**C481/581 — Computer Graphics — A. Hanson**

# Cubic Spline Algorithms

**INPUT.** A cubic spline curve is specified by subsets of four *control points* ($\vec{x}_0$, $\vec{x}_1$, $\vec{x}_2$, $\vec{x}_3$) that usually form a part of an arbitrarily long control point list. The first and last points of each subset determine properties such as the slope of the part of the curve coming from earlier or later segments. The actual drawing occurs in one section, sort of "between" $\vec{x}_1$ and $\vec{x}_2$, but which may or may not actually pass through these two points: the advantage of the Catmull-Rom spline is that it passes exactly through these two points, while other splines have more easily controlled slopes than Catmull-Rom.

**Algorithm Summary.** The fundamental algorithm is based on a tower of interpolations, whose coefficient functions $f_{ij}(t)$ are given in the tables below, and which depend on the single parameter $t$, $0 \le t \le 1$:

1. Compute $\vec{x}_{10}$ using $f_{10}(t)$ to interpolate between the control points $\vec{x}_0$ and $\vec{x}_1$; repeat to get $\vec{x}_{11}$ from $f_{11}(t)$ and $\vec{x}_1$ and $\vec{x}_2$; repeat to get $\vec{x}_{12}$ from $f_{12}(t)$ and $\vec{x}_2$ and $\vec{x}_3$.

2. Compute $\vec{x}_{20}$ using $f_{20}(t)$ to interpolate between the computed points $\vec{x}_{10}$ and $\vec{x}_{11}$; repeat to get $\vec{x}_{21}$ from $f_{21}(t)$ and $\vec{x}_{11}$ and $\vec{x}_{12}$.

3. Finish by computing the output point $\vec{x}(t)$ using $f_3(t)$ to interpolate between the computed points $\vec{x}_{20}$ and $\vec{x}_{21}$. (Note: the *same* value of $t$ is used throughout.)

**OUTPUT.** The output curve is actually many small line segments drawn from $\vec{x}(t)$ to $\vec{x}(t+\epsilon)$, with $0 \le t \le 1$. This large number of small straight segments makes it appear that we have drawn a smooth curve among the control points that is continuous and differentiable in both the value and the first derivative, and is also continuous in the second derivative.

**DATA STRUCTURES.** The functions to be used in the algorithms for different splines are the following:

| | | | |
|---|---|---|---|
| **Catmull-Rom** | $f_{10} = t+1$ | $f_{11} = t$ | $f_{12} = t-1$ |
| | $f_{20} = \frac{(t+1)}{2}$ | | $f_{21} = \frac{t}{2}$ |
| | $f_3 = t$ | | |
| **Bezier** | $f_{10} = t$ | $f_{11} = t$ | $f_{12} = t$ |
| | $f_{20} = t$ | | $f_{21} = t$ |
| | $f_3 = t$ | | |
| **Uniform B-spline** | $f_{10} = \frac{(t+2)}{3}$ | $f_{11} = \frac{(t+1)}{3}$ | $f_{12} = \frac{t}{3}$ |
| | $f_{20} = \frac{(t+1)}{2}$ | | $f_{21} = \frac{t}{2}$ |
| | $f_3 = t$ | | |

**ALGORITHM.** First define a linear interpolation function in $t$ that acts simultaneously on all components of a vector (1D, 2D, 3D, or whatever dimension) defined by a pair of points ($\vec{p}_0$, $\vec{p}_1$), and returns the components of the interpolated vector:

```
Function linear(p0, p1, t)
 begin
  pt = (1 - t)*p0 + t*p1;
  return(pt);
 end
```

Next define a series of 3 nested linear interpolations to get the interpolated (vector) curve lying "more or less" between $\vec{x}_1$ and $\vec{x}_2$ for any value of $t$:

```
Function spline(x0, x1, x2, x3, t)
 begin
 # Comment: 1st level of interpolation reduces 4 controls to 3 outputs
  p10 = linear (x0, x1, f10(t));
  p11 = linear (x1, x2, f11(t));
  p12 = linear (x2, x3, f12(t));

 # Comment: 2nd level of interpolation reduces 3 outputs to 2 outputs
  p20 = linear (p10, p11, f20(t));
  p21 = linear (p11, p12, f21(t));

 # Comment: last level returns interpolation of last two outputs
  x = linear(p20, p21, f3(t);
  return(x);
 end
```

To get different splines, simply insert different functions $f(t)$ from the data structures table above.

Finally, select an increment (something like .1, .05, or .02) for $t$ and use the `spline` procedure to return the endpoints of successive line segments which you will draw using the usual line-drawing calls for straight line segments.