Matematika pro real-time grafiku

© 2005-2010 Josef Pelikán, MFF UK Praha
http://cgg.mff.cuni.cz/~pepca/
pepca@cgg.mff.cuni.cz

Obsah

- homogenní souřadnice, maticové transformace
  - převod mezi souřadnými soustavami
- souřadné soustavy, projekční transformace, frustum
- perspektivně korektní interpolace
- reprezentace orientace
  - Eulerovy úhly, kvaterniony
  - interpolace orientací
- hladké aproximace a interpolace
  - spline funkce, přirozený spline, B-spline, Catmull-Rom, ...
- výpočet světelných modelů a mlhy

Geometrické transformace v 3D

- vektor 3D souřadnic [x, y, z]
- transformace násobením maticí 3×3
  - řádkový vektor se násobí zprava (DirectX)

$$[x, y, z] \cdot \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = [x', y', z']$$

  - sloupcový vektor se násobí zleva (OpenGL)
- transformační matice 3×3 mají podstatné omezení – **nelze** je použít pro **posunutí** (translaci)

Homogenní souřadnice

- vektor **homogenních souřadnic** [x, y, z, w]
- transformace násobením maticí 4×4

$$[x, y, z, w] \cdot \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = [x', y', z', w']$$

- homogenní maticí lze i **posunovat** (translace) a provádět **perspektivní projekci**

Převod homogenních souřadnic

- homogenní souřadnice [x, y, z, w] se převádějí na běžné kartézské souřadnice vydělením (je-li w≠0) **[x/w, y/w, z/w]**
- souřadnice **[x, y, z, 0]** neodpovídají žádnému vlastnímu bodu v prostoru
  - lze je chápat jako reprezentaci **směrového vektoru** (bod v nekonečnu)
- převod z obyčejných souřadnic do homogenních je jednoduchý: **[x, y, z] ... [x, y, z, 1]**

Elementární transformace

Nejběžnější jsou afinní transformace:

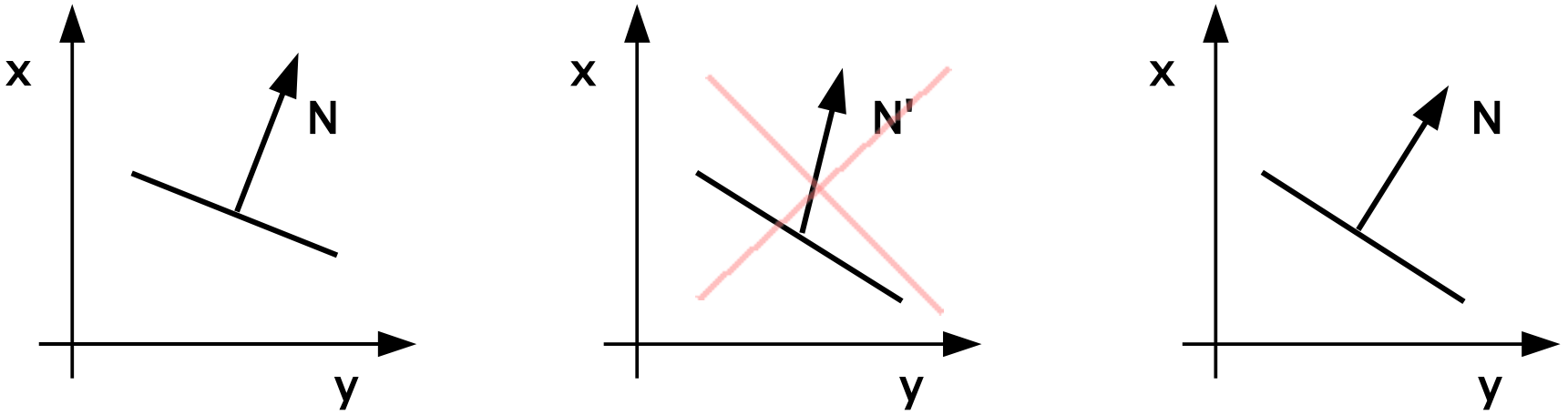
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ t_1 & t_2 & t_3 & 1 \end{bmatrix}$$

- levá horní podmatice **[a_{11} až a_{33}]** vyjadřuje rozměr a orientaci, případně zkosení
- vektor **[t_1, t_2, t_3]** udává posunutí (translaci)
  - posunutí je až poslední operace

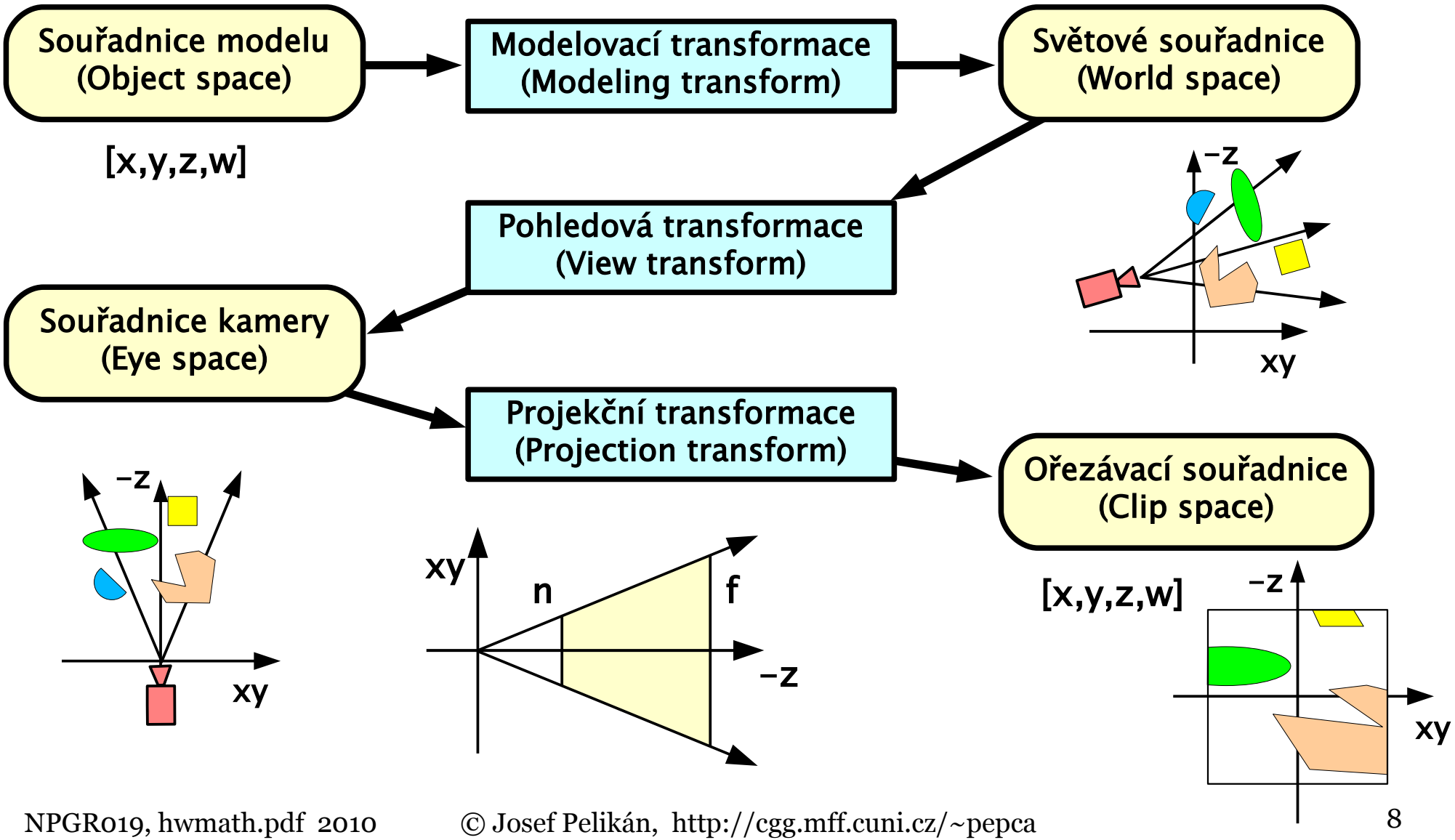
Transformace normál

- normálové vektory se nesmí transformovat běžnými transformačními maticemi
  - výjimka: matice M je rotační (ortonormální)
- matice pro **transformaci normál N**:

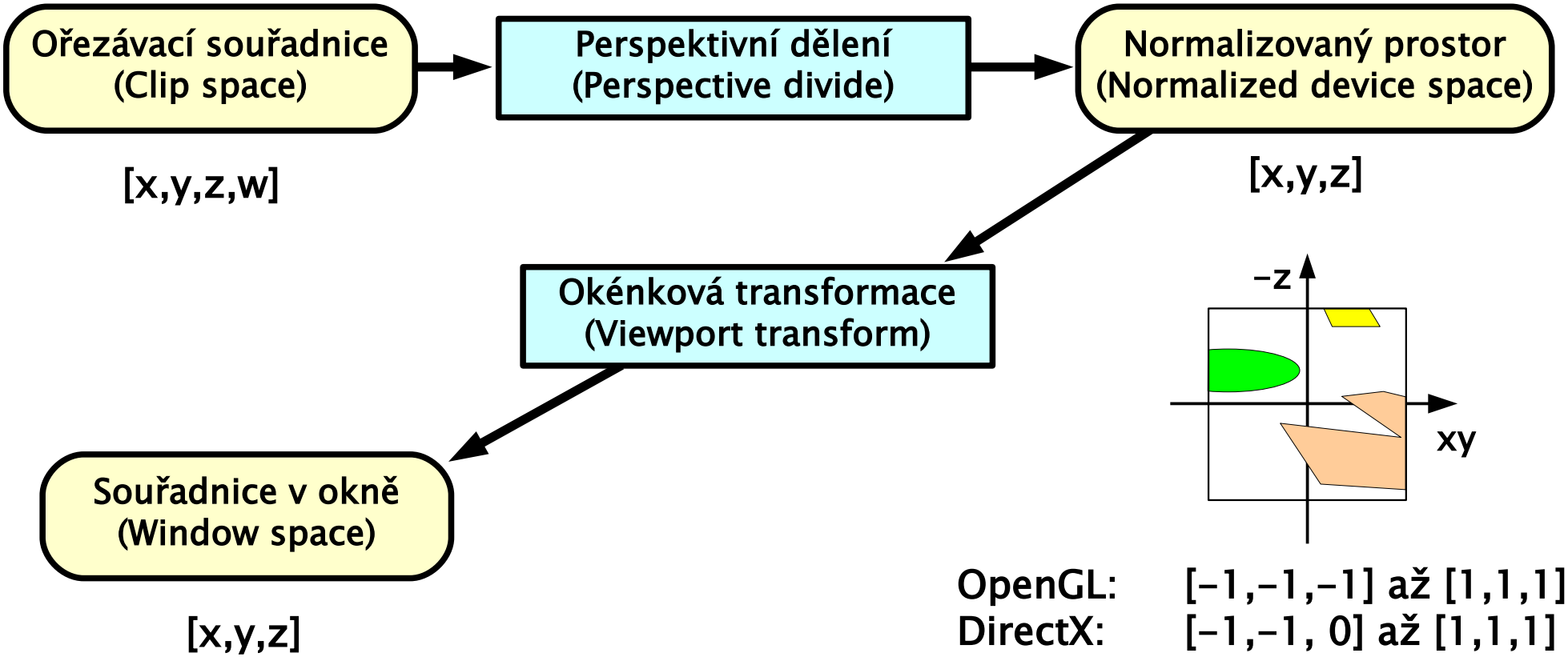
$$N = (M^{-1})^T$$



Souřadné soustavy



Souřadné soustavy



[x,y] skutečná velikost v pixelech na obrazovce (fragmenty)
z hloubka kompatibilní s z-bufferem

Souřadné soustavy

- **souřadnice modelu**
  - databáze objektů, ze kterých se skládá scéna
  - 3D modelovací programy (3DS, Maya, ..)
- **světové souřadnice**
  - absolutní souřadnice virtuálního 3D světa
  - vzájemná poloha jednotlivých instancí objektů
- **souřadnice kamery**
  - 3D svět se transformuje do relativních souřadnic kamery
  - střed projekce: **počátek**, směr pohledu: **-z** (nebo **z**)

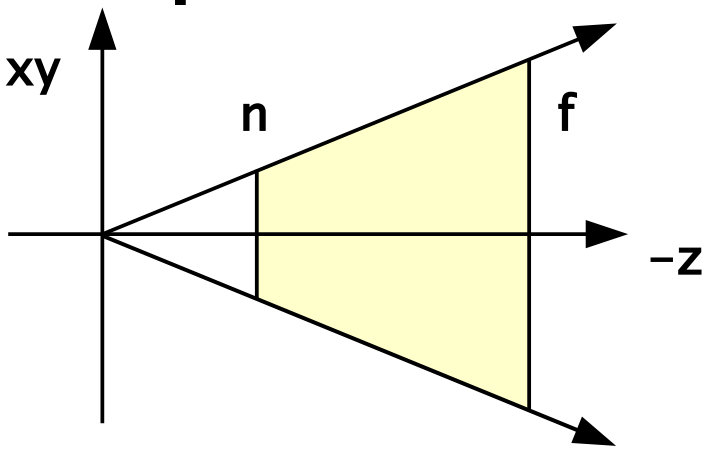
Souřadné soustavy a transformace

- **transformace model → kamera**
  - společná transformační matice "modelview"
  - světové souřadnice nejsou moc důležité
- **projekční transformace**
  - definuje zorný objem = **frustum** [**l**, **r**, **b**, **t**, **n**, **f**]
  - přední a zadní ořezávací vzdálenost: **n**, **f**
  - výsledkem je homogenní souřadnice (před ořezáním):
- **ořezávací souřadnice** ("clip space")
  - **výstupní souřadnice** vertex shaderu !

