Param. křivka

Q(t) = [x(t),y(t),z(t)] ~ **q**(t) = (...); teč. vekt. **q**'(t) = dq(t)/dt; tečna P(m) = $Q(t_0)$ + m**q**'(t_0)

param. spoj. C^n 0 – dráha, 1 – směr a rychlost, 2 – zrychlení; geom. G^n podmínka - $\mathbf{q}^{1\prime}(1) = k * \mathbf{q}^{2\prime}(0)$, 1 – jen směr

Param. kubika

$$\begin{aligned} x(t) &= a_x t^3 + b_x t^2 + c_x t + d_x, \\ y(t) &= a_y t^3 + b_y t^2 + c_y t + d_y, \\ z(t) &= a_z t^3 + b_z t^2 + c_z t + d_z, \end{aligned} \qquad Q(t) = \mathbf{TC} = [t^3\ t^2\ t\ 1] \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix}$$

$$Q(t) = \mathbf{TMG} = [t^3\ t^2\ t\ 1] \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix}$$

M – matice konstant, bázová m.; G – vektor geom. podmínek

sledujeme - invariance k lineárním transformacím (nez. na pořadí) konvexnost obálky, lokalita změn, křivka (ne)prochází počátečním a koncovým bodem řídícího polygonu.

Fergusonova (Hermitova) kubika

$$Q(t) = [t^3\ t^2\ t\ 1] \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ \vec{p_0} \\ \vec{p_1} \end{bmatrix}$$

$$Q(t) = P_0 F_1(t) + P_1 F_2(t) + \vec{p_0} F_3(t) + \vec{p_1} F_4(t),$$

$$\begin{aligned} F_1(t) &= 2t^3 - 3t^2 + 1, \\ F_2(t) &= -2t^3 + 3t^2, \\ F_3(t) &= t^3 - 2t^2 + t, \\ F_4(t) &= t^3 - t^2. \end{aligned}$$

Bezierovy křivky

bk n-stupně urč. n+1 body tvoř. řídící polygon

B_i - Bernsteinovy polynomy

$$Q(t) = \sum_{i=0}^{n} P_i B_i^n(t), \qquad B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}.$$

$$\vec{q_0} = n(P_1 - P_0), \qquad \vec{q_1} = n(P_n - P_{n-1})$$

Beziérova kubika

$$Q(t) = \sum_{i=0}^{3} P_i B_i^3(t)$$

$$\begin{aligned} B_0^3(t) &= (1-t)^3, \\ B_1^3(t) &= 3t(1-t)^2, \\ B_2^3(t) &= 3t^2(1-t), \\ B_3^3(t) &= t^3. \end{aligned}$$

$$Q(t) = [t^3\ t^2\ t\ 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

$$\vec{q'}(0) = 3(P_1 - P_0), \qquad \vec{q'}(1) = 3(P_3 - P_2).$$

Casteljau

Beziér – půlení stran

Coonsova kubika

TODO

Coonsův kubický B–spline

TODO

Cohen-Sutherland

přímky vůči obdélníku, lze použít pro 3D. 4 bity UBRL, případy, koncové body: 0 – uvnitř, stejný jeden bit – mimo, jinak prochází hranou. Nutnost float, mult, div.

X_{min}, X_{max}, Y_{min}, Y_{max} – hranice oblasti; zjištění kódu oblasti koncových bodů, jsou-li 0 oba, přeskočit ořezávání

cyklus; jestliže (k1 & k2) != 0, pak úsečka je mimo

výběr bodu, který není uvnitř, X, Y, kód bodu; test na jedničky v kódu:

(k & 1) != 0 -> y = ((y2–y1) / (x2–x1)) * (Xmin–x) + y; x = Xmin; (k & 2) != 0 -> y = ((y2–y1) / (x2–x1)) * (Xmax–x) + y; x = Xmax;

(k & 4) != 0 -> x = ((x2–x1) / (y2–y1)) * (Ymin–y) + x; y = Ymin; (k & 8) != 0 -> x = ((x2–x1) / (y2–y1)) * (Ymax–y) + x; y = Ymax;

opravení kódu pro nový bod; nerovnají-li se oba kódy 0, na začátek cyklu; vykreslit úsečku; další úsečka

