

## Splines (9/26/6)

*Lecturer: James Kuffner**Scribes: Rosen Diankov*

Continuity - A curve is  $\mathcal{C}^k$  continuous if its  $i^{\text{th}}$  derivative is continuous for all  $i \leq k$ .

Cubic splines - Given  $n$  points, fits  $n-2$  local cubic curves on points  $i-1$ ,  $i$ , and  $i+1$  such that the piecewise combination is  $\mathcal{C}^1$  continuous.

Hermite splines - Given 2 endpoints and 2 tangent values, generates a curve from one endpoint to the other such that the first derivatives at the end points are equal to the tangents.

Bezier curves - Given 2 endpoints  $P_0$  and  $P_3$  and 2 tangent control points  $P_1$  and  $P_2$ , fit a curve in the convex hull of the points such that the tangents at the points  $P_0$  and  $P_3$  are equal to  $P_1 - P_0$  and  $P_3 - P_2$  respectively. The easiest way to generate them is by the DeCasteljau algorithm.

Let  $P(u) = c_0 + c_1u + c_2u^2 + c_3u^3$  be the Bezier curve such that:

$$P(0) = P_0 = c_0$$

$$P(1) = P_3 = c_0 + c_1 + c_2 + c_3$$

$$P'(0) = 3(P_1 - P_0) = c_1$$

$$P'(1) = 3(P_3 - P_2) = c_1 + 2c_2 + 3c_3$$

(1)

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = M_B \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} \quad \text{where,} \quad M_B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

$$u = \begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \end{bmatrix}$$

$$b(u) = [1 \quad u \quad u^2 \quad u^3] M_B \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} = \begin{bmatrix} (1-u)^3 \\ 3u(1-u)^2 \\ 3u^2(1-u) \\ u^3 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

Note that Bezier splines turn out to be a weighted sum of 3rd degree polynomials. These polynomials are called Bernstein basis polynomials. Their more general form of degree  $n$  is:

$$b_{k,n}(u) = \binom{n}{k} u^k (1-u)^{n-k}$$

They can be used to interpolate any  $n$  points  $\{P_i\}$  using

$$b(u) = \sum_{k=0}^n P_i b_{k,n}(u)$$

For the Bezier case,  $b_{0,3}(u) = (1-u)^3$ , etc.

A Bezier surface can easily be generated by 16 control points in 3D arranged in a simple 4x4 grid. Before a Bezier surface can be rendered, it has to be *tesselated* into primitives that the GPU understands (usually triangles or quads).

Catmull-Rom Splines

Interpolate a series of points  $\{P_i\}_{i=1}^n$  such that the final curve is  $C^1$  continuous through all points. Can create cubic curves  $p_i(u)$  through every interval  $P_i$  and  $P_{i+1}$  such that:

$$\begin{aligned} p_i(0) &= P_i \\ p_i(1) &= P_{i+1} \\ p'_i(0) &= (P_{i+1} - P_{i-1}) \tau \\ p'_i(1) &= (P_{i+2} - P_i) \tau \end{aligned}$$

where  $\tau$  is the tension factor, usually  $\tau = 0.5$ . Note that  $p_i(1) = p_{i+1}(0)$  and  $p'_i(1) = p'_{i+1}(0)$ .

Given that  $p_i(u) = c_0 + c_1u + c_2u^2 + c_3u^3$ , we need to find the coefficients  $c_j$  such that the above equations hold. Can reconstruct a matrix as in the Bezier case to solve for them. Note that  $p'_i(u) = c_1 + 2c_2u + 3c_3u^2$  and evaluating  $p_i(u)$  can be represented as a dot product where:

$$p_i(u) = [1 \quad u \quad u^2 \quad u^3] \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ -\tau & 0 & \tau & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -\tau & 0 & \tau \end{bmatrix} \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix} = \begin{bmatrix} p_i(0) \\ p'_i(0) \\ p_i(1) \\ p'_i(1) \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_0 + c_1 + c_2 + c_3 \\ c_1 + 2c_2 + 3c_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Solving for  $c_j$  and plugging into final polynomial gives:

$$p_i(u) = [1 \quad u \quad u^2 \quad u^3] \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\tau & 0 & \tau & 0 \\ 2\tau & \tau - 3 & 3 - 2\tau & -\tau \\ -\tau & 2 - \tau & \tau - 2 & \tau \end{bmatrix} \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix}$$

B-splines are more general than Bezier curves. Each control point  $P_i$  has its own basis function  $N_{i,k}$  of order  $k$  such that

$$p(t) = \sum_{i=1}^{n+1} N_{i,k}(t) P_i$$

Interpolation points  $P_i$  are called knot points. The bigger the multiplicity of the point, the greater weight it has. Each control point  $P_i$  is placed at  $t_i$ . Uniform B-splines are defined when the interval between all adjacent control points is the same. In this case,  $N_{i,k}$  is given by:

$$N_{i,1}(t) = \begin{cases} 1 & t_i \leq t < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

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

A knot sequence =  $[P_0 P_0 P_0 P_0 P_1 P_1 P_1 P_1 P_2 P_2 P_2 P_2 P_3 P_3 P_3 P_3]$  has a multiplicity of 4 at each point. If we're approximating this sequence with a 4th degree polynomial, the curve will only be  $\mathcal{C}^0$  continuous.

NURBS - Non-uniform Rational B-Splines

The curve is defined as a rational function  $f(u) = \frac{p(u)}{q(u)}$  where  $p$  and  $q$  are B-splines.