Projekční transformace

- vzdálený bod **f** lze posunout do nekonečna

$$\begin{bmatrix} \frac{2n}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2n}{t-b} & 0 & 0 \\ -\frac{r+l}{r-l} & -\frac{t+b}{t-b} & \frac{f+n}{f-n} & 1 \\ 0 & 0 & -\frac{2fn}{f-n} & 0 \end{bmatrix} \qquad \begin{bmatrix} \frac{2n}{r-l} & 0 & 0 & 0 \\ 0 & \frac{2n}{t-b} & 0 & 0 \\ -\frac{r+l}{r-l} & -\frac{t+b}{t-b} & 1 & 1 \\ 0 & 0 & -2n & 0 \end{bmatrix}$$

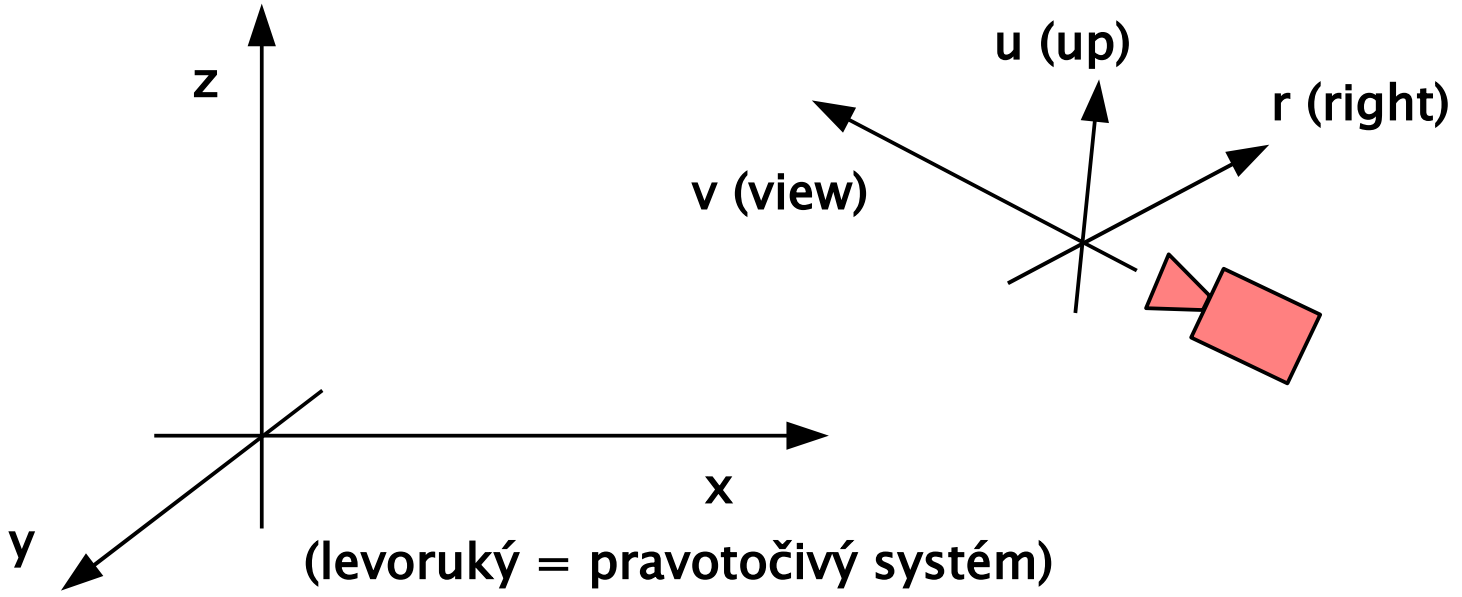


Souřadné soustavy a transformace

- **perspektivní dělení**
  - pouze převádí homogenní souřadnice do kartézských
- **normalizované souřadnice zařízení** (“**NDC**”)
  - kvádr standardní velikosti
  - OpenGL: [-1,-1,-1] až [1,1,1]
  - DirectX: [-1,-1,0] až [1,1,1]
- **souřadnice v okně** (“window space”)
  - výsledkem okénkové transf. a transformace hloubky
  - používají se při **rasterizaci** a práci s **fragmenty**

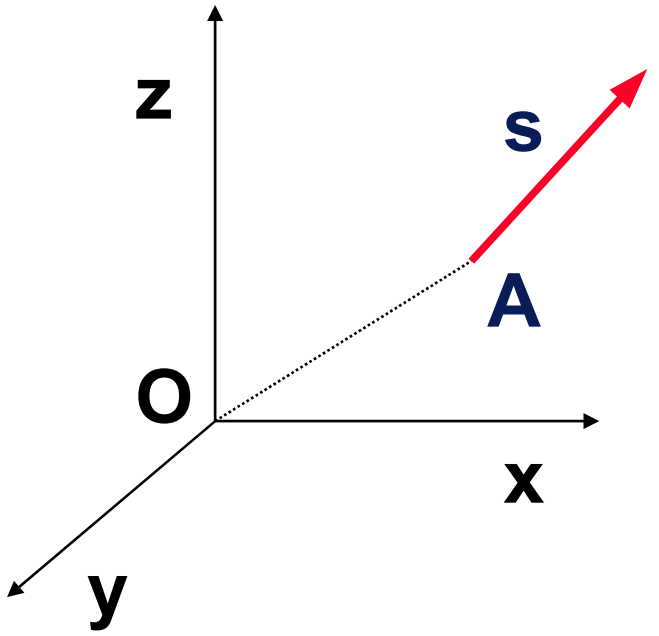
Transformace tuhého tělesa

- zachovává **tvar těles**, mění pouze jejich **orientaci**
  - skládá se jenom z posunutí a otáčení
  - často se používá k **převodu mezi souřadnicovými systémy** (např. mezi „světovými" souřadnicemi a systémem spojeným s pozorovatelem)



Přenesení polopřímky do osy *z*

- konstrukce matice postupně v několika krocích:

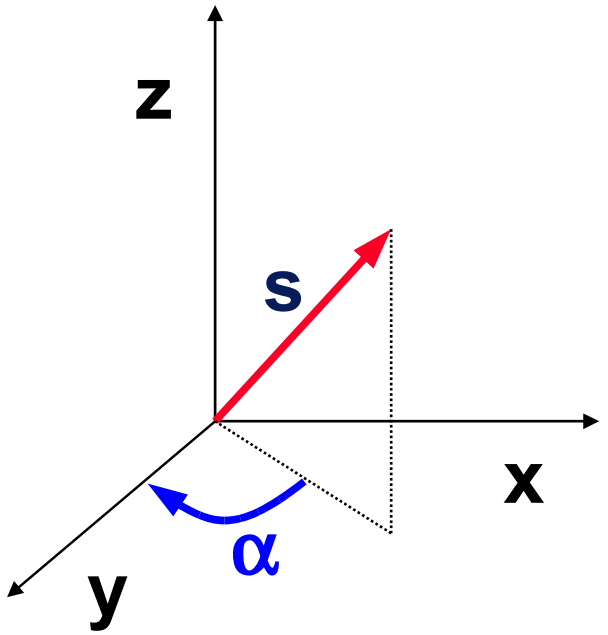


Polopřímka je zadána bodem **A** a směrovým vektorem **s**

$$\mathbf{M} = \mathbf{T}(-\mathbf{A})$$
$$\mathbf{M}^{-1} = \mathbf{T}(\mathbf{A})$$

1. krok:
přenesení bodu **A** do počátku

Přenesení polopřímky do osy *z*



$$\sin\alpha = \frac{s_x}{\sqrt{s_x^2 + s_y^2}} \qquad \cos\alpha = \frac{s_y}{\sqrt{s_x^2 + s_y^2}}$$

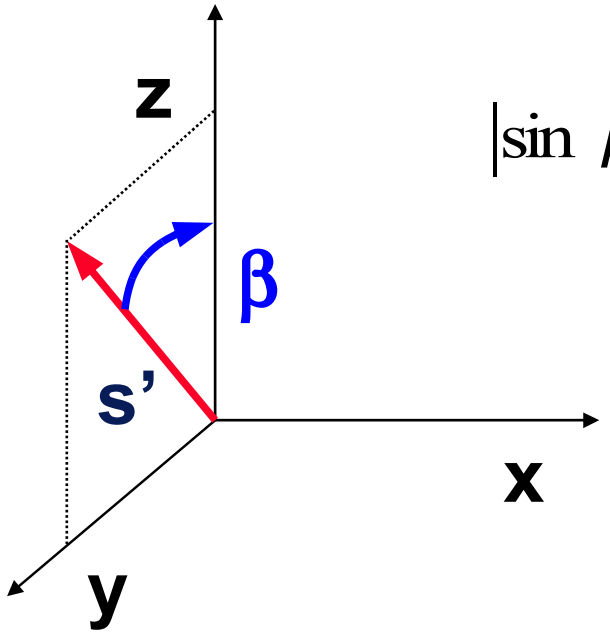
$$\mathbf{M} = \mathbf{T}(-\mathbf{A}) \cdot \mathbf{R}_z(\alpha)$$

$$\mathbf{M}^{-1} = \mathbf{R}_z(-\alpha) \cdot \mathbf{T}(\mathbf{A})$$

2. krok:

otočení polopřímky do roviny **yz** (okolo osy **z**)

Přenesení polopřímky do osy *z*



$$|\sin\beta| = \frac{\sqrt{s_x^2 + s_y^2}}{\sqrt{s_x^2 + s_y^2 + s_z^2}} \qquad \cos\beta = \frac{s_z}{\sqrt{s_x^2 + s_y^2 + s_z^2}}$$

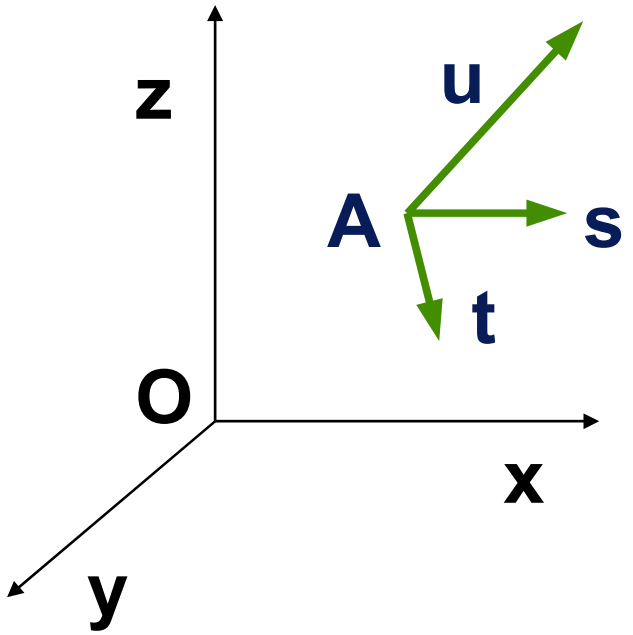
$$M = T(-A) \cdot R_z(\alpha) \cdot R_x(\beta)$$

$$M^{-1} = R_x(-\beta) \cdot R_z(-\alpha) \cdot T(A)$$

3. krok:

otočení polopřímky do osy **z** (okolo osy **x**)

