Transformace (2)

Petr Felkel
Katedra počítačové grafiky a interakce, ČVUT FEL
místnost KN:E-413 (Karlovo náměstí, budova E)
E-mail: felkel@fel.cvut.cz

S použitím materiálů Bohuslava Hudce, Jaroslava Sloupa a Vlastimila Havrana

Poslední změna: 3.5.2016

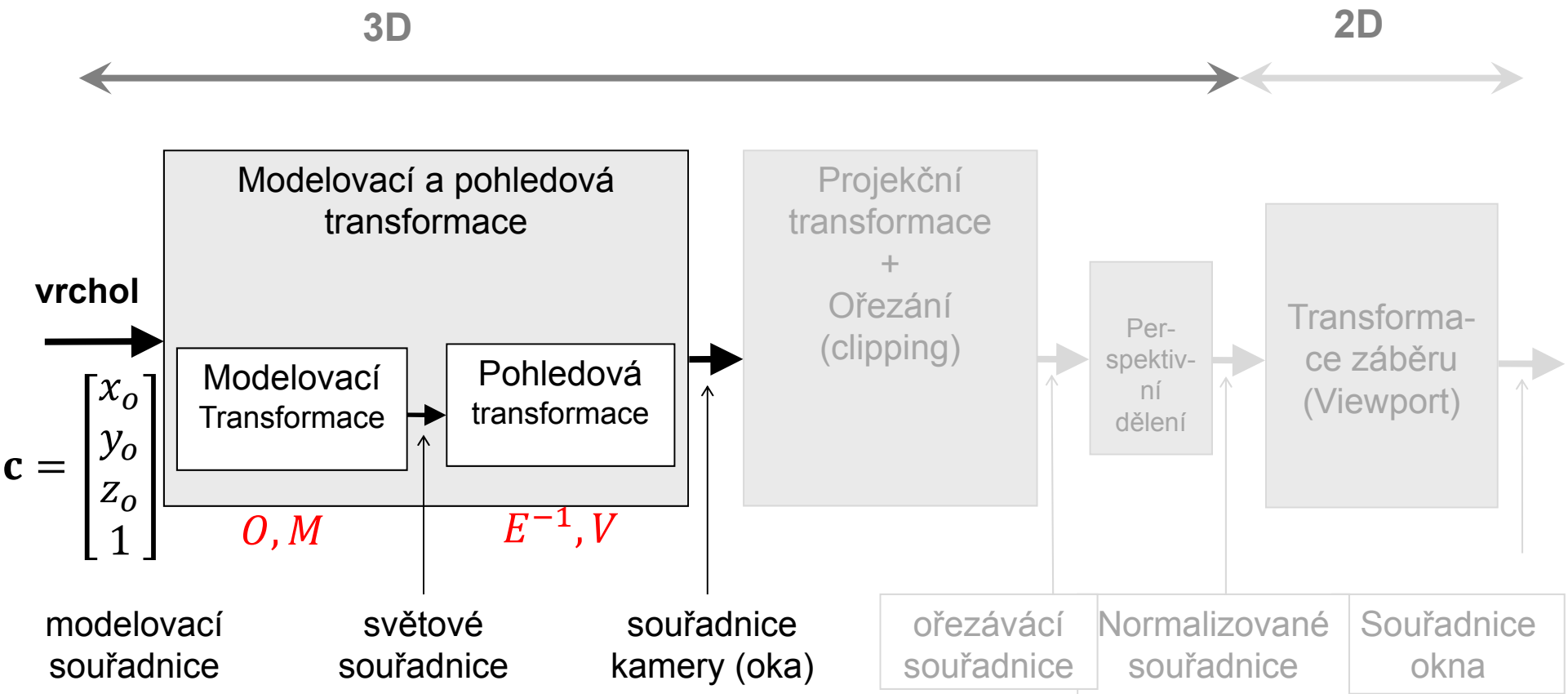
Osnova

Opakování transformací z minula

Transformace (2)

- Projekce a viewport
- Homogenní souřadnice
- Rotace podle Eulerových úhlů a Gimbal lock

