Hardware pro počítačovou grafiku NPGR019

Matematika pro real-time grafiku

Josef Pelikán
Jan Horáček

http://cgg.mff.cuni.cz/
MFF UK Praha

2012

Obsah

Geometrické transformace v 3D

- vektor 3D souřadnic $[x, y, z]$
- transformace násobením maticí 3x3
  - **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*)

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

- transformační matice 3x3 mají podstatné omezení - **nelze** je použít pro **posunutí** (translaci)

Homogenní souřadnice

- vektor **homogenních souřadnic** $[x, y, z, w]$
- transformace násobením maticí $4x4$

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

- homogenní matici 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 $w \neq 0$) $[x/w, y/w, z/w]$
- 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: $[x, y, z] \Rightarrow [x, y, z, 1]$

Elementární transformace

- nejběžnější jsou **afinní transformace**

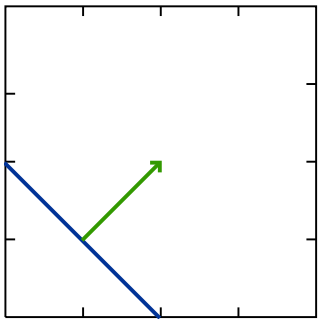
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ t_1 & t_2 & t_3 & 1 \end{bmatrix}$$