Převod mezi souřadnými soustavami

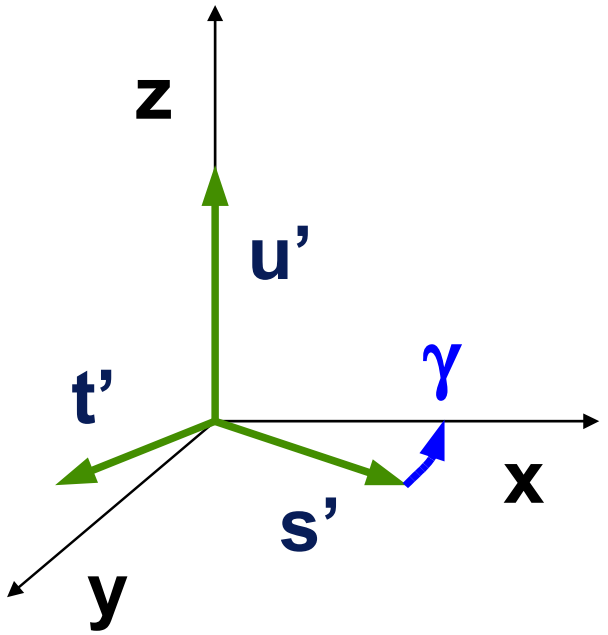


Souřadný systém je zadán svým počátkem **A** a trojicí vektorů **s, t, u**

$$Cs = M(A,u)$$
$$Cs^{-1} = M(A,u)^{-1}$$

1. krok:
přenesení polopřímky **(A,u)** do osy **z**

Převod mezi souřadnými soustavami



$$\sin\gamma = \frac{[s \cdot M(a,u)]_y}{|s \cdot M(a,u)|}$$

apod.

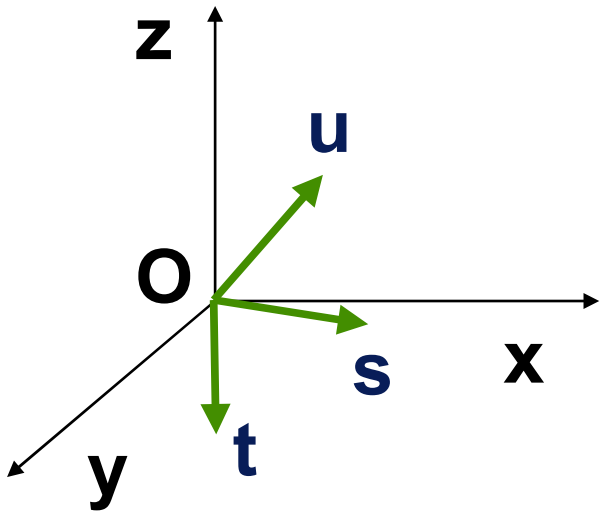
$$Cs = M(A,u) \cdot R_z(\gamma)$$

$$Cs^{-1} = R_z(-\gamma) \cdot M(A,u)^{-1}$$

2. krok:

ztotožnění os **s’→x** a **t’→y** (otočením kolem **z=u’**)

Převod mezi dvěma orientacemi II



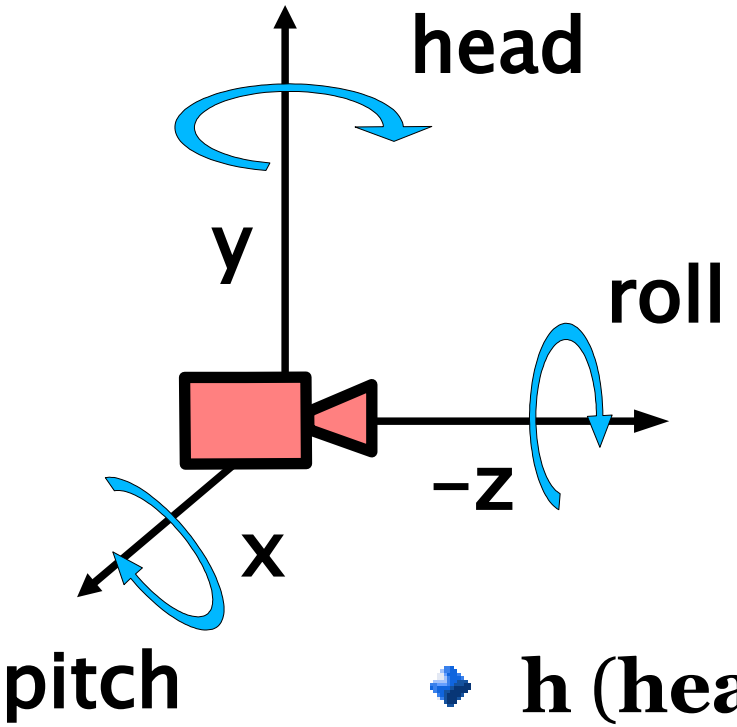
Souřadný systém má počátek v **O** a je zadán trojicí jednotkových vektorů **[s, t, u]**

$$M_{stu \to xyz} = \begin{bmatrix} s_x & s_y & s_z \\ t_x & t_y & t_z \\ u_x & u_y & u_z \end{bmatrix}$$

$$[1,0,0] \cdot M_{stu \to xyz} = s$$
$$[0,1,0] \cdot M_{stu \to xyz} = t$$
$$[0,0,1] \cdot M_{stu \to xyz} = u$$

$$M_{xyz \to stu} = M^T_{stu \to xyz}$$

Eulerova transformace



- rozklad obecné rotace na **tři složky**
  - Leonard Euler (1707-1783)

$$E(h,p,r) = R_y(h) \cdot R_x(p) \cdot R_z(r)$$

- **h** (**head**, yaw): otočení hlavy v půdorysu
- **p** (**pitch**): zvednutí/sklonění hlavy
- **r** (**roll**): otočení kolem osy pohledu

Eulerova transformace II

- výsledná matice rotace:

$$E = \begin{pmatrix} c(r)c(h)-s(r)s(p)s(h) & s(r)c(h)+c(r)s(p)s(h) & -c(p)s(h) \\ -s(r)c(p) & c(r)c(p) & s(p) \\ c(r)s(h)+s(r)s(p)c(h) & s(r)s(h)-c(r)s(p)c(h) & c(p)c(h) \end{pmatrix}$$

- s(x) .. sin(x), c(x) .. cos(x)
- možnost zpětného výpočtu úhlů h, p, r:
  - p .. e_{23}
  - r .. e_{21}/e_{22}
  - h .. e_{13}/e_{33}

Rotace – jiné konvence

- **hlavní konvence**:
  - 1. rotace kolem **z** o úhel **φ**
  - 2. rotace kolem **x'** o úhel **θ**
  - 3. rotace kolem **z''** o úhel **ψ**
- **x-konvence**:
  - 1. rotace kolem osy **z**
  - 2. rotace kolem původní osy **x**
  - 3. rotace kolem původní osy **z**
- **další systémy**: aeronautika, gyroskopy, fyzika, ..

Kvaterniony

- Sir William Rowan **Hamilton**, 1843
  - $i^2 = j^2 = k^2 = ijk = -1$
  - aplikace v grafice až v 1985 (Shoemake)
  - **zobecnění komplexních čísel** do 4D prostoru
- $\mathbf{q} = (\mathbf{v}, w) = i\,x + j\,y + k\,z + w = \mathbf{v} + w$
- **imaginární část** $\mathbf{v} = (x, y, z) = i\,x + j\,y + k\,z$
- $i^2 = j^2 = k^2 = -1, \quad jk = -kj = i, \quad ki = -ik = j, \quad ij = -ji = k$

Kvaterniony – operace

- **sčítání**
  - $(\mathbf{v}_1, w_1) + (\mathbf{v}_2, w_2) = (\mathbf{v}_1+\mathbf{v}_2, w_1+w_2)$
- **násobení**
  - $\mathbf{q}\,\mathbf{r} = (\mathbf{v}_q \times \mathbf{v}_r + w_r\mathbf{v}_q + w_q\mathbf{v}_r,\ w_q w_r - \mathbf{v}_q \cdot \mathbf{v}_r)$

$$
\begin{aligned}
&i(q_y r_z - q_z r_y + r_w q_x + q_w r_x),\\
&j(q_z r_x - q_x r_z + r_w q_y + q_w r_y),\\
&k(q_x r_y - q_y r_x + r_w q_z + q_w r_z),\\
&q_w r_w - q_x r_x - q_y r_y - q_z r_z
\end{aligned}
$$

Kvaterniony – operace II

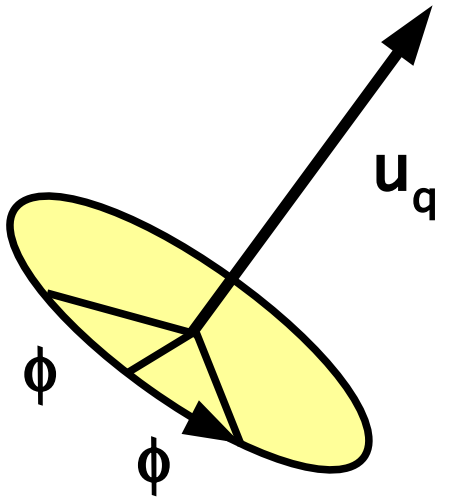
- **konjugace**
  - $(\mathbf{v}, w)^* = (-\mathbf{v}, w)$
- **norma** (čtverec absolutní hodnoty)
  - $||\mathbf{q}||^2 = n(\mathbf{q}) = \mathbf{q}\,\mathbf{q}^* = x^2 + y^2 + z^2 + w^2$
- **jednotka**
  - $\mathbf{i} = (\mathbf{o}, 1)$
- **převrácená hodnota**
  - $\mathbf{q}^{-1} = \mathbf{q}^* / n(\mathbf{q})$
- **násobení skalárem**
  - $s\,\mathbf{q} = (0, s)\,(\mathbf{v}, w) = (s\,\mathbf{v}, s\,w)$

Jednotkové kvaterniony

- **jednotkový kvaternion** lze vyjádřit goniometricky
  - $\mathbf{q} = (\mathbf{u}_q \sin\phi, \cos\phi)$
  - pro nějaký **jednotkový 3D vektor** $\mathbf{u}_q$
- reprezentace **rotace (orientace)** v 3D
  - nejednoznačnost: $\mathbf{q}$ i $\mathbf{-q}$ reprezentují stejnou rotaci!
  - identita (nulové otočení): $(\mathbf{o},1)$
- mocnina, exponenciála, logaritmus:
  - $\mathbf{q} = \mathbf{u}_q \sin\phi + \cos\phi = \exp(\phi\,\mathbf{u}_q), \quad \log\mathbf{q} = \phi\,\mathbf{u}_q$
  - $\mathbf{q}^t = (\mathbf{u}_q \sin\phi + \cos\phi)^t = \exp(t\phi\,\mathbf{u}_q) = \mathbf{u}_q \sin t\phi + \cos t\phi$

Rotace pomocí kvaternionu

- **jednotkový kvaternion**
  - $\mathbf{q} = (\mathbf{u}_q \sin\phi, \cos\phi)$
  - $\mathbf{u}_q$ ... osa rotace, ϕ ... úhel



- bod nebo vektor v 3D: $\mathbf{p} = [p_x, p_y, p_z, p_w]$
- **rotace** bodu (vektoru) $\mathbf{p}$ kolem osy $\mathbf{u}_q$ o úhel 2ϕ:
  - $\mathbf{p'} = \mathbf{q}\,\mathbf{p}\,\mathbf{q}^{-1} = \mathbf{q}\,\mathbf{p}\,\mathbf{q}^*$

Převod mezi kvaternionem a maticí

- kvaternion $\mathbf{q}$ po převedení na matici:

$$M = \begin{pmatrix} 1-2(y^2+z^2) & 2(xy+wz) & 2(xz-wy) \\ 2(xy-wz) & 1-2(x^2+z^2) & 2(yz+wx) \\ 2(xz+wy) & 2(yz-wx) & 1-2(x^2+y^2) \end{pmatrix}$$

- převod matice na kvaternion – základem jsou rovnice ($):

$$m_{23} - m_{32} = 4wx$$
$$m_{31} - m_{13} = 4wy$$
$$m_{12} - m_{21} = 4wz$$
$$tr\ M + 1 = 4w^2$$