Logické kroky při transformaci vrcholů



$$\vec{o}^t = \vec{w}^t O \qquad \vec{e}^t = \vec{w}^t E$$

$$\begin{aligned} \tilde{p} &= \vec{o}^t \mathbf{c} \\ &= \vec{w}^t O \mathbf{c} \\ &= \vec{e}^t E^{-1} O \mathbf{c} \end{aligned}$$ … maticově

Opakování: Souřadnicové soustavy v PG

- $\vec{\boldsymbol{w}}^t$ - daná, pozice objektů i kamery se definuje vůči ní
- $\vec{\boldsymbol{o}}^t = \vec{\boldsymbol{w}}^t O$ objekt někam umístěný do světa
  $\vec{\boldsymbol{o}}^t = \vec{\boldsymbol{w}}^t O_2$ druhý objekt umístěný jinam do světa
- O_1, O_2 modelové matice – matice přechodu z souřadnicové soustavy světa $\vec{\boldsymbol{w}}^t$ do lokální soustavy objektu
  (lokální = globální * O)
  Převádí lokální souřadnice modelu do globálních.
- $\vec{\boldsymbol{e}}^t = \vec{\boldsymbol{w}}^t E$ soustava kamery
- $E^{-1} = V$ pohledová matice

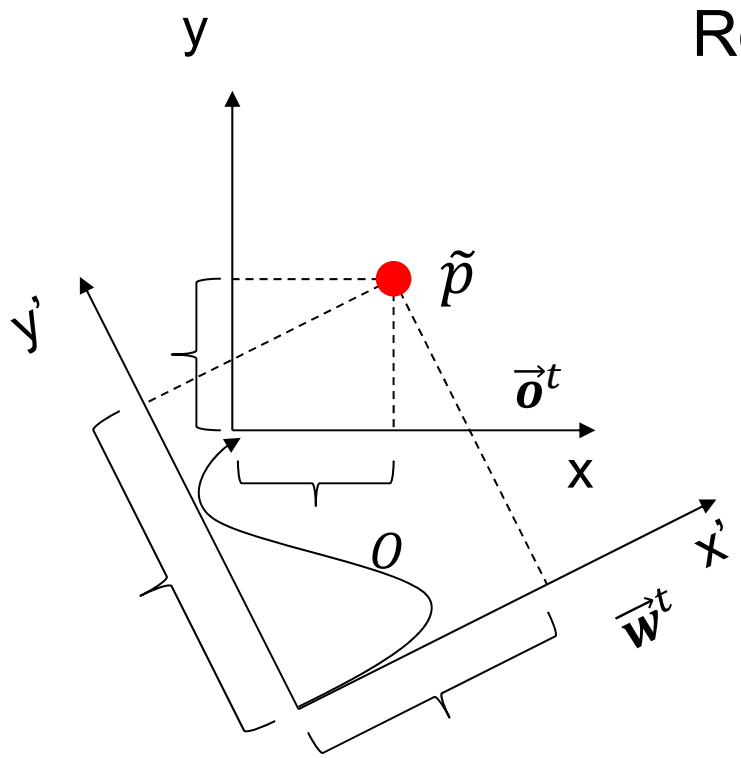
$$\vec{\boldsymbol{e}}^t = \vec{\boldsymbol{w}}^t E$$

$$\vec{\boldsymbol{w}}^t = \vec{\boldsymbol{e}}^t E^{-1}$$

$$\vec{\boldsymbol{o}}^t = \vec{\boldsymbol{w}}^t O = \vec{\boldsymbol{e}}^t E^{-1} O$$

Souřadnice bodu objektu v různých soustavách souřadnic

- Souřadnice bodu objektu v různých souř. soustavách:



Dvě souřadné soustavy,
lišící se počátky a směry os

Rovnost bází: $\vec{o}^t = \vec{w}^t O = \vec{e}^t E^{-1} O$

$$\vec{e}^t = \vec{w}^t E \rightarrow \vec{w}^t = \vec{e}^t E^{-1}$$

$$\tilde{p} = \vec{o}^t \mathbf{c}$$
$$= \vec{w}^t O \mathbf{c}$$

$$O = \begin{pmatrix} \downarrow & \downarrow & \downarrow & \begin{matrix} t_x \\ t_y \\ t_z \\ 1 \end{matrix} \end{pmatrix}$$