- 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

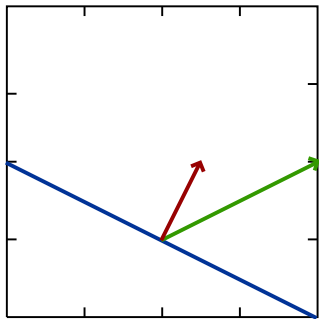
Transformace normál

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

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

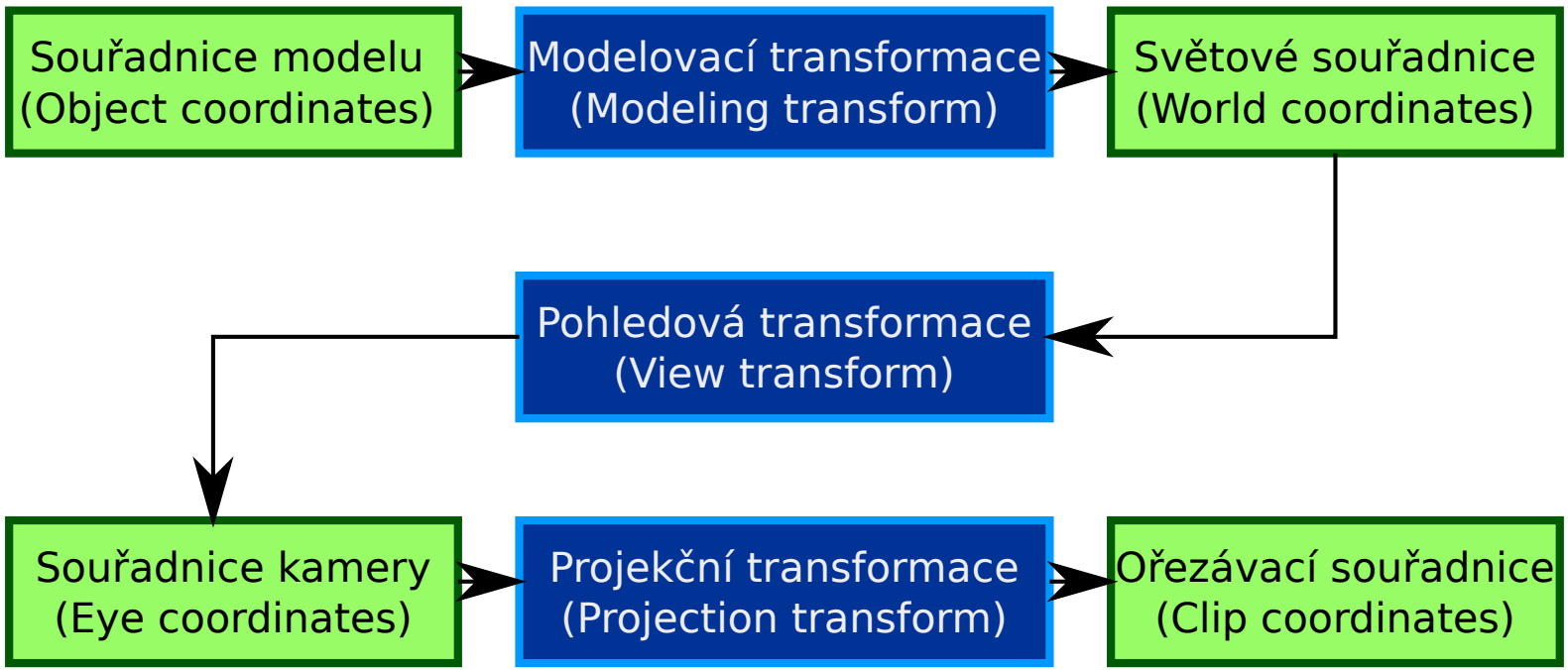


Původní objekt

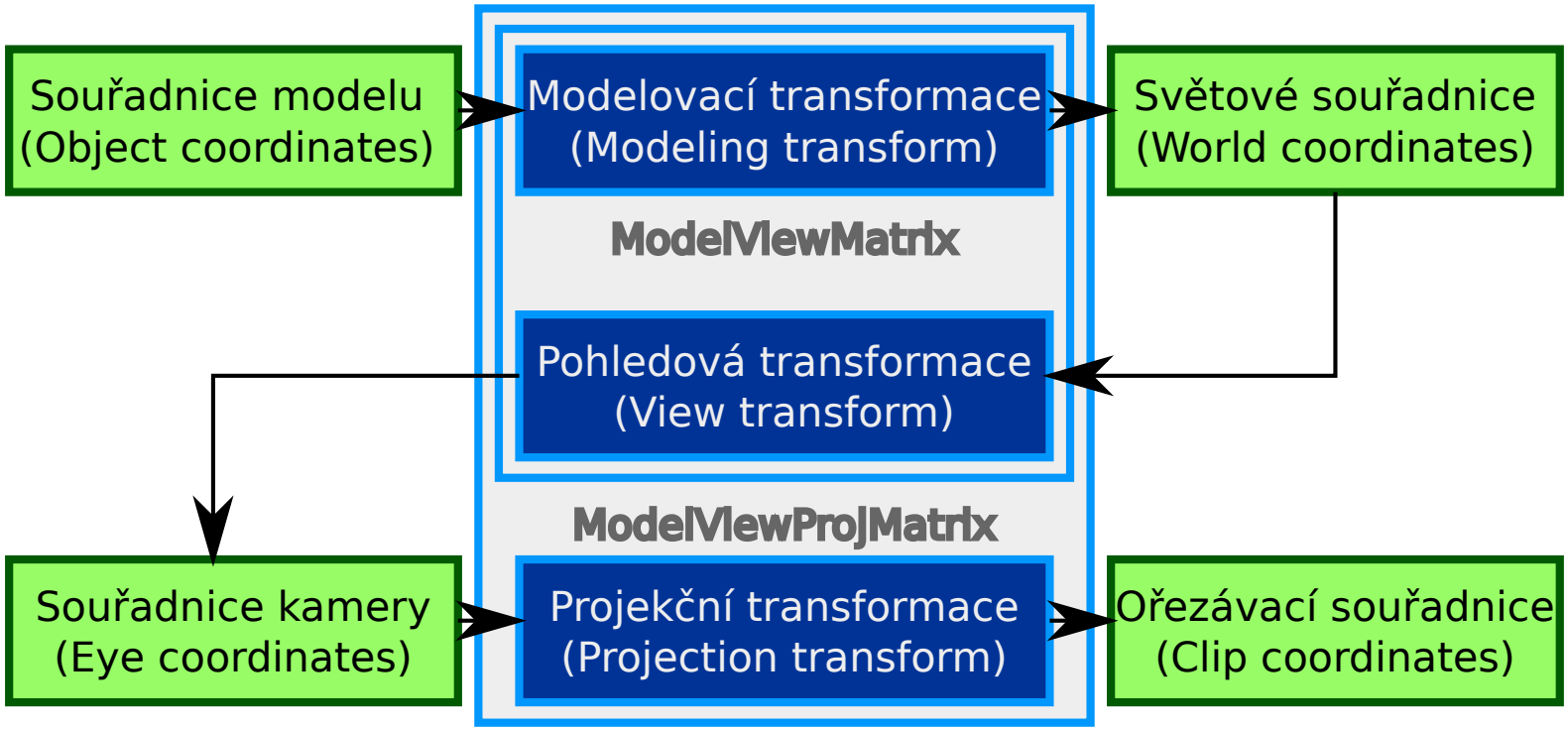


Transformovaný objekt

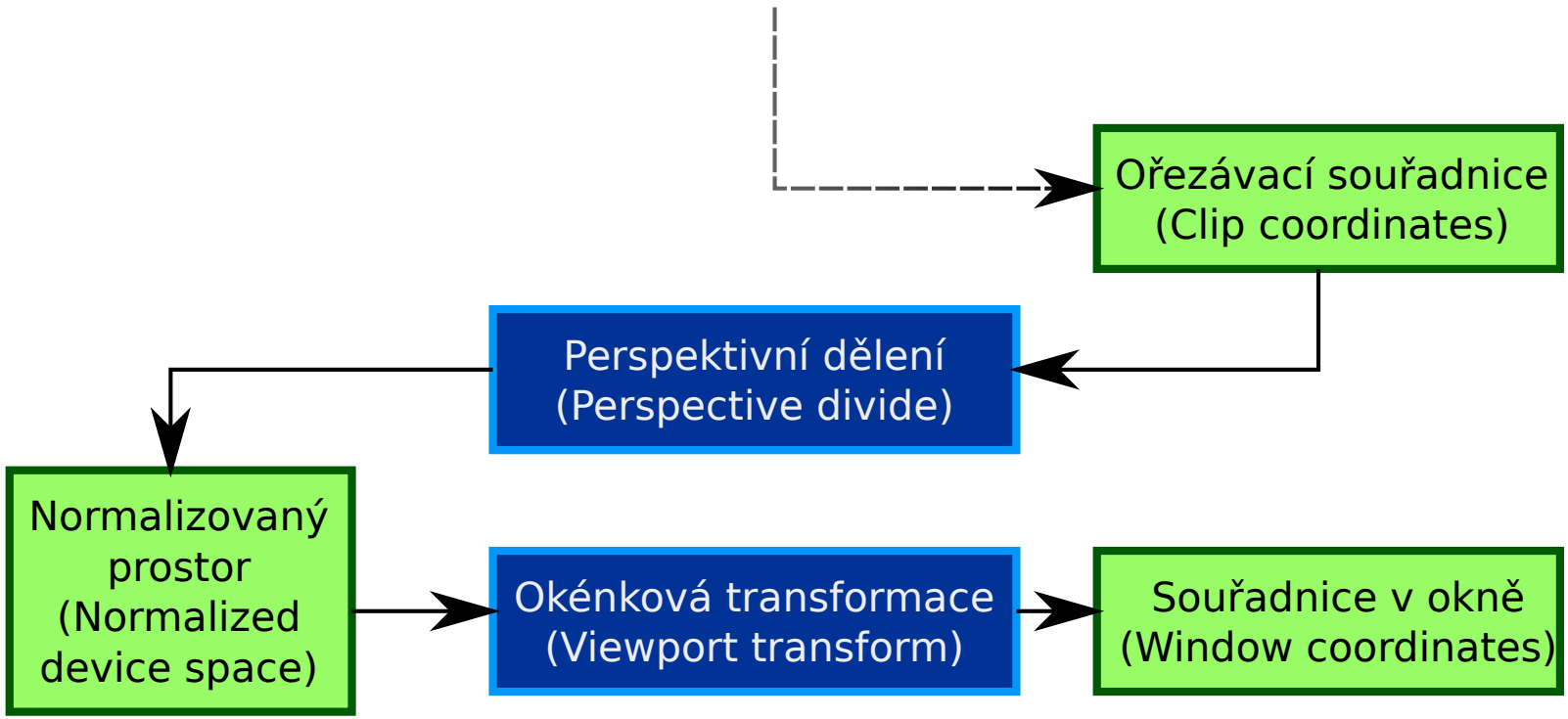
Souřadné soustavy



Souřadné soustavy



Souřadné soustavy 2



Souřadné soustavy 3

- 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*
  - pro zobrazování nemají příliš velký význam (pouze teoretický)
- souřadnice kamery
  - 3D svět se transformuje do relativních souřadnic kamery