Převod matice na kvaternion II

1. pokud je "stopa matice+1" v absolutní hodnotě **velká**:

$$w = \frac{1}{2}\sqrt{tr\,M + 1} \qquad x = \frac{m_{23} - m_{32}}{4w}$$

$$y = \frac{m_{31} - m_{13}}{4w} \qquad z = \frac{m_{12} - m_{21}}{4w}$$

2. v opačném případě se nejprve spočítá složka s maximální absolutní hodnotou, pak se použije $:

$$4x^2 = 1 + m_{11} - m_{22} - m_{33}$$

$$4y^2 = 1 - m_{11} + m_{22} - m_{33}$$

$$4z^2 = 1 - m_{11} - m_{22} + m_{33}$$

Sférická lineární interpolace (slerp)

- dva kvaterniony **q** a **r** (**q** · **r** ≥ 0, jinak **-q**)
- reálný parametr **0 ≤ t ≤ 1**

interpolovaný kvaternion slerp(q,r,t) = q (q* r)^t

- jiné vyjádření:

$$slerp(q,r,t) = \frac{\sin(\phi(1-t))}{\sin\phi}\cdot q + \frac{\sin(\phi t)}{\sin\phi}\cdot r$$

$$\cos\phi = q_x r_x + q_y r_y + q_z r_z + q_w r_w$$

Nejkratší sférický oblouk mezi orientacemi **q** a **r**

Otočení mezi dvěma vektory

- dva směrové vektory **s** a **t:**

1. normalizace vektorů **s**, **t**
2. jednotková rotační osa $\mathbf{u} = (\mathbf{s} \times \mathbf{t}) / ||\mathbf{s} \times \mathbf{t}||$
3. vyjádření úhlu mezi s a t: $\mathbf{e} = \mathbf{s} \cdot \mathbf{t} = \cos 2\phi$

   $||\mathbf{s} \times \mathbf{t}|| = \sin 2\phi$
4. výsledný kvaternion: $\mathbf{q} = (\mathbf{u} \cdot \sin \phi, \cos \phi)$

$$q=(q_v, q_w)=\left(\frac{1}{\sqrt{2(1+e)}}(s\times t), \frac{\sqrt{2(1+e)}}{2} \right)$$

Slerp rotačních matic

- dvě rotační matice **Q** a **R**
- reálný parametr $\mathbf{0 \le t \le 1}$

interpolovaná matice $\mathbf{slerp(Q,R,t) = Q\,(Q^T R)^t}$

- **technický problém:** obecná mocnina rotač. matice
- nutnost výpočtu osy a úhlu otočení $Q^T R$
  - výpočetně málo efektivní

(podrobnosti viz "RotationIssues.pdf")

Vyjádření rotace – shrnutí

- **rotační matice**

  \+ HW podpora, efektivní transformace vektoru

  – paměť (float[9]), neefektivní ostatní operace

- **osa a úhel otočení**

  \+ paměť (float[4] nebo float[6]), podobná kvaternionu

  – neefektivní kompozice, interpolace

- **kvaternion**

  \+ paměť (float[4]), kompozice, interpolace

  – neefektivní transformace vektoru

(podrobnosti viz “RotationIssues.pdf”)

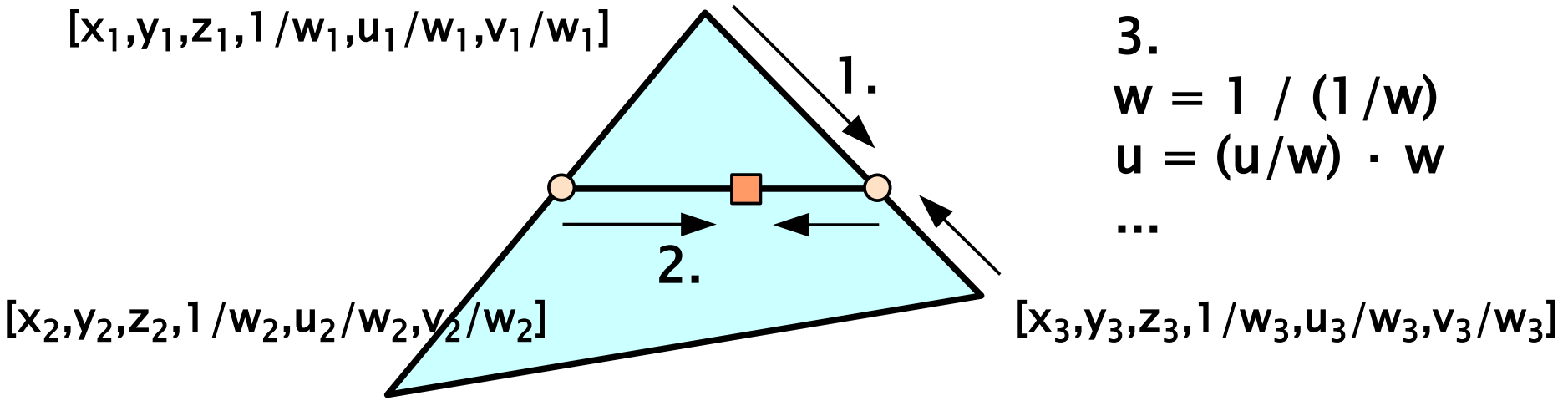
Interpolace na obrazovce

- týká se souřadnic od **ořezávacích** (včetně)
  - clip space: [x, y, z, w]
  - NDS: [x/w, y/w, z/w, w] ("w" někdy zůstává)
  - window space (fragment): [x_i, y_i, z_i, w]
- **projekční perspektivní transformace** mapuje hloubku **z** do NDS **nelineárně!**

  – **nerovnoměrné využití přesnosti** z-bufferu (méně přesné vzdálené partie: minimalizace poměru **f/n** !)

  \+ **interpolaci hloubky z** lze dělat na obrazovce **lineárně**
- **W-buffer** místo z-bufferu (dnes se moc nepoužívá)

Perspektivně korektní interpolace

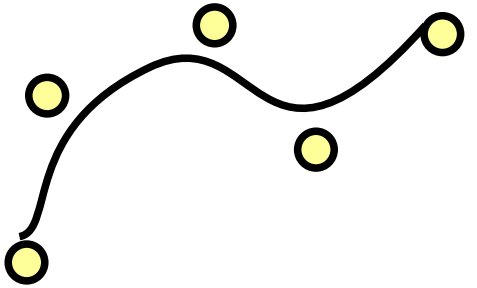


Snahou je používat co nejvíce **lineární interpolaci**

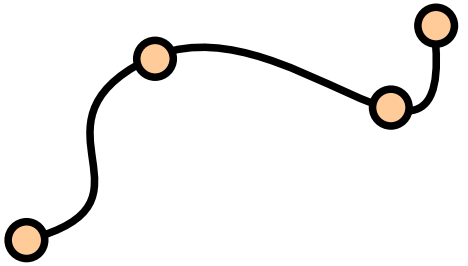
- **hloubku** „**z**” lze interpolovat lineárně
- pro **texturové souřadnice** nebo „**w**” se musí implementovat perspektivně-korektní interpolace:
- **lineárně** interpoluji „**1/w**”, „**u/w**” i „**v/w**”, **[u, v]** pak dopočítám dělením (**hyperbolická interpolace**)

Aproximace a interpolace

- **aproximace** (např. B-spline)
  - křivka neprochází přesně řídícími body



- **interpolace** (např. Catmull-Rom)
  - křivka prochází řídícími body



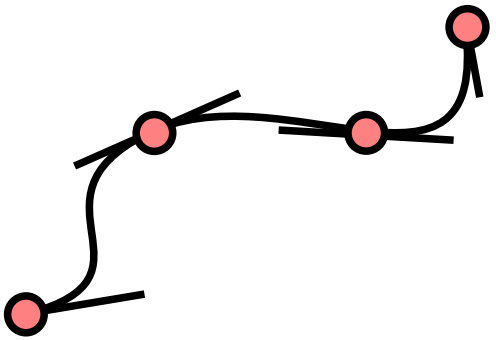
- **spojitost křivky**
  - $\mathbf{G^n}$ – geometrická spojitost n-tého řádu ($\mathbf{G^0}$ – prostá spojitost, $\mathbf{G^1}$ – tečna, $\mathbf{G^2}$ – křivost)
  - $\mathbf{C^n}$ – analytická spojitost n-tého řádu, spoj. n-té derivace ($\mathbf{C^1}$ – rychlost, $\mathbf{C^2}$ – zrychlení), silnější než geometrická

Historie

- **křivky pro modelování**
  - Paul de Faget **de Casteljau** u firmy Citroën (1959)
  - Pierre **Bèzier** (u firmy Renault 1933-1975, UNISURF)
    - začal později, ale jeho výsledky se lépe prosadily
  - aplikace spline funkcí – zejména v USA (James **Ferguson**, 1964, Boeing, C^2 spline křivky)
- **teorie spline funkcí**
  - B-spline: Isaac Jacob **Schoenberg**, (balistika, Aberdeen, MD, 1946)
  - rozvoj teorie: Carl **de Boor** (zkuš. z General Motors)
  - Gordon, Riesenfeld sjednotili Bèzierovy a B-spline křivky (1972)

„Free–form“ křivky

- definují se posloupností **řídících bodů**
  - řídící polygon
  - aproximace nebo interpolace
  - jiné podmínky na okrajích

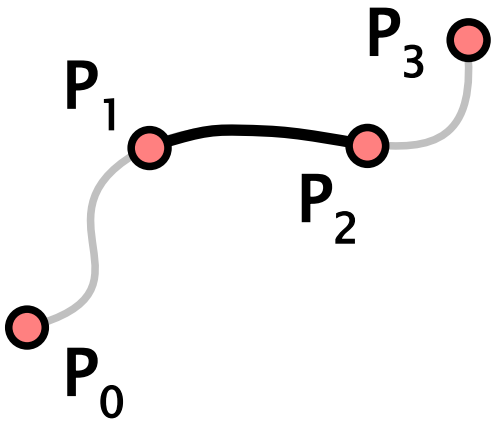


- **kontrolovatelnost**
  - v uzlech se někdy přidávají i tečné vektory (Hermit)
  - interpolace ⇒ přísnější kontrola
- **lokalita**
  - změnou jednoho řídícího bodu (jednoho tečného vektoru) se změní jen omezené okolí křivky

„Free–form“ křivky

- parametrické vyjádření (**0 ≤ t ≤ 1**):

$$P(t) = \sum_{i=0}^{N-1} w_i(t)\, P_i$$



- vlastnost **konvexního obalu**
  - křivka leží v konvexním obalu svého řídícího polygonu
- **Cauchyova podmínka** pro váhové funkce
  - postačující pro konvexní obal
  - zajišťuje invarianci na afinní transformace

$$\sum_{i=0}^{N-1} w_i(t) = 1$$

Spline funkce

Pojmenování podle pružného pravítka používaného v loďařském designu (podepřeného v několika bodech)

- definice – **spline funkce stupně n**:
  - po částech polynom (stupně **n**)
  - napojení s maximální možnou hladkostí:

    $\mathbf{C^{n-1}}$ – spojitost **(n-1).** derivace (pro polynomy stupně **n**)
  - globální parametrizace **u**, $\mathbf{u_o \le u \le u_N}$ $[u_o, u_1, \dots u_N]$
  - jednotlivé části jsou často parametrizované pravidelně – uniformní spline: $\mathbf{t_i = (u - u_i) / (u_{i+1} - u_i)}$, $\mathbf{0 \le t_i \le 1}$

Polynomiální křivka

- maticové vyjádření:

$$P(t) = \boldsymbol{TC} = [\, t^n, t^{n-1}, \ldots t, 1 \,] \cdot \begin{bmatrix} x_n & y_n & z_n \\ x_{n-1} & y_{n-1} & z_{n-1} \\ .. & .. & .. \\ x_1 & y_1 & z_1 \\ x_0 & y_0 & z_0 \end{bmatrix}$$

- bázová matice **M** a vektor geometrických podmínek **G**:

$$\boldsymbol{C} = \boldsymbol{MG} = [\, m_{ij} \,]_{i=n,\, j=1}^{0,\, k} \cdot \begin{bmatrix} G_1 \\ .. \\ G_k \end{bmatrix} \qquad P(t) = \boldsymbol{TMG}$$