Průměty bázových vektorů $\vec{o}^t$ do bází $\vec{w}^t$

Transformace vzhledem k bázi

- $\vec{\boldsymbol{b}}^t \mathbf{c} \Rightarrow \vec{\boldsymbol{b}}^t M \mathbf{c}$ …. Transformace maticí M vzhledem k bázi $\vec{\boldsymbol{b}}^t$
- $\vec{\boldsymbol{o}}^t = \vec{\boldsymbol{w}}^t O = \vec{\boldsymbol{e}}^t E^{-1} O$

Transformace vzhledem:

k lokální soustavě objektu

ke světovým souřadnicím

k soustavě souřadnic kamery

Transformace vzhledem k bázi / soustavě souřadnic

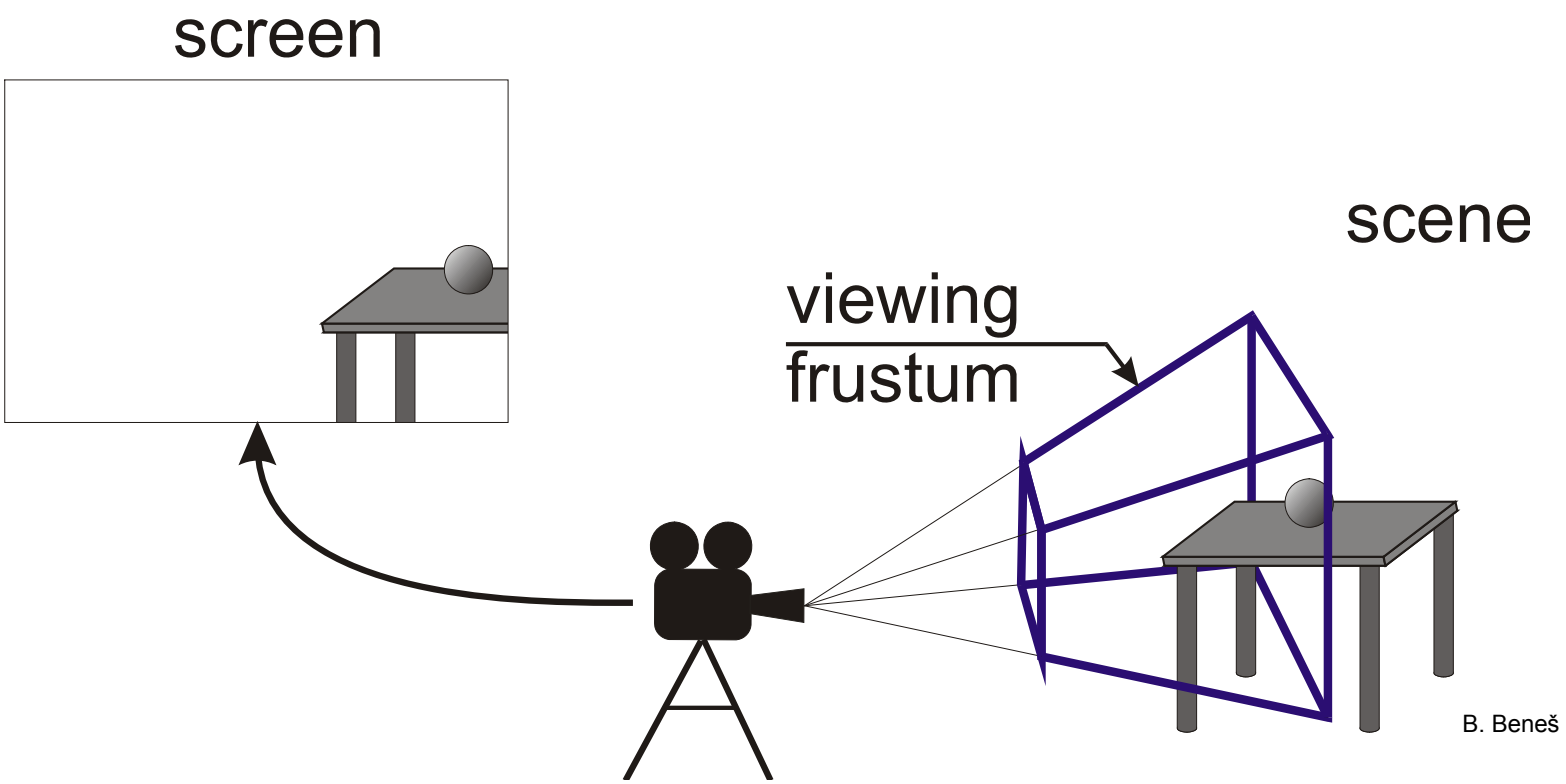
- V soustavě $\vec{\boldsymbol{f}}^t$: $\vec{\boldsymbol{f}}^t\mathbf{c} \Rightarrow \vec{\boldsymbol{f}}^t S\mathbf{c}$
- V soustavě $\vec{\boldsymbol{a}}^t$: $\vec{\boldsymbol{a}}^t\mathbf{d} \Rightarrow \vec{\boldsymbol{a}}^t S\mathbf{d}$
- Zpět vůči $\vec{\boldsymbol{f}}^t$: $\vec{\boldsymbol{f}}^t \overbrace{AS}\overbrace{A^{-1}\mathbf{c}}$,
-

