convex region containing points
    - think of a rubber band around some pins
  - some splines stay inside convex hull of control points
    - simplifies clipping, culling, picking, etc.



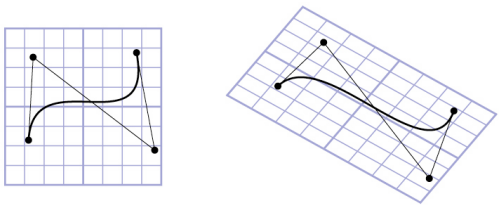
Convex hull

- If basis functions are all positive, the spline has the convex hull property
  - we're requiring them to sum to 1

  - if any basis function is ever negative, no convex hull prop.
    - proof: take the other three points at the same place

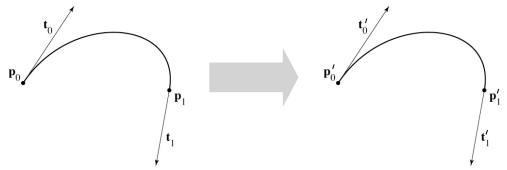
Affine invariance

- Transforming the control points is the same as transforming the curve
  - true for all commonly used splines
  - extremely convenient in practice…



Affine invariance

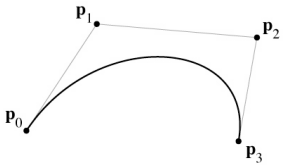
- Basis functions associated with points should always sum to 1



$$\begin{aligned}\mathbf{p}(t) &= b_0\mathbf{p}_0 + b_1\mathbf{p}_1 + b_2\mathbf{v}_0 + b_3\mathbf{v}_1 \\ \mathbf{p}'(t) &= b_0(\mathbf{p}_0 + \mathbf{u}) + b_1(\mathbf{p}_1 + \mathbf{u}) + b_2\mathbf{v}_0 + b_3\mathbf{v}_1 \\ &= b_0\mathbf{p}_0 + b_1\mathbf{p}_1 + b_2\mathbf{v}_0 + b_3\mathbf{v}_1 + (b_0 + b_1)\mathbf{u} \\ &= \mathbf{p}(t) + \mathbf{u}\end{aligned}$$

Hermite to Bézier

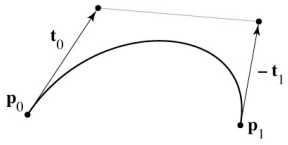
- Mixture of points and vectors is awkward
- Specify tangents as differences of points



  - note derivative is defined as 3 times offset

Hermite to Bézier

$$\begin{aligned}\mathbf{p}_0 &= \mathbf{q}_0 \\ \mathbf{p}_1 &= \mathbf{q}_3 \\ \mathbf{v}_0 &= 3(\mathbf{q}_1 - \mathbf{q}_0) \\ \mathbf{v}_1 &= 3(\mathbf{q}_3 - \mathbf{q}_2)\end{aligned}$$



$$\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\\ \mathbf{c}\\ \mathbf{d}\end{bmatrix} = \begin{bmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 3 & 0 & 0\\ 1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\mathbf{q}_0\\ \mathbf{q}_1\\ \mathbf{q}_2\\ \mathbf{q}_3\end{bmatrix}$$

Bézier matrix

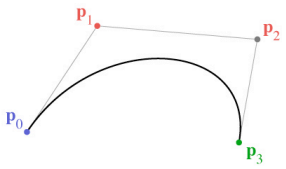
$$\mathbf{p}(t) = \begin{bmatrix}t^3 & t^2 & t & 1\end{bmatrix}\begin{bmatrix}-1 & 3 & -3 & 1\\ 3 & -6 & 3 & 0\\ -3 & 3 & 0 & 0\\ 1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\mathbf{p}_0\\ \mathbf{p}_1\\ \mathbf{p}_2\\ \mathbf{p}_3\end{bmatrix}$$