Maticové vyjádření křivky

- **P(t) = T C = T M G**
  - oddělení parametru (**T**) od polynomiální báze (**M**) a geometrie řídících prvků (**G**)
  - derivace (tečna, křivost) lze dělat jen s **T**
  - řídící polynom **TM** krát geometrie **G**

- **kubika**: n = 3, k = 4

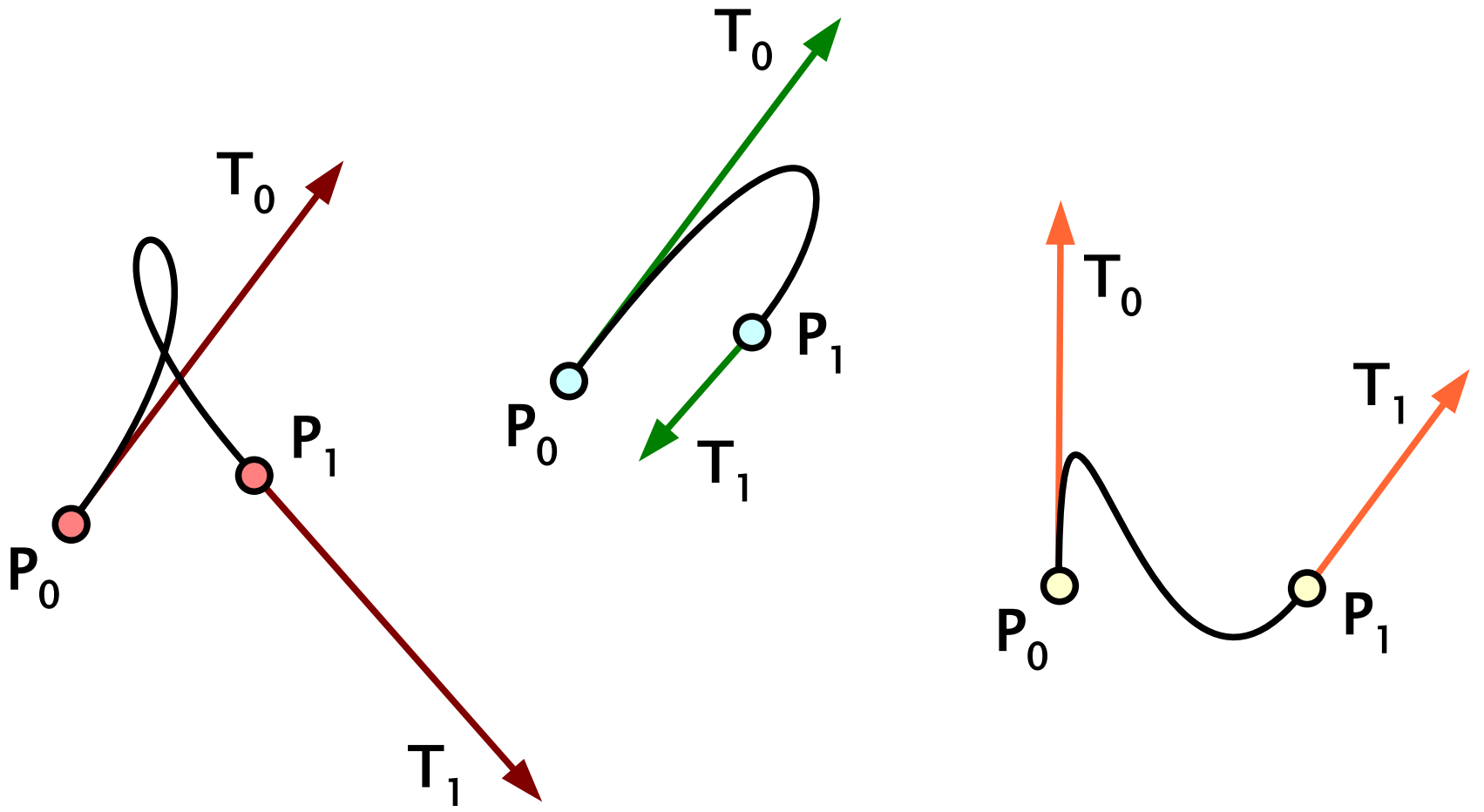
$$Q(t) = [\, t^{3,}\ t^{2,}\ t,\ 1\ ] \cdot \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \cdot \begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix}$$

Hermitovská kubika

- **Fergusonova křivka** (kubika)
- geometrie: **koncové body** a **tečné vektory** v nich
  - začátek křivky ($\mathbf{P_0}$), konec křivky ($\mathbf{P_1}$)
  - tečna na začátku ($\mathbf{T_0}$) a na konci ($\mathbf{T_1}$)

$$F(t) = [\, t^3,\ t^2,\ t,\ 1\,] \cdot \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} P_0 \\ P_1 \\ T_0 \\ T_1 \end{bmatrix}$$

Hermitovská kubika



Další křivky

- **interpolační kubiky** odvozené od Hermitovy:
  - obecná: **Kochanek-Bartels** (KB-spline, TCB kubika)
  - speciální: **cardinal** spline, **Catmull-Rom** spline
  - **Akimova** interpolace (není C^2)
  - **D-spline** kubika

- další křivky:
  - **Bèzierovy** křivky
  - **B-spline** křivka, **Coonsův** spline (aproximace)
  - **přirozený** spline (interpolace)

Kochanek–Bartels kubika (KB-spline)

- odvozená od **Hermitovy** kubiky (3DS Max, Lightwave)
  - výpočet tečných vektorů z řídících bodů
  - tři další parametry (implicitně nulové):
    - **napětí**, **tension t**: ostrost křivky v řídícím bodě (absolutní velikost tečny)
    - **spojitost**, **continuity c**: spojitost v řídícím bodě
    - **sklon**, **bias b**: směr tečny v řídícím bodě

Levá a **pravá** tečna ($\mathbf{T_0}$ a $\mathbf{T_1}$ v lokálním smyslu):

$$L_i = \frac{(1-t)(1-c)(1+b)}{2}\cdot(P_i - P_{i-1}) + \frac{(1-t)(1+c)(1-b)}{2}\cdot(P_{i+1} - P_i)$$

$$R_i = \frac{(1-t)(1+c)(1+b)}{2}\cdot(P_i - P_{i-1}) + \frac{(1-t)(1-c)(1-b)}{2}\cdot(P_{i+1} - P_i)$$

Cardinal spline, Catmull–Rom spline

Speciální případy KB-spline:

- **cardinal spline**
  - parametr **a** (ve skutečnosti souvisí s "**t**", **c = b = 0**)

$$T_i = a \cdot (P_{i+1} - P_{i-1}) \qquad 0 \leqslant a \leqslant 1$$

- **Catmull-Rom spline**
  - a = t = 1/2

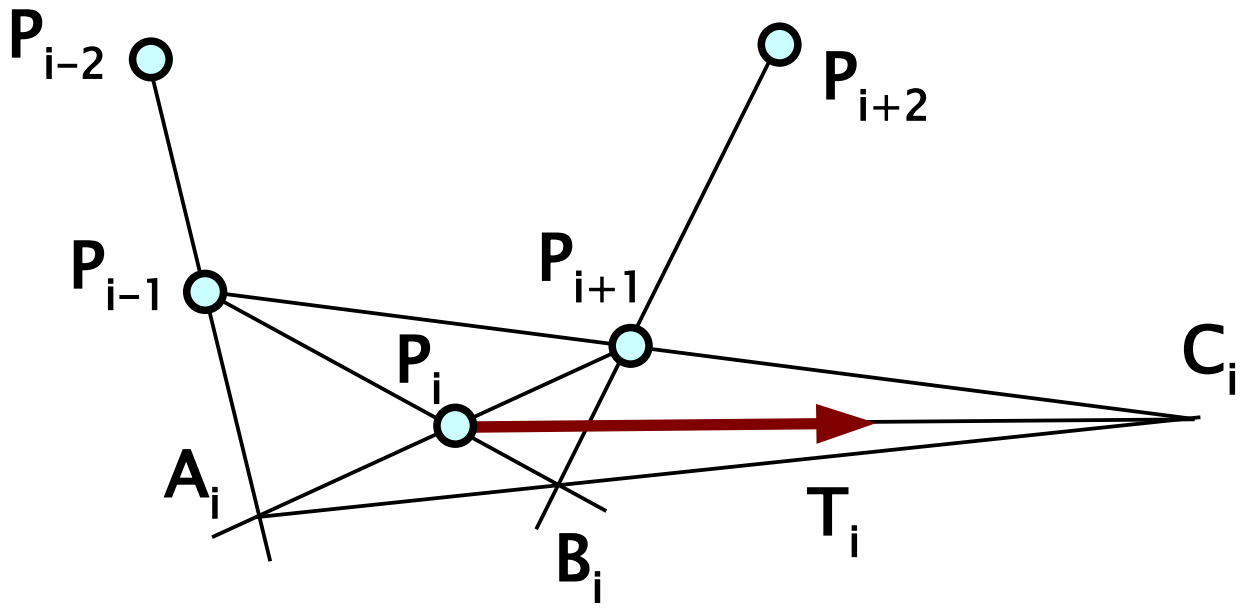
$$T_i = \frac{1}{2} \cdot (P_{i+1} - P_{i-1})$$

$$\boldsymbol{MG} = \frac{1}{2} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix}$$

Akimova interpolace

Další alternativa výpočtu tečných vektorů pro Hermitovu kubiku:

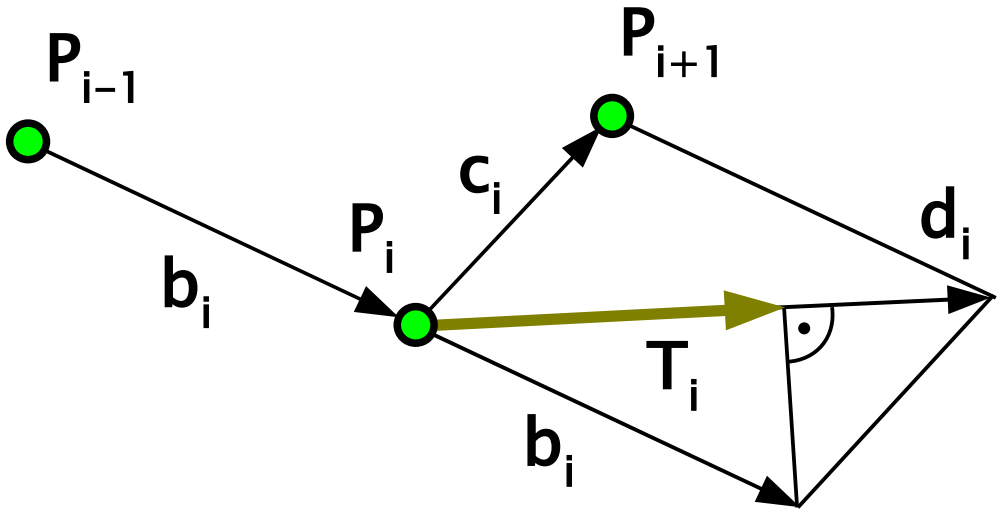
- není C^2 !



$$|T_i| = |P_{i+1} - P_{i-1}|$$

D-spline kubika

- další varianta Hermitovy kubiky
  - tečný vektor se spočítá tzv. **D-interpolací**



$$\boldsymbol{G} = \begin{bmatrix} P_i \\ P_{i+1} \\ T_i \\ T_{i+1} \end{bmatrix}$$

Bèzierovy křivky

- **polynomiální křivky stupně N**
  - **N+1** řídících bodů
    - krajní řídící body definují přímo konce křivky
    - krajní dvojice definují tečné vektory křivky na koncích
  - parametrické vyjádření používá Bernsteinovy polynomy
  - lze je snadno napojovat **G¹** nebo **C¹** spojitostí
  - složitějším způsobem z nich lze postavit spline

Bernsteinovy polynomy:

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \quad 0 \leqslant i \leqslant n, \quad 0 \leqslant t \leqslant 1$$