$$\vec{\boldsymbol{a}}^t = \vec{\boldsymbol{f}}^t A$$

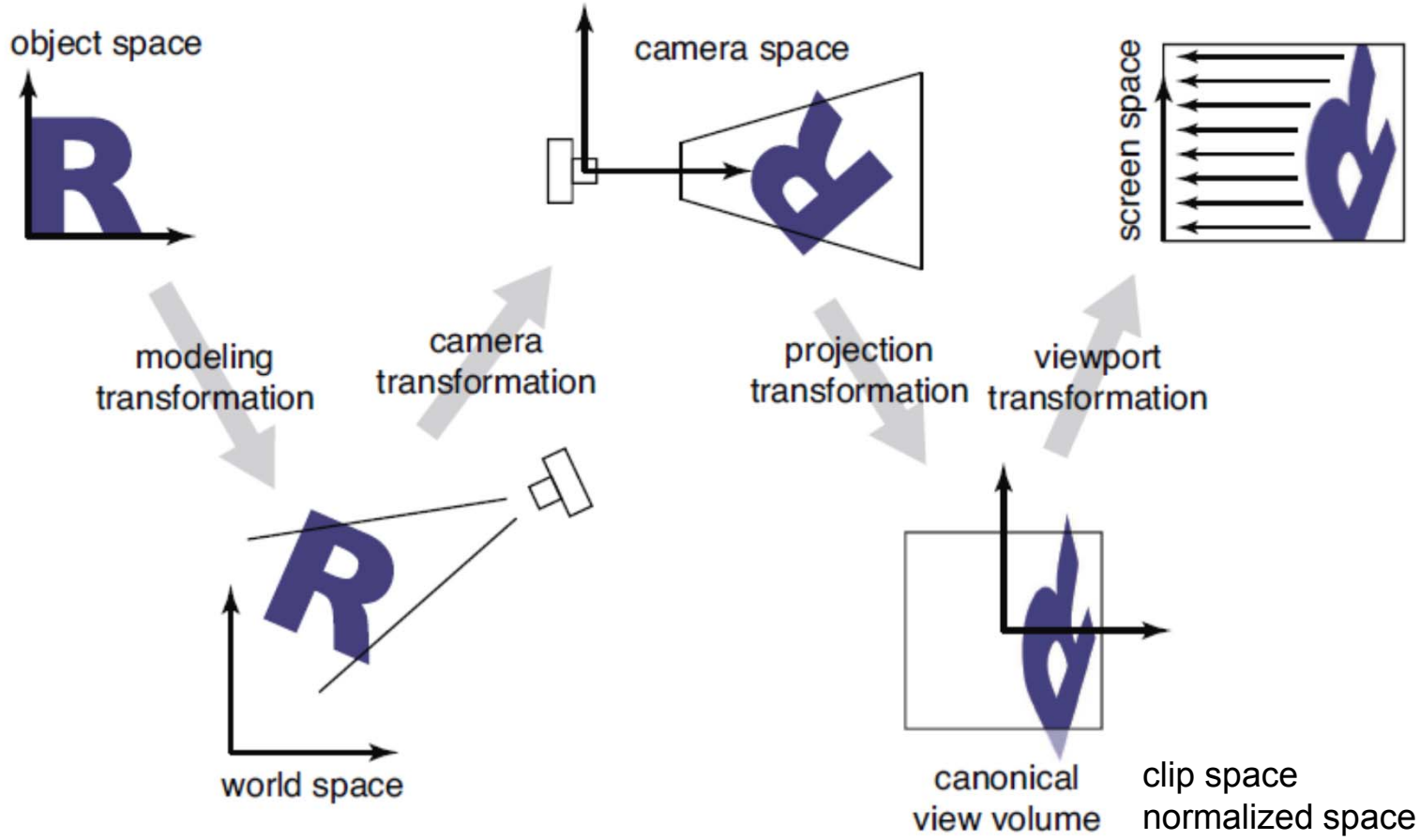
$$\vec{\boldsymbol{a}}^t A^{-1} = \vec{\boldsymbol{f}}^t$$

