6B. More on C^2 Interpolating Splines

Reading

Recommended:

- Bartels, Beatty, and Barsky, *An Introduction to Splines for use in Computer Graphics and Geometric Modeling*, 1987. Chapter 3.

Motivation

The method we talked about for constructing C^2 interpolating cubic splines had an ugly asymmetry:

Q: What do you really want?

Outline

We'll build up the algorithm for creating "natural" C^2 interpolating cubic splines in stages:

1. C^1 ("Hermite") interpolation
2. C^2 formulation
3. Fast implementation
4. Dealing with closed curves

Hermite interpolation

Suppose you want to draw a C^1 interpolating cubic spline through $m+1$ control points:

An interpolating cubic spline (BBB, fig 3.1)

We get to pick derivatives at every control point.

Called "Hermite interpolation."

Derivation

We can think in terms of what happens to a particular component, say, $Y(u)$:

The Y component of the previous curve (BBB, fig 3.2)

We know two things about

$$Y_i(u) = a_i + b_i u + c_i u^2 + d_i u^3$$

namely that

$$Y_i(0) =$$

$$Y_i(1) =$$

Assigning tangents

The tangents give two more equations. Let D_i be the derivative at y_i. Then:

$$Y_i'(0) =$$

$$Y_i'(1) =$$

These four equations can be solved to yield:

$$a_i = y_i$$

$$b_i = D_i$$

$$c_i = 3(y_{i+1} - y_i) - 2D_i - D_{i+1}$$

$$d_i = 2(y_i - y_{i+1}) + D_i + D_{i+1}$$

Achieving C^2 continuity

Q: How do we leverage this result into C^2 continuity?

Q: For C^2 continuity, how many conditions are imposed with m segments?

Q: What additional conditions could we impose?

Derivation

Let's use the new constraints to solve for the D_i.

At each internal joint we want to choose D_i so that:

$$Y''_{i-1}(1) \;=\; Y''_i(0)$$

$$=$$

Substituting in and rearranging gives:

$$D_{i-1} + 4D_i + D_{i+1} \;=\; 3(y_{i+1} - y_{i-1})$$

Derivation, cont.

For natural cubic splines, we also assume that the second derivatives at the endpoints are 0:

$$Y''_0(0) \;=\; 0$$

$$=$$

Once again, substituting and rearranging gives:

$$2D_0 + D_1 = 3(y_1 - y_0)$$

Similarly, at the other endpoint:

$$D_{m-1} + 2D_m = 3(y_m - y_{m-1})$$

Derivation, cont.

We can collect all these equations in a matrix:

$$\begin{bmatrix} 2 & 1 & & & & & \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & 1 & 4 & 1 & & \\ & & & \cdot & \cdot & \cdot & \cdot \\ & & & & 1 & 4 & 1 \\ & & & & & 1 & 2 \end{bmatrix} \begin{bmatrix} D_0 \\ D_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ D_m \end{bmatrix} = \begin{bmatrix} 3(y_1 - y_0) \\ 3(y_2 - y_0) \\ \cdot \\ \cdot \\ \cdot \\ 3(y_m - y_{m-2}) \\ 3(y_m - y_{m-1}) \end{bmatrix}$$

Solved using forward elimination and back substitution.

Closed curves

Closed curves are similar, with a slightly different matrix:

$$\begin{bmatrix} 4 & 1 & & & & & 1 \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & 1 & 4 & 1 & & \\ & & & \cdot & \cdot & \cdot & \cdot \\ & & & & 1 & 4 & 1 \\ 1 & & & & & 1 & 4 \end{bmatrix} \begin{bmatrix} D_0 \\ D_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ D_m \end{bmatrix} = \begin{bmatrix} 3(y_1 - y_m) \\ 3(y_2 - y_0) \\ \cdot \\ \cdot \\ \cdot \\ 3(y_m - y_{m-2}) \\ 3(y_0 - y_{m-1}) \end{bmatrix}$$

Solved using a modified forward elimination and (the same) back substitution.

Closed curves for animation

For "animation cycles," you want the parameter values (y-values) to wrap around, while the time values (t-values) continue increasing.

So, in solving for the time coordinates, we'll plug in "phantom" t-values to fill in the top and bottom of the right-hand-side vector:

$$\begin{bmatrix} 3(t_1 - t_{-1}) \\ 3(t_2 - t_0) \\ . \\ . \\ . \\ 3(t_m - t_{m-2}) \\ 3(t_{m+1} - t_{m-1}) \end{bmatrix}$$

where

$$\begin{aligned} t_{-1} &= t_m - N \\ t_{m+1} &= t_0 + N \\ N &= \text{the number of frames in the cycle} \end{aligned}$$