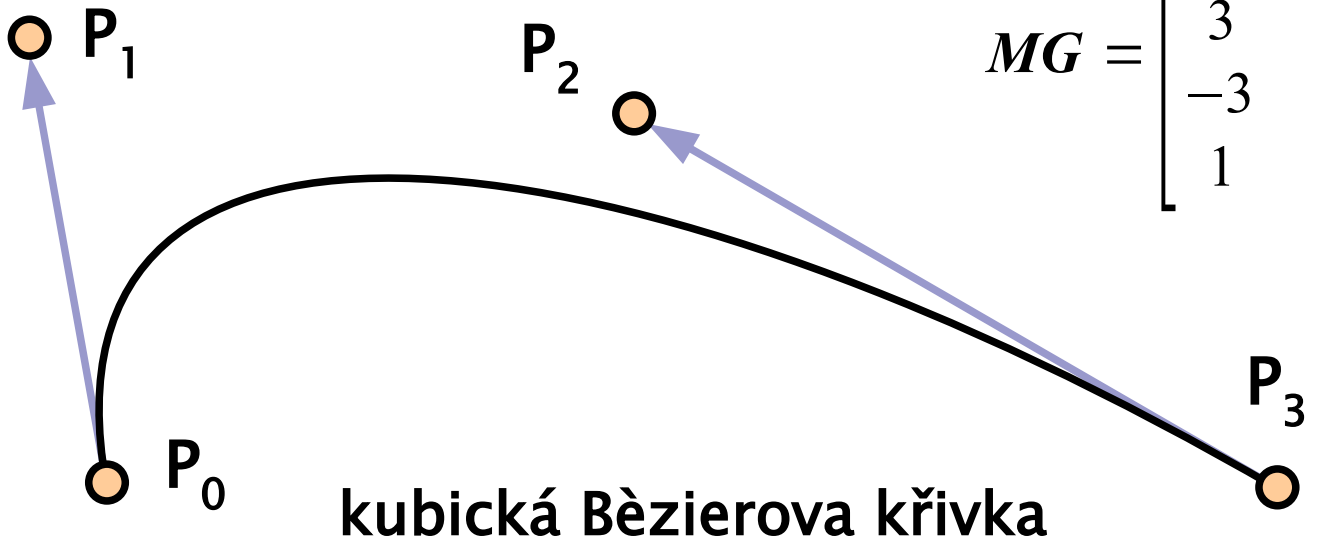
Bèzierovy křivky

Cauchyova podmínka

$\Rightarrow$ konvexní kombinace řídících bodů

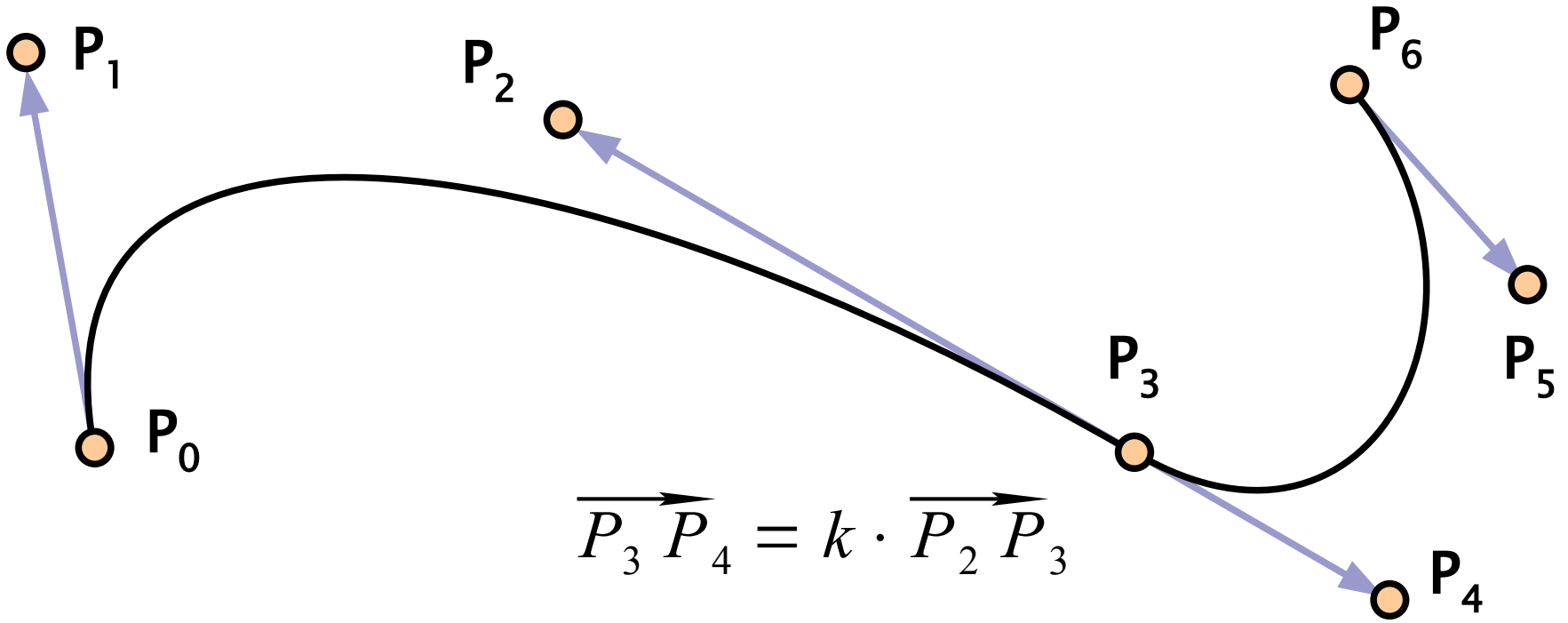
$$\sum_{i=0}^{n} B_i^n(t) = 1 \qquad pro \quad 0 \leqslant t \leqslant 1$$

$$\boldsymbol{MG} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$



Napojování Bèzierových křivek

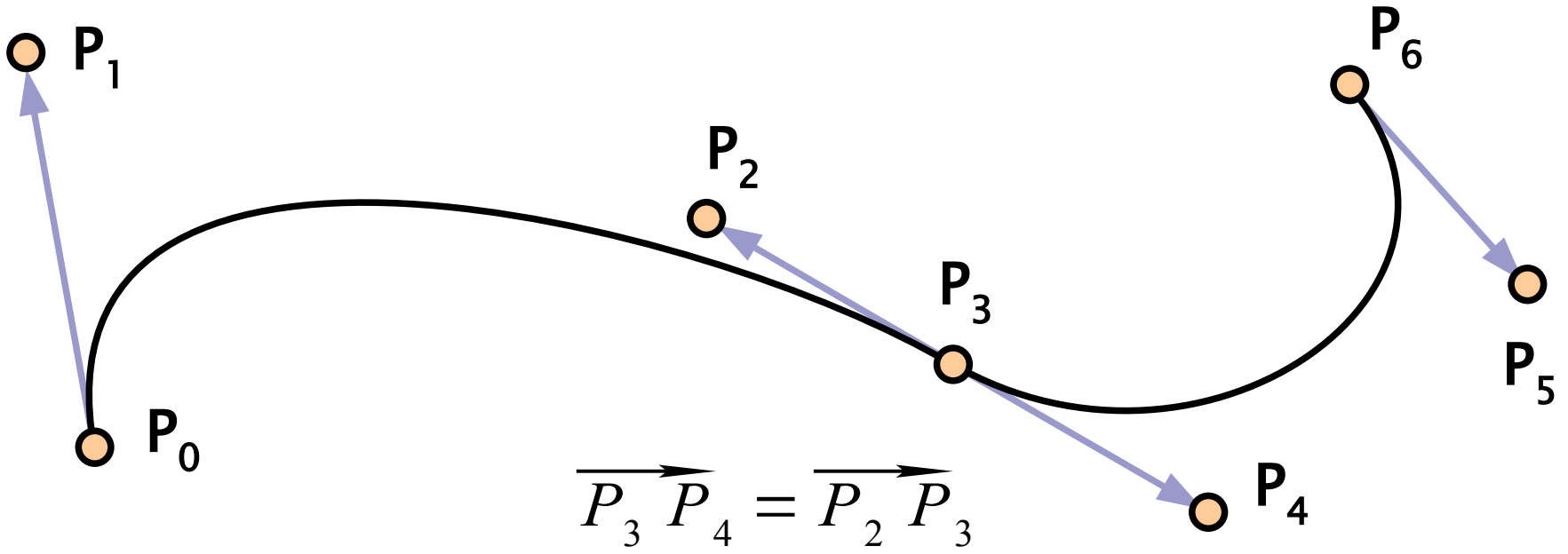
G^1 napojení (kolineární tečny)



$$\overrightarrow{P_3 P_4} = k \cdot \overrightarrow{P_2 P_3}$$

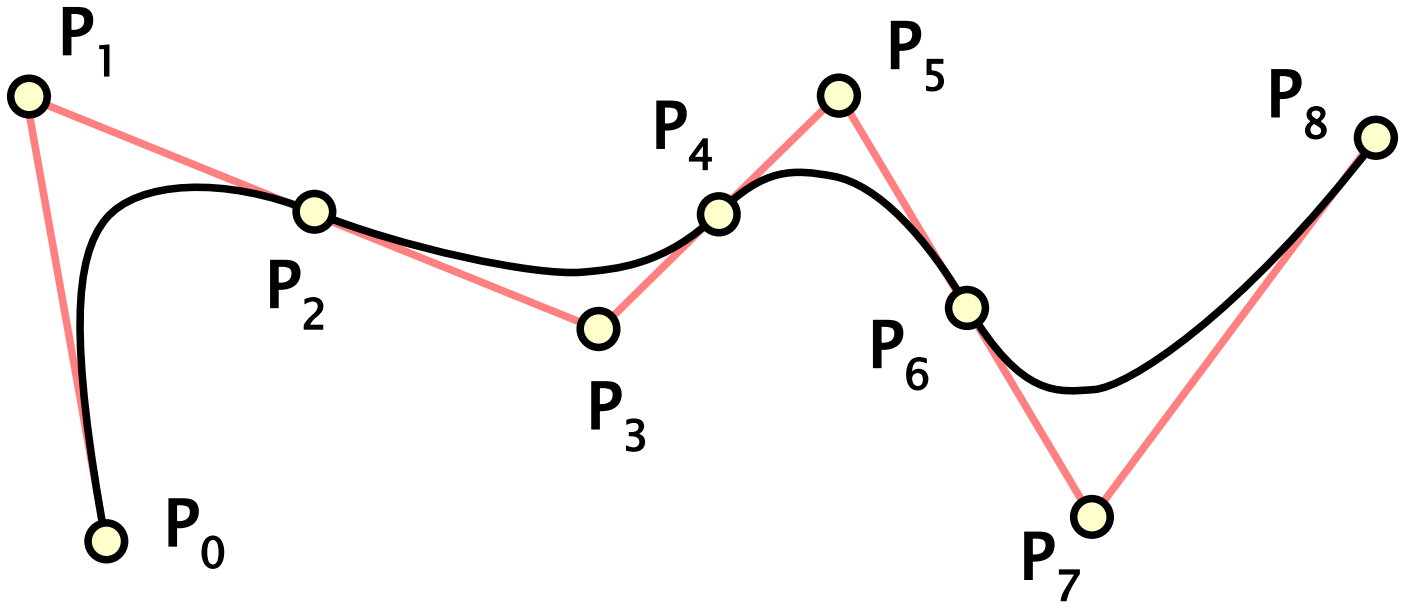
Napojování Bèzierových křivek II

C^1 napojení (shodné tečny)



Napojování Bèzierových křivek III

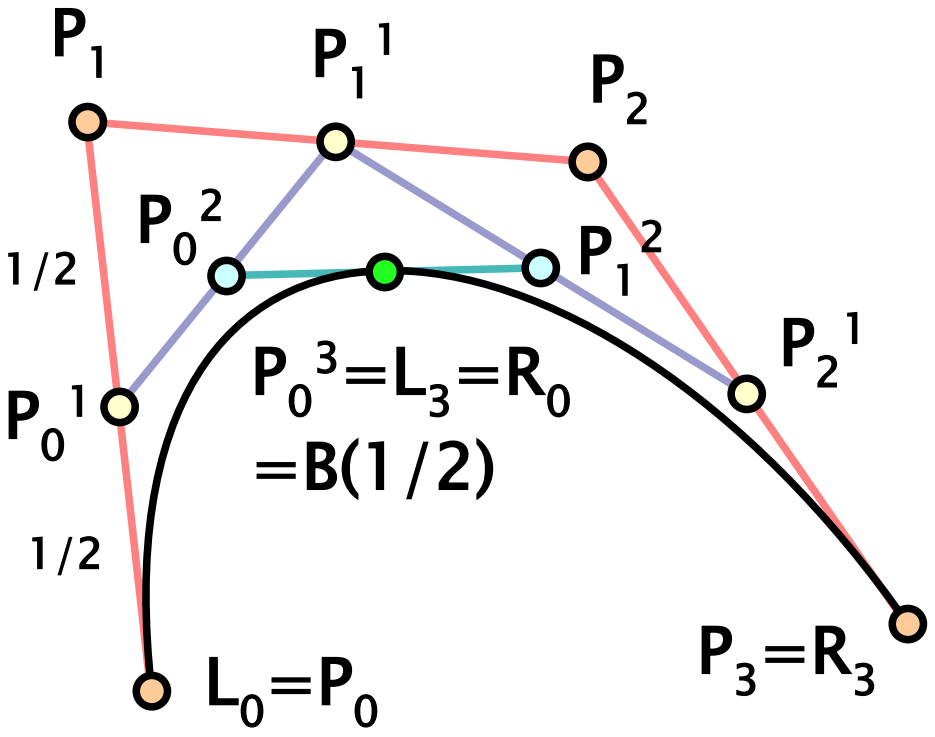
Kvadratický spline z Bèzierových segmentů