Cyrus-Beck

Úsečky vůči konvex. n-úhelníku – nutno znát normály hran

A,B –koncové body úsečky; Ni – normála hrany; Yi – bod na hraně n-úhelníka; tmin = 0; tmax = 1

cyklus – projede všechny hrany

if (Ni * (B-A)) = 0 -> hrana rovnoběžná s úsečkou

if (Ni *(B-Pi)) = 0 -> úsečka na hraně, konec algoritmu, jinak pokračovat další hranou

if (Ni * (B-A)) != 0 -> spočítat průsečík

t = - (Ni*(A-Pi) / Ni * (B-A)) -> výpočet průsečíku

if (Ni *(B-A)) < 0 -> úsečka jde směrem do oblasti, tak tmin = max(t,tmin), jinak tmax = min(t,tmax)

dalsčí hrana

if (tmax >= tmin) -> vykreslit úsečku, jinak je úsečka mimo

Weiler-Atherton

Si – vrcholy n-úhelníku, Ci – vrcholy ořezávací oblasti, Ii – průsečíky; ve směru hod. 1. řádek vrcholy n-úh. a průsečíky, 2. řádek vrcholy oblasti + průsečíky; pokud má vzniknout víc n-úhelníků, tak ve třetí řádce _vstupní_ průsečíky. Vstup na průsečíku. Přechod do druhého řádku při průsečíku na stejný průsečík. Jsou li díry, pak proti směru hod.

Alg: Označit vrcholy oblasi po směru, vrcholy díry proti směru, vrcholy n-úh. po směru, díry n-úh. proti směru, označit všechny vzniklé průsečíky; Vypsat vrcholy n-ú. a průsečíky po směru (+1.), za to vrcholy díry a průsečíky proti směru; Vypsat vrcholy oblasti a průsečíky po směru (+1.), za to vrcholy díry oblasti a průsečíky proti směru. Seznam vstupních průsečíků; Začátek vstupním průsečíkem a odebrat ze seznamu, pak procházet seznamem vrcholů n-úhelníka. If (průsečík): if (!počáteční) ->pokračovat z 2. seznamu a odebrat ho ze seznamu. Vrchol jen přidat a pokračovat. Na konci pokračovat dál ze seznamu vst. vrcholů.

Phongův osvětl. model

N – normála, S – směr k bodovému zdroji, R – směr odraženého světla, V – směr ke kameře.

ambientní: Ia – intenzita světla, ka – koeficient amb. odrazu; difuzní a spekulární: kd – koef. dif. odrazu, ks – koef. spek. odr., IS – síla bodového světla, n – útlum; zrcadlový – index r, průhledný – index t

$L = Ia\ ka + kd\ IS\ (N.S)/d^2 + IS\ ks\ (\mathbf{R.V})^n/d^2 + kr\ Lr + kt\ Lt$

Phongův výpočet odrazu lesklého povrchu = spekulární viz výše

Konstantní stínování – výpočet barvy plochy podle normály

Gouraudovo stínování – výpočet barvy vrcholů z jejich normál (ty se spočtou aritm. prům. ze všech normál ve vrcholu), hrany interpolací mezi vrcholy: IA = [I1.(YS-Y2)+I2(Y1-YS)]/(Y1-Y2); IB = [I1.(YS-Y3)+I3(Y1-YS)]/(Y1-Y3), pak podle jedné osy lin. interp. celý trojúh.: IQ = [IA.(XB-XQ)+IB.(XQ-XA)]/(XB-XA)

Phongovo stínování – z normál ve vrcholech se interpolací vypočtou normály vnitřních bodů a z nich samotná barva bodu.

norm. vektory: nQ = nA+(nB-nA).t

Optická densita – log10(Imax / Imin)

Půltónování – metoda vzorů – jasný **Půlt. – M. konst. prahu** – jasný

Půlt. – Floyd-Steinberg – chyba distrib. doprava – dolů 3/8, ¼, 3/8
Půlt. – dithering (rozmývání) – vkládání chyb dle vzoru [0 2 | 3 1], větší vzory podle $^{(2n)}D = [4^{(n)}U \mid 4^{(n)}D + 2^{(n)}U \mid\mid 4^{(n)}D + 3^{(n)}U \mid 4^{(n)}D + U^{(n)}]$
Bressenham:

```
line (x1, y1, x2, y2) {
        x = x1; y = y1
        dx = x2 – x1; dy = y2 – y1;
        d = 2 *dy – dx; a = 2 dy; b = 2 (dy – dx)
        for (i = 0 to dx)
                if (d <= 0) -> d = d + a; x = x + 1; step H (rovně)
                else d = d + b; x = x + 1; y = y + 1; step D (šikmo)
```