Exercise Computer graphics – (till November 6, 2007)

Bezier curves

Exercise 15: Custom-made splines

A designer wants N-1 curves to interpolate N knots. He does not care for the derivation at the end-points. N is not known in advance.

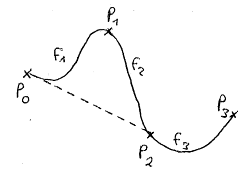
(a) Invent a smooth and easy to calculate spline which satisfies these constraints. Keep the degree as low as possible.

(b) Implement your solution by altering one of your sample applications.

Exercise Computer graphics – (till November 6, 2007)

Bezier curves

Exercise 15: Custom-made splines



P_n = knot # n

R_n = tangent in knot # n

Designer does not care for the starting- and ending tangent. So 2nd degree polynomial are sufficient. Every f_{n+1} should be 1st order smooth (C1) with its predecessor.

f_1 has no prior curve, so use $P_2 - P_0$ as tangent as known from the Catmull-Rom splines. Thus $R_0 = P_2 - P_0$

$f_n(t) = at^2 + bt + c$

$f_n'(t) = 2at + b$

<u>constraints</u>

$f_n(0) = P_{n-1} = c$

$f_n(1) = P_n = a + b + c$

$f_n'(0) = R_{n-1} = b$

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} P_{n-1} \\ P_n \\ R_{n-1} \end{pmatrix} \Rightarrow \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} -1 & 1 & -1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} P_{n-1} \\ P_n \\ R_{n-1} \end{pmatrix}$$

$$f_n(t) = \begin{pmatrix} t^2 & t & 1 \end{pmatrix} \begin{pmatrix} -1 & 1 & -1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} P_{n-1} \\ P_n \\ R_{n-1} \end{pmatrix} = \begin{pmatrix} t^2 & t & 1 \end{pmatrix} \begin{pmatrix} -P_{n-1} + P_n - R_{n-1} \\ R_{n-1} \\ P_{n-1} \end{pmatrix}$$

$$= t^2(-P_{n-1} + P_n - R_{n-1}) + t(R_{n-1}) + P_{n-1}$$

$f_n'(t) = 2t(-P_{n-1} + P_n - R_{n-1}) + R_{n-1}$

$f_n'(1) = R_1$ (which is used for the calculation of f_{n+1} and so on)

Exercise Computer graphics – (till November 6, 2007)

Bezier curves

Exercise 16: Weights for the Bezier Blending function

In the lecture we have expressed the Bezier curve analytically. We started with the degenerated instance of the curve consisting of two points only and extended it to a bent curve using three points.

Both, the straight and the bent curve consisted of weights for each knot.

a) Extend the approach from three to four knots and calculate the weights for each of the four knots.

Solution: (the weights are underlined on first occurrence)

Blending a single knot only yields a single knot.

$$P_0^{(0)} = knot\,0$$
$$P_1^{(0)} = knot\,1$$
$$P_2^{(0)} = knot\,2$$
$$P_3^{(0)} = knot\,3$$

Blending two knots yields a line. Four knots define three consecutive lines:

$$P_0^{(1)} = (\underline{1-s})P_0^{(0)} + \underline{s}P_1^{(0)}$$
$$P_1^{(1)} = (1-s)P_1^{(0)} + sP_2^{(0)}$$
$$P_2^{(1)} = (1-s)P_2^{(0)} + sP_3^{(0)}$$

Exercise Computer graphics

Bezier curves

Exercise 16: Weights for the Bezier Blending function

Solution:

Blending two lines yields a (bent) curve. Three lines define two curves:

$$P_0^{(2)} = (1-s)\,P_0^{(1)} + sP_1^{(1)}$$

$$P_0^{(2)} = (1-s)\left((1-s)\,P_0^{(0)} + sP_1^{(0)}\right) + s\left((1-s)\,P_1^{(0)} + sP_2^{(0)}\right)$$

$$P_0^{(2)} = \underline{(1-s)}^2\,P_0^{(0)} + 2\,\underline{(1-s)\,s}P_1^{(0)} + \underline{s}^2\,P_2^{(0)}$$

$$P_1^{(2)} = (1-s)\,P_1^{(1)} + sP_2^{(1)}$$

$$P_1^{(2)} = (1-s)\left((1-s)\,P_1^{(0)} + sP_2^{(0)}\right) + s\left((1-s)\,P_2^{(0)} + sP_3^{(0)}\right)$$

$$P_1^{(2)} = (1-s)^2\,P_1^{(0)} + 2\,(1-s)\,sP_2^{(0)} + s^2\,P_3^{(0)}$$

Blending two curves yields another (higher order) curve. Two curves define the single final Bezier Curve consisting of four knots.

$$P_0^{(3)} = (1-s)\,P_0^{(2)} + sP_1^{(2)}$$

$$P_0^{(3)} = (1-s)\left((1-s)^2\,P_0^{(0)} + 2(1-s)\,sP_1^{(0)} + s^2\,P_2^{(0)}\right) + s\left((1-s)^2\,P_1^{(0)} + 2\,(1-s)\,sP_2^{(0)} + s^2\,P_3^{(0)}\right)$$

$$P_0^{(3)} = (1-s)^3\,P_0^{(0)} + 2\,(1-s)^2\,sP_1^{(0)} + (1-s)\,s^2\,P_2^{(0)} + s(1-s)^2\,P_1^{(0)} + 2\,(1-s)\,s^2\,P_2^{(0)} + s^3\,P_3^{(0)}$$

Exercise Computer graphics

Bezier curves

Exercise 16: Weights for the Bezier Blending function

Solution:

Blending two curves yields another (higher order) curve. Two curves define the single final Bezier Curve consisting of four knots.

$$P_0^{(3)} = (1-s)P_0^{(2)} + sP_1^{(2)}$$

$$P_0^{(3)} = (1-s)\left((1-s)^2 P_0^{(0)} + 2(1-s)sP_1^{(0)} + s^2 P_2^{(0)}\right) + s\left((1-s)^2 P_1^{(0)} + 2(1-s)sP_2^{(0)} + s^2 P_3^{(0)}\right)$$

$$P_0^{(3)} = \underline{(1-s)^3} P_0^{(0)} + \underline{3(1-s)^2 s} P_1^{(0)} + \underline{3(1-s)s^2} P_2^{(0)} + \underline{s^3} P_3^{(0)}$$

b) When going from 2, to 3 and finally to 4 knots, can you find a pattern or schema for the weights? Express the weight for knot n in a curve consisting of N knots.

Hint: The factor for each weight is the binomial coefficient.

General formulation of a weight:

$$w_n = \binom{N}{n}(1-s)^{N-n}s^n = \frac{N!}{n!(N-n)!}(1-s)^{N-n}s^n$$

N = number of knots-1

It is also known as the Bernstein blending function.