$$\mathbf{d} = A^{-1}\mathbf{c}$$

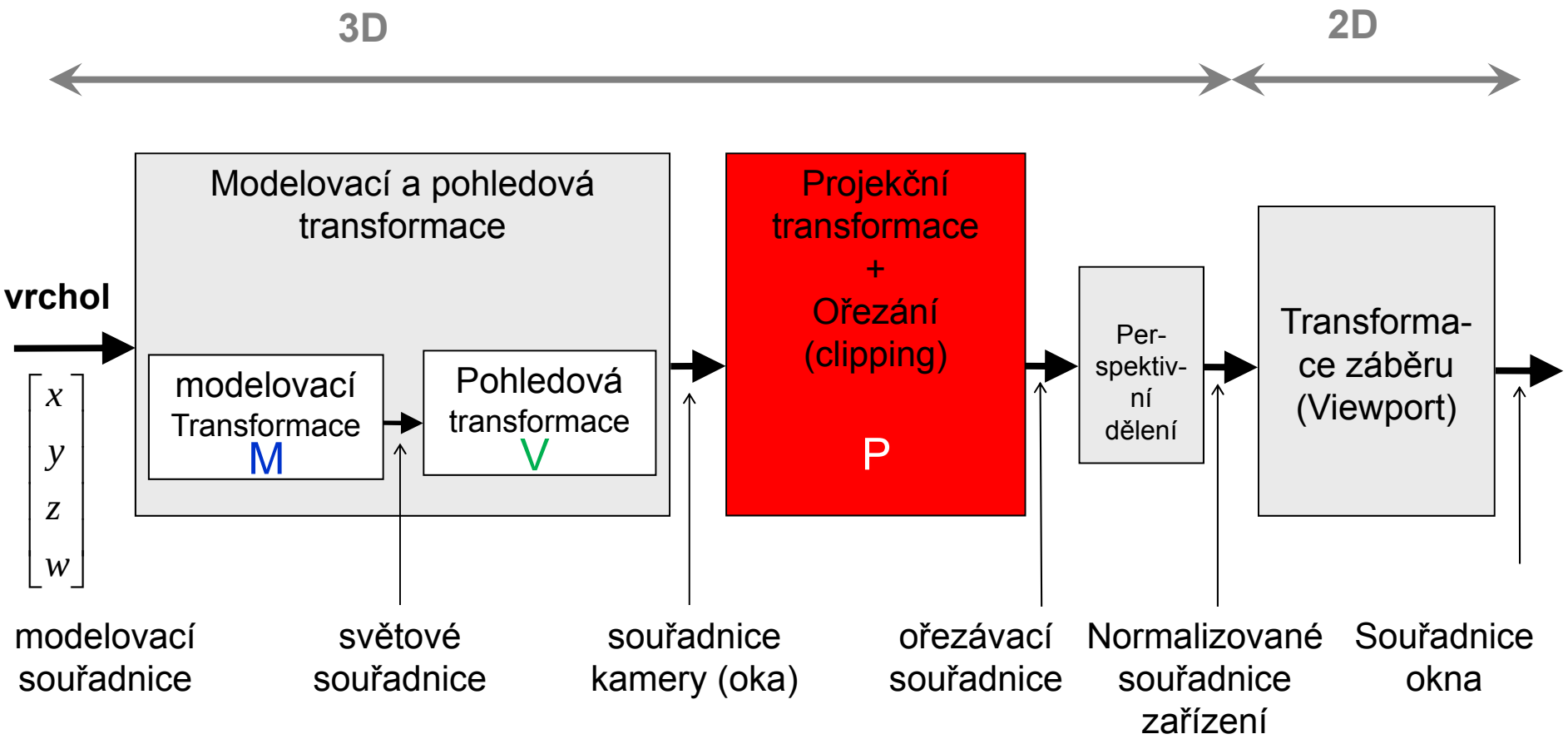
Analogie s fotoaparátem



Vizualizace transformací - zopakování



Projekční transformace

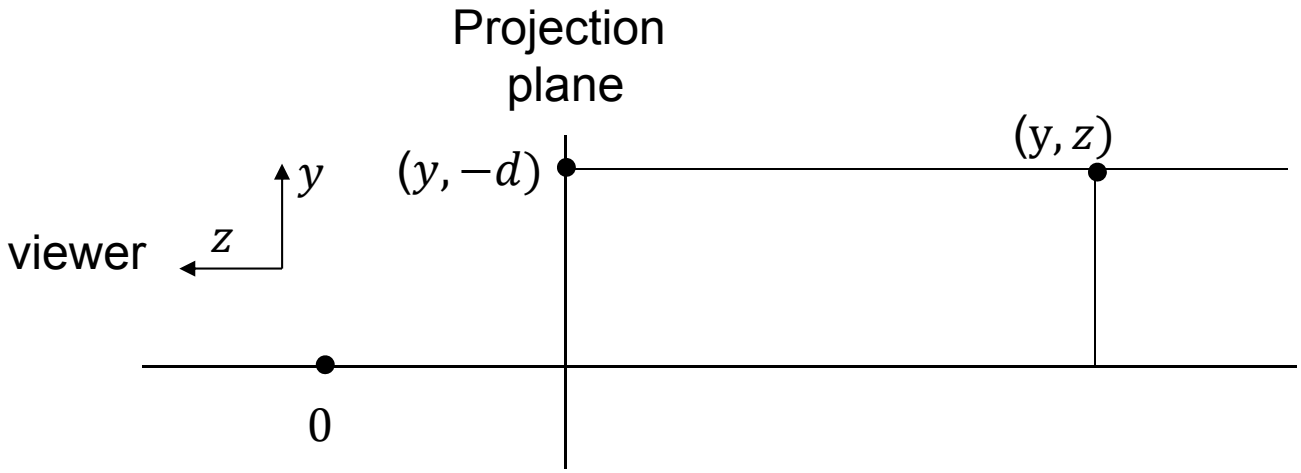


Projekční transformace - P

- definuje tvar pohledového objemu (**viewing volume, frustrum)**
- pohledový objem „komolý jehlan“
  - určuje, jak se objekt pomítá na průmětnu (*perspektivní* či *paralelní* projekce) – VS
  - definuje polohu ořezávacích rovin, tj., které objekty či jejich části budou oříznuty (*clipping planes*) – fixní část
  - zadává se v souřadné soustavě kamery