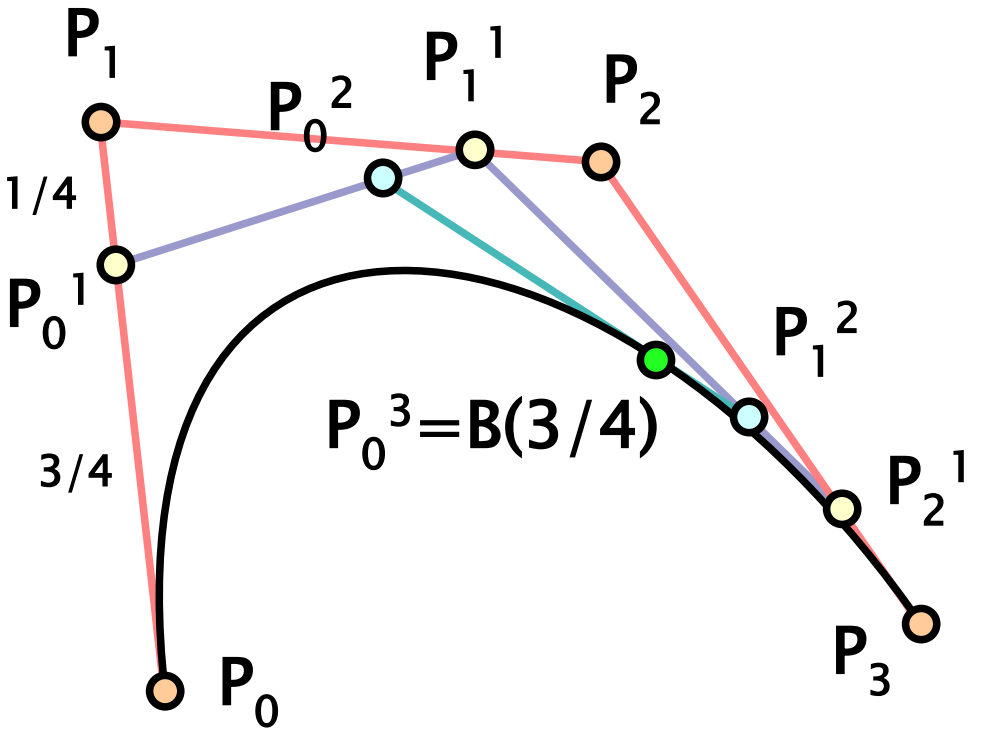


$$\overrightarrow{P_1P_2} = \overrightarrow{P_2P_3} \quad \overrightarrow{P_3P_4} = \overrightarrow{P_4P_5} \quad \dots \quad \overrightarrow{P_{2k-1}P_{2k}} = \overrightarrow{P_{2k}P_{2k+1}}$$

De Casteljau (de Boor) algoritmus

- **geometrická konstrukce** Bèzierovy křivky
  - "**subdivision**" schéma i výpočet jednoho daného bodu





Kubický spline

- funkce sestavená z **kubických polynomů**
  - sousední polynomy jsou napojeny C^2
  - vyjádření pružného "spline-pravítka" (viz konstrukce)
- **interpolační kubický spline**
  - v uzlech $\mathbf{x_0}, \mathbf{x_1}, \ldots \mathbf{x_n}$ jsou předepsány funkční hodnoty $\mathbf{y_0}, \mathbf{y_1}, \ldots \mathbf{y_n}$

$$S(x) = S_k(x) = s_{k,0} + s_{k,1}(x - x_k) + s_{k,2}(x - x_k)^2 + s_{k,3}(x - x_k)^3$$
$$x \in [x_k, x_{k+1}], \quad k = 0, 1, \ldots, n-1$$

- podm. **A**: $S(x_k) = y_k \qquad k = 0, 1, \ldots, n$

Interpolační kubický spline

- podm. **B** (C^0 spojitost):

$$S_k(x_{k+1}) = S_{k+1}(x_{k+1}) \qquad k = 0, 1, \ldots, n-2$$

- podm. **C** (C^1 spojitost):

$$S_k'(x_{k+1}) = S_{k+1}'(x_{k+1}) \qquad k = 0, 1, \ldots, n-2$$

- podm. **D** (C^2 spojitost):

$$S_k''(x_{k+1}) = S_{k+1}''(x_{k+1}) \qquad k = 0, 1, \ldots, n-2$$

- pro **přirozený kubický spline** navíc podm. **E**:

$$S''(x_0) = S''(x_n) = 0$$

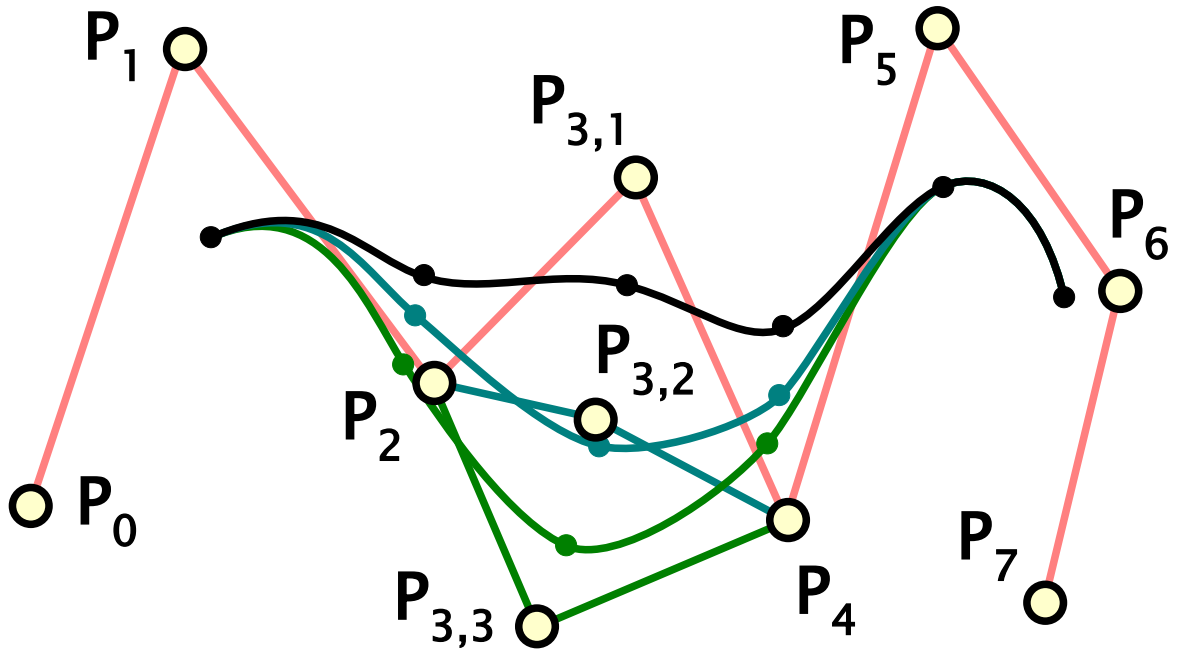
Přirozený kubický spline

- **interpolační spline**
  - jednoznačně určen danými podmínkami (řeší se soustava lineárních rovnic pro $\mathbf{s}_{k,l}$)
  - nemá lokální vlastnost (při změně jednoho uzlu se změní celý spline)
- **otevřený spline**:
  - podmínky **A**, **B**, **C**, **D** nestačí, zbývají 2 stupně volnosti
  - přidává se podmínka **E** (druhé derivace na okraji)
- **uzavřený** (cyklický) **spline**: $\mathbf{x}_0 = \mathbf{x}_n$
  - **C** a **D** nám pro $\mathbf{x}_0$ dají chybějící dvě podmínky

B-spline (basis spline)

- "**free-form**" křivka
  - tvar je definován posloupností **řídících bodů**
  - parametrické vyjádření pomocí **bazických funkcí** (závislost bodu křivky na řídících bodech)
  - **má lokální vlastnost** (změna řídícího bodu změní křivku jen lokálně)
- **uniformní kubický B-spline** (Coonsova křivka)
  - jednotná sada bazických funkcí (kubických polynomů)
- **neuniformní B-spline**
  - složitější definice pomocí posloupnosti uzlů $\mathbf{0 \le t_i \le 1}$

Coonsův B-spline

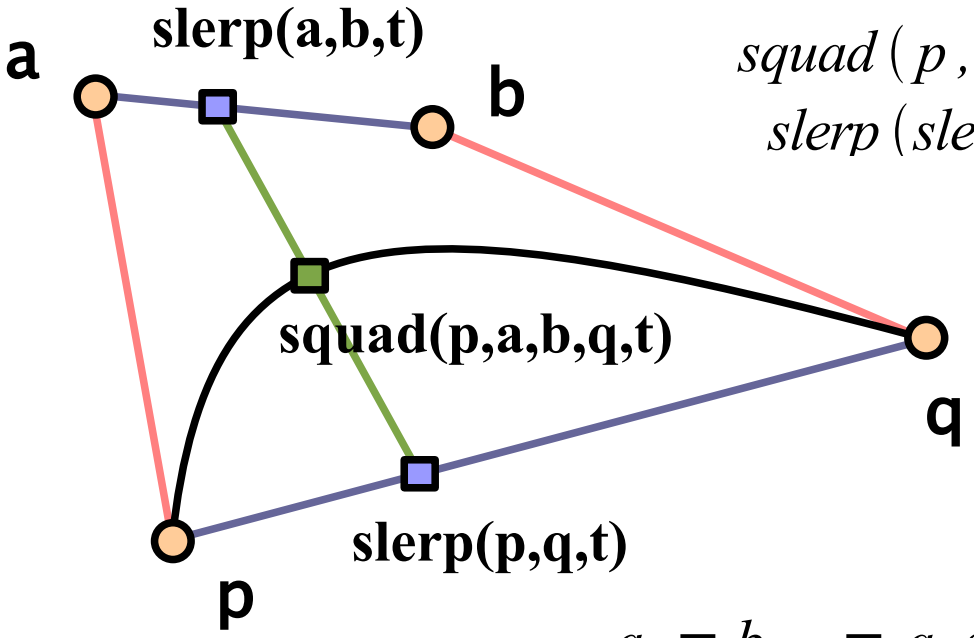


- automatická spojitost C^2
- sdílení 3 řídících bodů
- změna řídícího bodu ovlivní sousední 4 segmenty

$$MG = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix}$$

Spline interpolace kvaternionů

- postupná interpolace posloupností orient. $\mathbf{q_0}$, $\mathbf{q_1}$, ... $\mathbf{q_n}$
  - **slerp($\mathbf{q_i}$,$\mathbf{q_{i+1}}$,t)** dává málo hladkou křivku (jen C^0)



$$squad(p,a,b,q,t) = slerp(slerp(p,q,t), slerp(a,b,t), 2t(1-t))$$

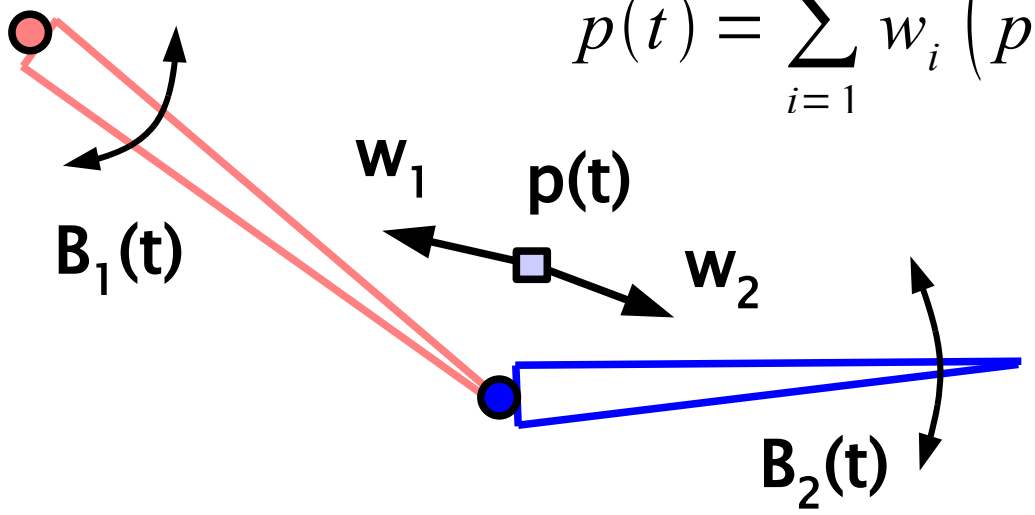
$$s_i(t) = squad(q_i, a_i, b_i, q_{i+1}, t)$$

$$a_i = b_{i-1} = q_i \exp\left[-\frac{\log(q_i^{-1}q_{i-1}) + \log(q_i^{-1}q_{i+1})}{4}\right]$$