  - střed projekce: **počátek**, směr pohledu: -**Z** (OpenGL) nebo +**Z** (DirectX)

Souřadné soustavy a jejich transformace

- transformace model ⇒ kamera
  - kombinovaná do **ModelView** matice
- projekční transformace
  - definuje zorný objem (*frustum* pro perspektivní projekci)
  - přední a zadní ořezávací vzdálenost: **n**, **f**
  - výsledkem je homogenní souřadnice (před ořezáním)
- ořezávací souřadnice (clip coordinates)
  - výstupní souřadnice vertex shaderu

Projekční transformace

- tzv. **frustum** pro perspektivní projekci je definováno parametry **n**, **f**, **l**, **r**, **t**, **b**
- vzdálený bod **f** lze pousnout do *nekonečna*

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

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

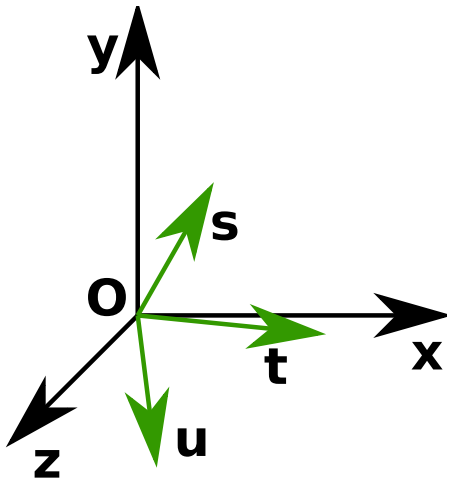
Souřadné soustavy a jejich transformace

- 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 coordinates)
  - výsledkem okénkové transformace (škálování + posun) a transformace hloubky
  - používají se při **rasterizaci** a práci s **fragmenty**

Transformace tuhého tělesa

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

Převod mezi dvěmi orientacemi



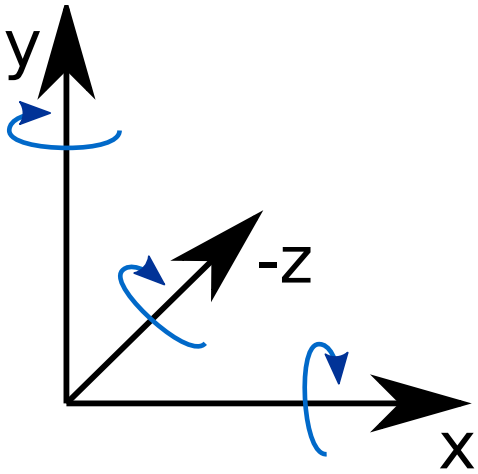
$$[1,0,0] \cdot M_{stu \to xyz} = s$$
$$[0,1,0] \cdot M_{stu \to xyz} = t$$
$$[0,0,1] \cdot M_{stu \to xyz} = u$$

