1

Notes

- Assignment 0 is due today!
- To get better feel for splines, play with formulas in MATLAB!

2

Review

- **Spline**: piecewise polynomial curve
- **Knots**: endpoints of the intervals on which each polynomial is defined
- **Control Points**: knots together with information on the value of the spline (maybe derivatives too: **Hermite** splines)
- **Interpolating**: goes through control points
- **Approximating**: goes near control points
- **Smoothness**: C^n means the n^{th} derivative is continuous across the control points

3

Choices in Animation

- Piecewise linear usually not smooth enough
- For motion curves, cubic splines basically always used
- Three main choices:
  - Hermite splines: interpolating, up to C^1
  - Catmull-Rom: interpolating C^1
  - B-splines: approximating C^2

4

Cubic Hermite Splines

- Our generic cubic in an interval $[t_i, t_{i+1}]$ is
  - $q_i(t) = a_i(t-t_i)^3 + b_i(t-t_i)^2 + c_i(t-t_i) + d_i$
- Make it interpolate endpoints:
  - $q_i(t_i)=y_i$ and $q_i(t_{i+1})=y_{i+1}$
- And make it match given slopes:
  - $q_i'(t_i)=s_i$ and $q_i'(t_{i+1})=s_{i+1}$
- Work it out to get

$$a_i = \frac{-2(y_{i+1}-y_i)}{(t_{i+1}-t_i)^3} + \frac{s_i + s_{i+1}}{(t_{i+1}-t_i)^2} \qquad c_i = s_i$$

$$b_i = \frac{3(y_{i+1}-y_i)}{(t_{i+1}-t_i)^2} - \frac{2s_i + s_{i+1}}{(t_{i+1}-t_i)} \qquad d_i = y_i$$

Hermite Basis

- Rearrange the solution to get

$$y_i\left(\frac{2(t-t_i)^3}{(t_{i+1}-t_i)^3}-\frac{3(t-t_i)^2}{(t_{i+1}-t_i)^2}+1\right)+y_{i+1}\left(\frac{-2(t-t_i)^3}{(t_{i+1}-t_i)^3}+\frac{3(t-t_i)^2}{(t_{i+1}-t_i)^2}\right)$$

$$+s_i\left(\frac{(t-t_i)^3}{(t_{i+1}-t_i)^2}-\frac{2(t-t_i)^2}{(t_{i+1}-t_i)}+(t-t_i)\right)+s_{i+1}\left(\frac{(t-t_i)^3}{(t_{i+1}-t_i)^2}-\frac{(t-t_i)^2}{(t_{i+1}-t_i)}\right)$$

- That is, we're taking a linear combination of four basis functions
  - Note the functions and their slopes are either 0 or 1 at the start and end of the interval

Breaking Hermite Splines

- Usually specify one slope at each knot
- But a useful capability: use a different slope on each side of a knot
  - We break C^1 smoothness, but gain control
  - Can create motions that abruptly change, like collisions
- Aside: artists like to break things! Animation systems should have as much flexibility as possible

Catmull-Rom Splines

- This is really just a C^1 Hermite spline with an automatic choice of slopes
  - Use a 2nd order finite difference formula to estimate slope from values

$$s_i=\left(\frac{t_i-t_{i-1}}{t_{i+1}-t_{i-1}}\right)\frac{y_{i+1}-y_i}{t_{i+1}-t_i}+\left(\frac{t_{i+1}-t_i}{t_{i+1}-t_{i-1}}\right)\frac{y_i-y_{i-1}}{t_i-t_{i-1}}$$

  - For equally spaced knots, simplifies to

$$s_i=\frac{y_{i+1}-y_{i-1}}{t_{i+1}-t_{i-1}}$$

Catmull-Rom Boundaries

- Need to use slightly different formulas for the boundaries
- For example, 2nd order accurate finite difference at the start of the interval:

$$s_0=\left(\frac{t_2-t_0}{t_2-t_1}\right)\frac{y_1-y_0}{t_1-t_0}-\left(\frac{t_1-t_0}{t_2-t_1}\right)\frac{y_2-y_0}{t_2-t_0}$$

  - Symmetric formula for end of interval
- Which simplifies for equal spaced knots:

$$s_0=2\frac{y_1-y_0}{\Delta t}-\frac{y_2-y_0}{2\Delta t}$$

Aside: Evaluation

- There are two main ways to evaluate splines
  - Subdivision
  - Horner's rule
- I won't discuss subdivision (CPSC 424)
- Horner's rule: instead of directly computing $a_i(t-t_i)^3+b_i(t-t_i)^2+c_i(t-t_i)+d_i$ use the more efficient expression

  $x=t-t_i$

  $((a_i x+b_i)x+c_i)x+d_i$

B-Splines

- We'll drop the interpolating condition, and instead design a basis that is C^2 smooth
  - So control points say how much of each basis function to use, not exactly where the curve goes
- This time a basis function overlaps more than one interval
- Want to be able to interpolate constants
- We won't cover full derivation

B-Spline Basis

- Define recursively, from zero-degree polynomials up to cubic (and beyond)

$$B_{i+\frac{1}{2},0}(t)=\begin{cases}1 & t\in[t_i,t_{i+1}]\\ 0 & \text{otherwise}\end{cases}$$

$$B_{i,1}(t)=\frac{t-t_{i-1}}{t_i-t_{i-1}}B_{i-\frac{1}{2},0}(t)+\frac{t_{i+1}-t}{t_{i+1}-t_i}B_{i+\frac{1}{2},0}(t)$$

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

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

  - Note: not well defined near start and end of knot sequence - you need more knots

Looking at B-splines

- $B_{i,3}(t)$ peaks at (or near) knot t_i, but is nonzero on the interval $[t_{i-2}, t_{i+2}]$
- Always ≥ 0,
  Always < 1,
  Basis functions add up to 1 everywhere
  - Any point on the spline curve is a weighted average of nearby control points

Control

- Local control: adjusting a control point only changes curve locally
  - Far away, curve stays exactly the same
- Global control: adjusting one control point changes entire curve
  - Not as desirable - working on one part of the curve can perturb the parts you already worked out to perfection
  - But, for decent splines, effect is small---decays quickly away from adjustment

Controlling Cubics

- All three of the cubic splines we saw have local control
- But if we enforce C^2 smoothness **and** make it interpolating, we end up with global control