Vertex blending = skinning

- **vrchol** každého trojúhelníka **p** je navázán na několik okolních "**kostí**" (souřadných soustav)
  - klidová poloha: t = 0
  - kosti se **animují** maticemi $\mathbf{B_i(t)}$ (object sp. → world)
  - počáteční transformace $\mathbf{M_i = B_i(0)}$

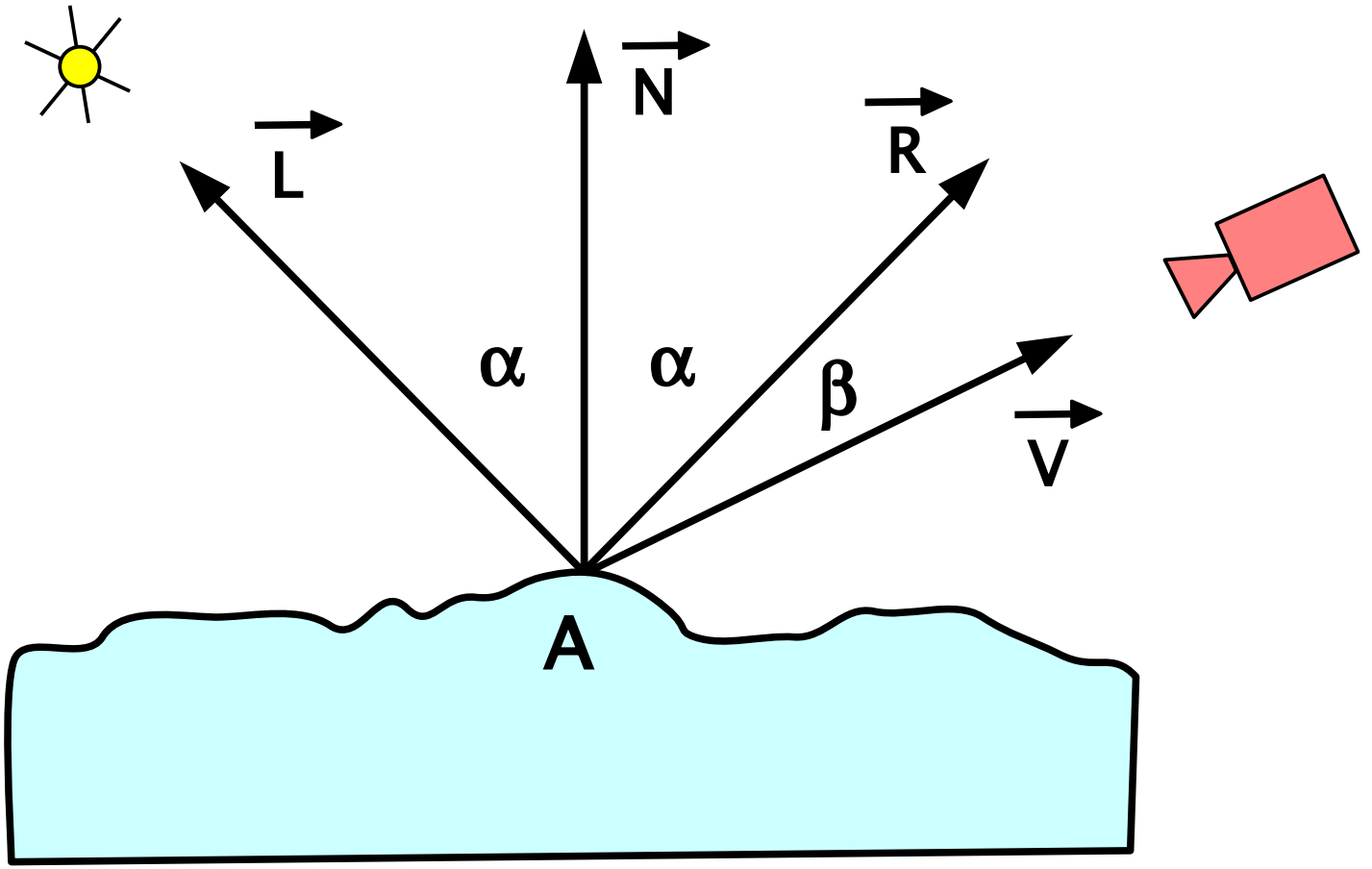
$$p(t) = \sum_{i=1}^{n} w_i \left(p\, M_i^{-1} B_i(t) \right), \qquad \sum_{i=1}^{n} w_i = 1$$



Stínování

- pozorovaný **odstín** na povrchu plošky je závislý na její orientaci v prostoru + na poloze světelných zdrojů
- existuje mnoho **osvětlovacích modelů**
  - lokální výpočet odrazu světla na povrchu tělesa
  - ideálem je přesné napodobení přírody
- **Phongův světelný model**
  - jednoduchý výpočet
  - kvalitativně přibližně souhlasí s fyzikální realitou
  - kvůli lesklé složce je nejlepší počítat ho v každém pixelu s interpolací normály

Situace



Phongův model osvětlení

- světlo se skládá ze tří složek:
  - **zbytkové světlo** („*ambient*”) – náhrada za nepočítané sekundární odrazy
  - **difusní odraz** („*diffuse*”) – ideálně matné těleso
  - **lesklý odraz** („*specular*”) – neostrý odraz, odlesk

$$L_a = C \cdot k_a$$
$$L_d = (C \cdot C_L) \cdot k_d \cdot \cos\alpha$$
$$L_s = C_L \cdot k_s \cdot \cos^h \beta$$

Parametry Phongova modelu

- optické vlastnosti povrchu tělesa („**materiál**”):
  - vlastní barva tělesa „**C**” (při osvětlení bílým světlem)
  - koeficienty „$\mathbf{k_a}$”, „$\mathbf{k_d}$” a „$\mathbf{k_s}$” (charakter povrchu: matný, lesklý, ..)
  - exponent odlesku „**h**” (čím větší, tím je odlesk ostřejší)
- vlastnosti **zdroje světla**:
  - barva a intenzita „$\mathbf{C_L}$” (třísložkový vektor, s barvou zdroje se násobí po složkách)
- **více** zdrojů světla:

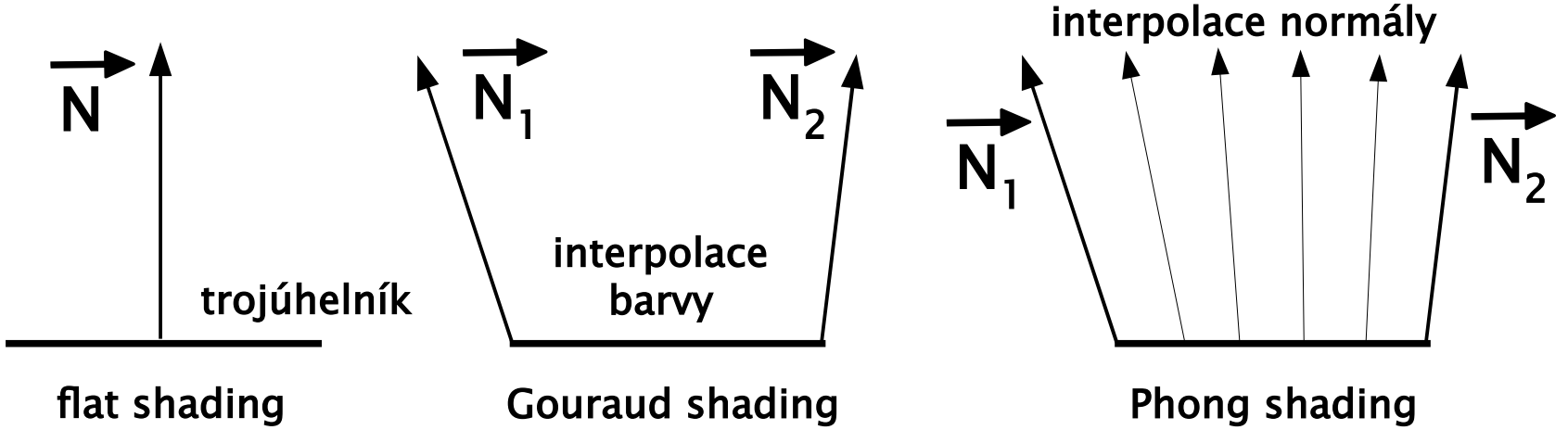
$$L = L_a + \sum L_d + L_s$$

Spojité stínování

Tři techniky použití osvětlovacího modelu:

- **konstantní stínování** („*flat shading*”)
  - celá ploška se vybarví stejným odstínem
- **Gouraudova interpolace barvy**
  - osvětlení se počítá ve vrcholech, uvnitř plošek se barva lineárně interpoluje (umějí běžné HW rasterizéry)
- **Phongova interpolace normály**
  - pro každý pixel plošky se normálový vektor lineárně interpoluje a spočítá světelný model
  - nejdokonalejší metoda, vyžaduje spolupráci HW (shader)

Spojité stínování



- **konstantní stínování** je adekvátní u hranatých těles
- **interpolace barvy** potlačuje „hranatý" vzhled aproximovaných těles
  - ostré odlesky se však nekreslí korektně
- **interpolace normály** je vhodná i pro vysoký lesk

Mlha

- lokální osvětlovací modely počítají pouze interakci světla s **povrchem předmětů**
- vliv **prostředí** (atmosféry) na šíření paprsku
  - nejjednodušší a nejčastější je výpočet **mlhy** (umí i HW)
  - přesný výpočet pohlcení/rozptylu paprsku v kouři nebo aerosolu je mnohem složitější
- **barva mlhy „C_f”** (obvykle bílá) se mísí s barvou tělesa
  - **lineární mlha** (koeficient se mění lineárně se vzdáleností)
  - **exponenciální mlha** (exponenciální závislost koeficientu na vzdálenosti předmětu od pozorovatele)

Mlha

- míchání mlhy s barvou tělesa
  - „$\mathbf{C_f}$" je barva mlhy, „$\mathbf{C_s}$" barva tělesa
  - „**f**" je koeficient

$$C = f \cdot C_s + (1 - f) \cdot C_f$$

- **lineární mlha**
  - „$\mathbf{z_{end}}$" a „$\mathbf{z_{start}}$" jsou vzdálenosti začátku a konce mlhy
  - „**z**" je vzdálenost fragmentu od pozorovatele
- **exponenciální mlha**
  - „**D**" je hustota mlhy (čím je větší, tím je mlha hustší)

$$f = \frac{z_{end} - z}{z_{end} - z_{start}} \qquad f = e^{-D \cdot z}$$

Literatura

- Tomas Akenine-Möller, Eric Haines: ***Real-time rendering, 2nd edition***, A K Peters, 2002, ISBN: 1568811829
- Randima Fernando, Mark J. Kilgard: ***The Cg Tutorial***, Addison-Wesley, 2003, ISBN: 0321194969
- J. Žára, B. Beneš, J. Sochor, P. Felkel: ***Moderní počítačová grafika***, 2. vydání, Computer Press, 2005, ISBN: 8025104540
- **http://www.geometrictools.com/** (Dave Eberly)