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

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

$$M_{xyz \to stu} = M_{stu \to xyz}^T$$

Eulerova transformace



- rozklad obecné rotace na **tři složky**

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

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

Eulerova transformace 2

- výsledná matice rotace:

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

- 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 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 roce 1985 (Shoemake)
  - **zobecnění komplexních čísel** do 4D prostoru
- $\mathbf{q} = (\mathbf{v}, w) = ix + jy + kz + w = \mathbf{v} + w$
- **imaginární část** $\mathbf{v} = (x, y, z) = ix + jy + kz$
- $i^2 = j^2 = k^2 = -1$
- $jk = -kj = i$
- $ki = -ik = j$
- $ij = -ji = k$

Kvaterniony - operace

- sčítání
  - $(\mathbf{v_1}, w_1) + (\mathbf{v_2}, w_2) = (\mathbf{v_1} + \mathbf{v_2}, w_1 + w_2)$
- násobení
  - $\mathbf{qr} = (\mathbf{v_q} \times \mathbf{v_r} + w_r \mathbf{v_q} + w_q \mathbf{v_r}, w_q w_r - \mathbf{v_q} \cdot \mathbf{v_r})$

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

Kvaterniony - operace

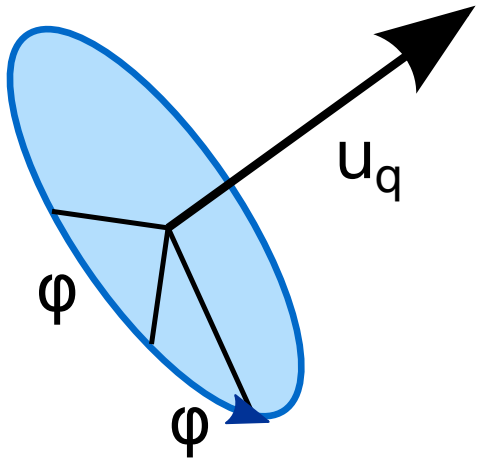
- konjugace
  - $(\mathbf{v}, w)^* = (-\mathbf{v}, w)$
- norma (čtverec absolutní hodnoty)
  - $\| \mathbf{q} \|^2 = n(\mathbf{q}) = \mathbf{q}\mathbf{q}^* = x^2 + y^2 + z^2 + w^2$
- jednotka
  - $\mathbf{i} = (\mathbf{0}, 1)$
- převrácená hodnota
  - $\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{n(\mathbf{q})}$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

Jednotkové kvaterniony

- **jednotkový kvaternion** lze vyjádřit goniometricky
  - $\mathbf{q} = (\mathbf{u_q} sin\varphi, cos\varphi)$
  - 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): $(\mathbf{0}, 1)$
- mocnina, exponenciála, logaritmus:
  - $\mathbf{q} = \mathbf{u_q} sin\varphi + cos\varphi = exp(\varphi \mathbf{u_q})$
  - $log\mathbf{q} = \varphi \mathbf{u_q}$
  - $\mathbf{q^t} = (\mathbf{u_q} sin\varphi + cos\varphi)^t = exp(t\varphi \mathbf{u_q}) = \mathbf{u_q} sin(t\varphi) + cos(t\varphi)$

Rotace pomocí kvaternionu

- jednotkový kvaternion
  - $\mathbf{q} = (\mathbf{u_q} sin\varphi, cos\varphi)$
  - $\mathbf{u_q}$ ... osa rotace
  - φ ... úhel
- bod nebo vektor v 3D:
  $p = [p_x, p_y, p_z, p_w]$
- **rotace** bodu (vektoru) **p** kolem osy $\mathbf{u_q}$ o úhel 2φ:
  - $\mathbf{p'} = \mathbf{qpq^{-1}} = \mathbf{qpq^*}$



