12/03/2020

Curves and Surfaces

36-5 (plus / minus some slides): obtaining the set of 4 equations in 4 unknows for interpolation. Matrix, inverse, etc.

36-14 the same for Hermite where the conditions include interpolating p(0), p(1) and the definition of p'(0) and p'(1).

Bezier Curves
Motivation is the fact that it is not always possible to accurately define p'(0) and p'(1).

Given f(x), what is the mathematical definition of f'(x).

Limit df/dx as dx approaches 0 of or limit dx approaches 0 of (f(x+dx) – f(x))/dx f(x) is an analog (continuous) function. But, the raster is not analog.

Assume just one line on the screen

For x = j dx is 1. ➔ approximate derivative as (f(j+1) – f(j))

p(u) is continuous in u.

In Bezier, the programmer is providing P0, P1, P2, P3
The curve interpolates P0, and P3.
P1, P2 are placed by the programmer at 1/3 2/3 in a way that enable approximating the derivative P'(0), p'(1).

Delta u (du) between P0 and P1 is 1/3
Delta p (dp) the increment in p(u) from p0 to p1 is approximated by P1-p0

~P'(0) ~p'(1) in slide 36-4
~Rate of change / ~slope In Wiki you can see the device called spline which is used to generate curves with a metal rod.

Equations (37-4)?

Equation 1 and 4 are the same as in interpolation and the same as Hermite Interpolating P0, P3

Equation 2 at u=0 (approximating the derivative on the left hand side using the

derivative of p(u) on the RHS
(p2-p1)/(1/3) = $(c_{0x} *u^0 + c_{1x}*u^1 + c_{2x}*u^{2} + c_{3x}*u^3)$

Equation 3 at u=1
(p2-p1)/(1/3) = $(c_{1x} + 2*c_{2x} + 3*c_{3x})$

Slide 37-5 equations
37-6 matrix (simplifying my two equations above)

Skip blending for all curves.

Bezier polynomials are a special case of the Bernstein Polynomials
For the glMap() function, OGL is efficiently using 37-8 (for the cubic case)

37-9
Convex Hull of P1, P2, P3, P4 includes the entire Bezier curve .

In B-spline we take a large set of control points

e.g.,
p0, P1, p2, p3, p4, p5, p6, p7, p8
and use every 4 with overlap of 2 to generate the Bezier curve

p0, p1, p2, p3 ➔ use the Bezier part to generate the curve from p1 to p2
p1, p2, p3, p4 ➔ Bezier curve from p2 to p3
p2, p3, p4, p5 ➔ Bezier curve from p2 to p3
p=[p_{i-2} p_{i-1} p_i p_{i+1}] ➔ Bezier curve from p_{i-1} p_i

37-12, 13 matrix

In 37 36 we discussed the glMap() part
Slides 38 discuss the way to solve the polynomial for given set of vertices given by glEval (evaluate)

One option is using DDA

The second is using the **deCasteljau Recursion**
38-8, 38-11 theory

38-12 final product using shift and add
Self-study to the point where you can use 38-12

To further explain:
We have a Polygon P0, P1, P2, P3, describing the entire Bezier curve
We generate two Beziers I and R one for each half.
Relations between P, I, R are given in 12.
Continue and divide each sub curve to 2 for enough resolution.

38-13 and 14 ➔ how to get other curves from Bezier.
Use P1’ P2’ to get a desired curve from Bezier