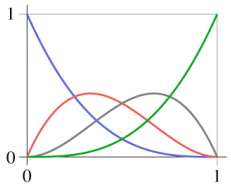
  - note that these are the Bernstein polynomials

$$C(n,k)\, t^k\, (1-t)^{n-k}$$

and that defines Bézier curves for any degree

Bézier basis



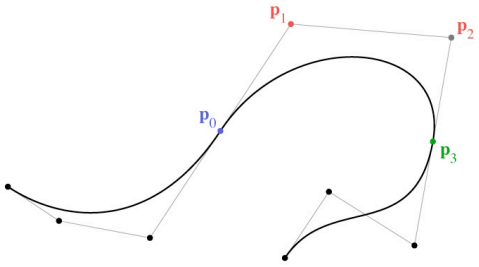


Chaining spline segments

- Hermite curves are convenient because they can be made long easily
- Bézier curves are convenient because their controls are all points and they have nice properties
  - and they interpolate every 4th point, which is a little odd
- We derived Bézier from Hermite by defining tangents from control points
  - a similar construction leads to the interpolating *Catmull-Rom* spline

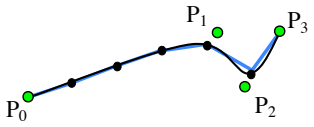
Chaining Bézier splines

- No continuity built in
- Achieve C^1 using collinear control points



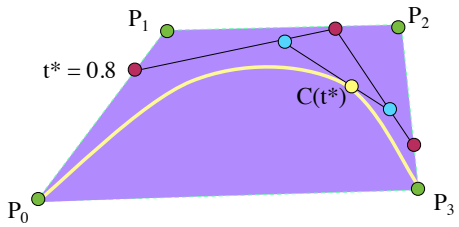
Rendering the curve

- Option 1: uniformly sample in t
- Problem
  - may oversample smooth regions: slow
  - may undersample highly curved regions: faceted rendering



Interpolation property

- C(t*) can be evaluated using interpolation
- $C(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t) P_2 + t^3 P_3$

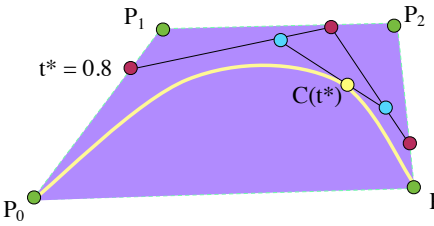


Interpolation property

$RedPt = (1-t) P_i + t P_{i+1}$

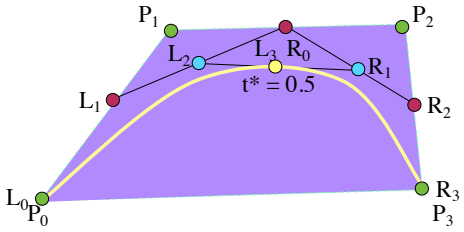
$BluePt = (1-t) RedPt_i + t RedPt_{i+1}$

$YellowPt = (1-t) BluePt_i + t BluePt_{i+1}$



De Casteljau algorithm

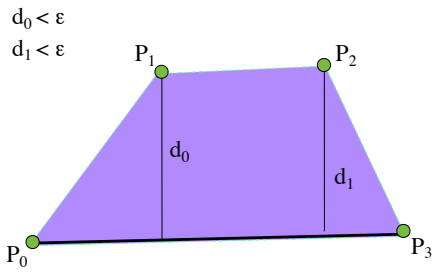
- Adaptive subdivision!



Recursive algorithm

```
void DrawRecBezier (float eps) {
  if Linear (curve, eps)
    DrawLine (curve);
  else
    SubdivideCurve (curve, leftC, rightC);
    DrawRecBezier (leftC, eps);
    DrawRecBezier (rightC, eps);
}
```


Test for Linearity



Cubic Bézier splines

- Very widely used type, especially in 2D
  - e.g. it is a primitive in PostScript/PDF
- Can represent C^1 and/or G^1 curves with corners
- Can easily add points at any position

- Disadvantage
  - Special points
  - Only C^1