Převod kvaternion $\leftrightarrow$ matice

- 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:

$$\begin{aligned} m_{23} - m_{32} &= 4wx \\ m_{31} - m_{13} &= 4wy \\ m_{12} - m_{21} &= 4wz \\ tr(M) + 1 &= 4w^2 \end{aligned} \tag{1}$$

Převod matice na kvaternion 2

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

$$w = \frac{1}{2}\sqrt{tr(M)+1} \quad x = \frac{m_{23}-m_{32}}{4w}$$
$$y = \frac{m_{31}-m_{13}}{4w} \quad x = \frac{m_{12}-m_{21}}{4w}$$

- v opačném případě se nejprve spočítá složka s maximální absolutní hodnotou, pak se použijí rovnice (1)

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

Sférická lineární interpolace (slerp)

- **nejkratší sférický** oblouk mezi orientacemi **q** a **r**
- vstup:
  - dva kvaterniony **q** a **r**
  - $\mathbf{q} \cdot \mathbf{r} \geq 0$, jinak $-\mathbf{q}$
  - reálný parametr $0 \leq t \leq 1$
- interpolovaný kvaternion:
  - $slerp(\mathbf{q}, \mathbf{r}, t) = \mathbf{q}(\mathbf{q}^* \mathbf{r})^t$
  - jiné vyjádření:

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

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

Otočení mezi dvěma vektory

- dva směrové vektory **s** a **t**
  1. normalizace vektorů **s**, **t**
  2. jednotková rotační osa $\mathbf{u} = \frac{(\mathbf{s}\times\mathbf{t})}{\|\mathbf{s}\times\mathbf{t}\|}$
  3. vyjádření úhlu mezi **s** a **t**:

$$e = \mathbf{s} \cdot \mathbf{t} = cos(2\varphi)$$

$$\| \mathbf{s} \times \mathbf{t} \| = sin(2\varphi)$$

  4. výsledný kvaternion:

$$\mathbf{q} = (\mathbf{u} \cdot sin\varphi, cos\varphi)$$

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

Slerp rotačních matic

- vstup:
  - dvě rotační matice Q a R
  - reálný parametr $0 \leq t \leq 1$
- interpolovaná matice:
  - $slerp(Q, R, t) = Q(Q^T R)^t$
  - technický **problém**: obecná mocnina rotační matice
  - nutnost výpočtu osy a úhlu otočení $Q^T R$
  - výpočetně málo efektivní

Vyjádření rotace - shrnutí

- rotační matice
  - + HW podpora, efektivní transformace vektoru
  - - paměť (9× float), neefektivní ostatní operace
- osa a úhel otočení
  - + paměť (4× float nebo 6× float), podobné kvaternionu
  - - neefektivní kompozice, interpolace
- kvaternion
  - + paměť (4× float), kompozice, interpolace
  - - neefektivní transformace vektoru
- podrobnosti viz *RotationIssues.pdf*

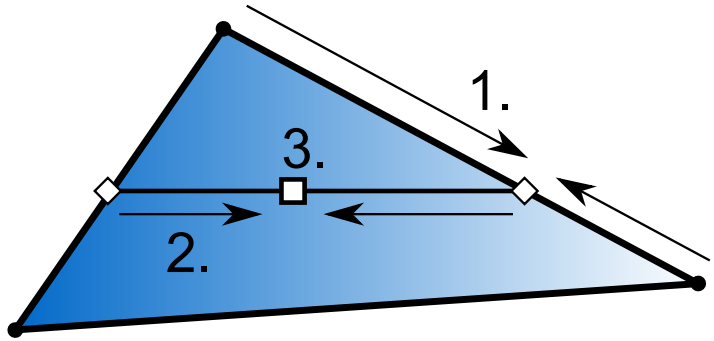
Interpolace na obrazovce

- týká se souřadnic od **ořezávacích** (včetně)
  - clip space: $[x, y, z, w]$
  - NDS: $[\frac{x}{w}, \frac{y}{w}, \frac{z}{w}]$ (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 $\frac{f}{n}$)
  - $+$ interpolaci hloubky z lze dělat na obrazovce **lineárně**
- **W-buffer** místo z-bufferu (dnes se moc nepoužívá)

Perspektivně korektní interpolace

1. ve straně trojúhelníka interpoluj $[x, y, z, \frac{1}{w}, \frac{u}{w}, \frac{v}{w}]$
2. interpoluj v rámci scanline
3. vypočítej správné parametry

$$\begin{aligned} w &= 1/\tfrac{1}{w} \\ u &= \tfrac{u}{w} \cdot w \\ &\dots \end{aligned}$$

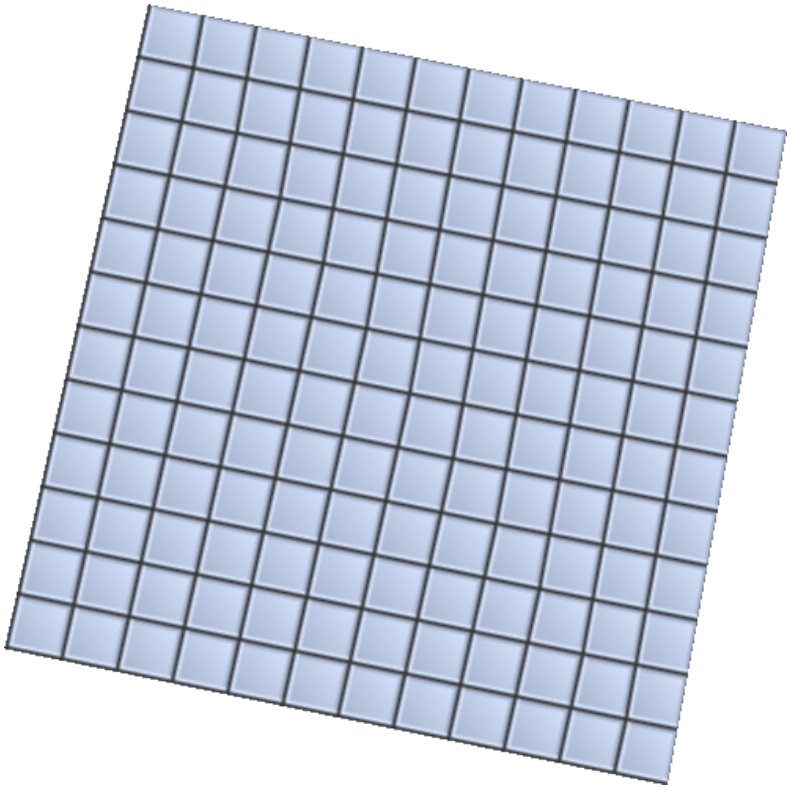


- snahou je používat co nejvíce **lineární interpolaci**
- hloubku **z** lze interpolovat lineárně
- pro **texturové souřadnice** nebo **w** se musí implementovat perspektivně korektní interpolace
- **lineárně** interpoluji $\frac{1}{w}$, $\frac{u}{w}$ i $\frac{v}{w}$, finální $[u, v]$ pak dopočítám dělením (hyperbolická interpolace)

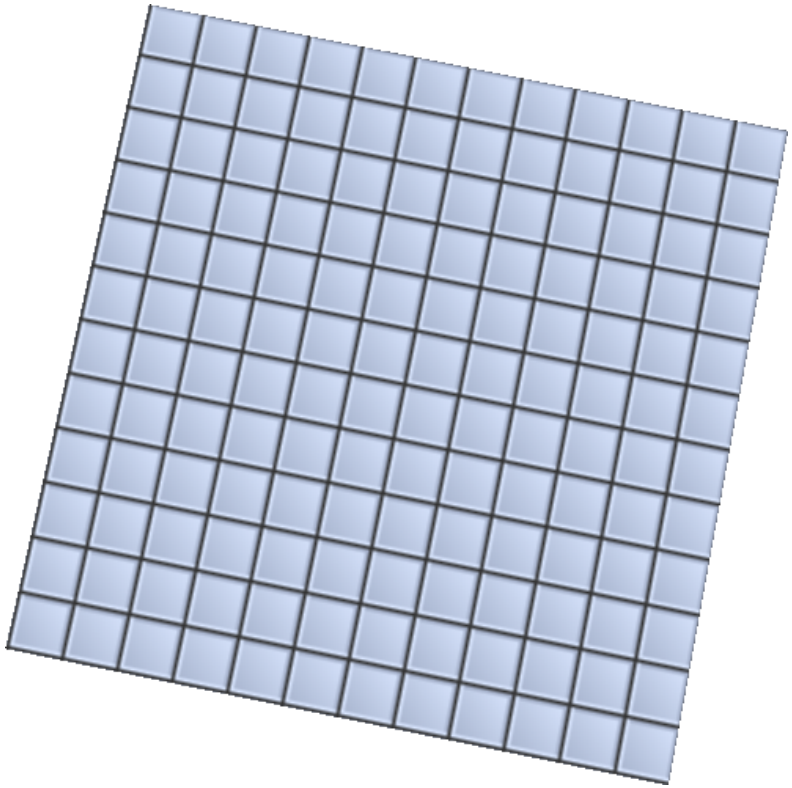
Perspektivně korektní interpolace - aplikace

- historicky výpočetně výrazně **náročnější** než lineární
- řešení - snížení náročnosti, perspektivně korektní interpolace počítána **jen v určitých bodech**, zbytek interpolován (bi)lineárně
  - Quake - jednou za 16 pixelů scanline
  - bilineární po blocích v souřadnicích obrazovky
  - pro konstantní z
- moderní karty mají **HW interpolátory**

Perspektivně korektní interpolace - příklad

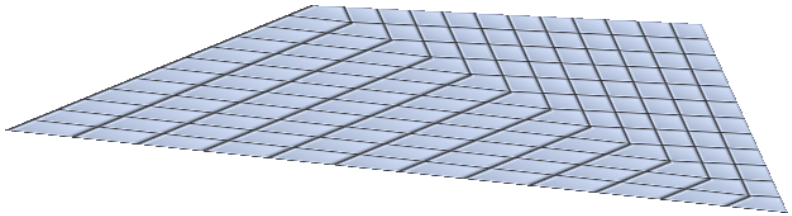


affinní

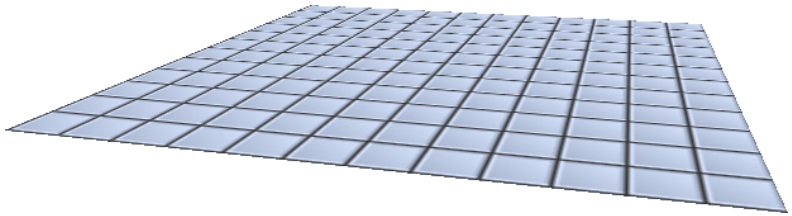


perspektivně správné

Perspektivně korektní interpolace - příklad 2



affinní



perspektivně správné

Literatura

- Tomas Akenine-Möler, Eric Haines: **Real-time rendering, 2nd edition**, A K Peters, 2002, ISBN:1568811829
- OpenGL Architecture Review Board: **OpenGL Programming Guide: The Official Guide to Learning OpenGL**, Addison-Wesley, nejnovější vydání (aktuálně 8. vydání pro OpenGL 4.1)
- David Eberly: **Rotation Representations and Performance Issues**, `http://www.geometrictools.com/Documentation/RotationIssues.pdf`, 2002-2008
- Wikipedia: `http://en.wikipedia.org/wiki/Rotation_matrix`, ...