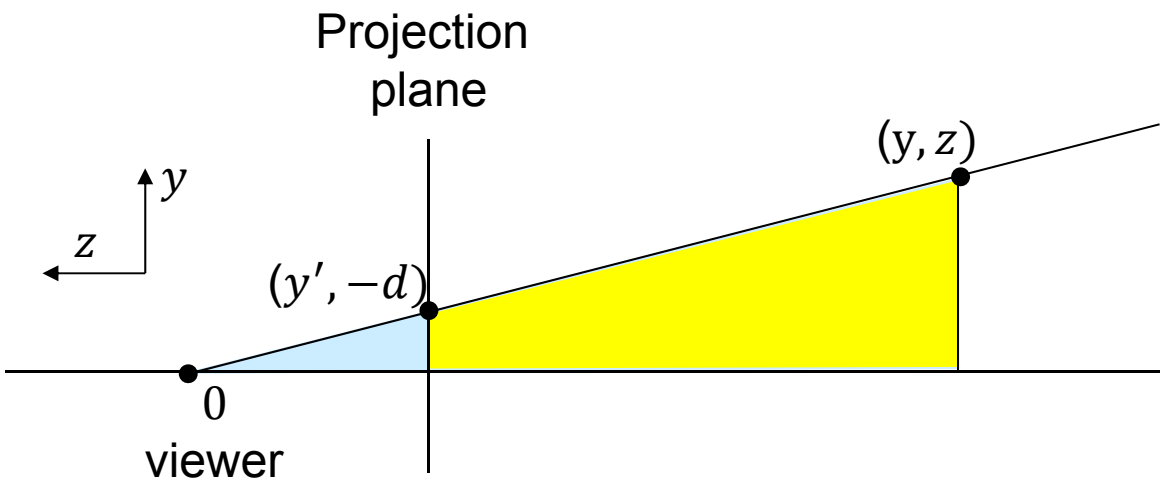
- OpenGL umožňuje jakoukoliv projekci definovanou uživatelem
- projekce obecně NENÍ afinní transformací (projekční matice nemá poslední řádek ve tvaru 0 0 0 1)
- nutné jsou funkce vytvářející matice pro
  - ortografickou projekci (paralelní)
  - perspektivní projekci

Základ paralelního zobrazení (ortho)



$$x' = x$$
$$y' = y$$
$$z' = -d$$ = near plane z

Základ perspektivního zobrazení



Similar triangles:

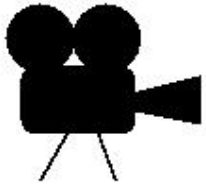
$$\frac{y'}{d} = \frac{y}{-z}$$

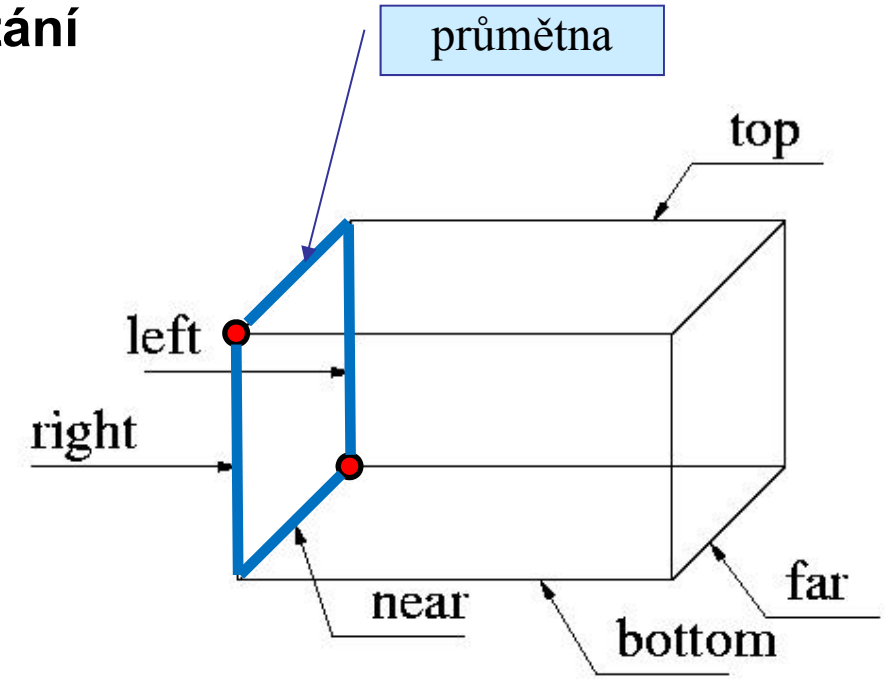
$$y' = -d\,\frac{y}{z}$$

Paralelní projekce

```
mat4 glm::ortho(  float left, float right,
                  float bottom, float top,
                  float near, float far);
```

- vytvoří matici pro **paralelní promítání**
- pohledový objem je kvádr
- [left, bottom, *] a [right, top, *] = body na blízké a vzdálené ořezávací rovině (* = near a far),
- jsou mapovány do dolního levého a pravého horního rohu formátu (viewport)

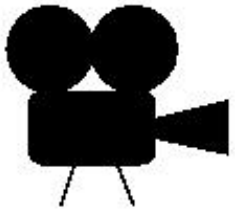


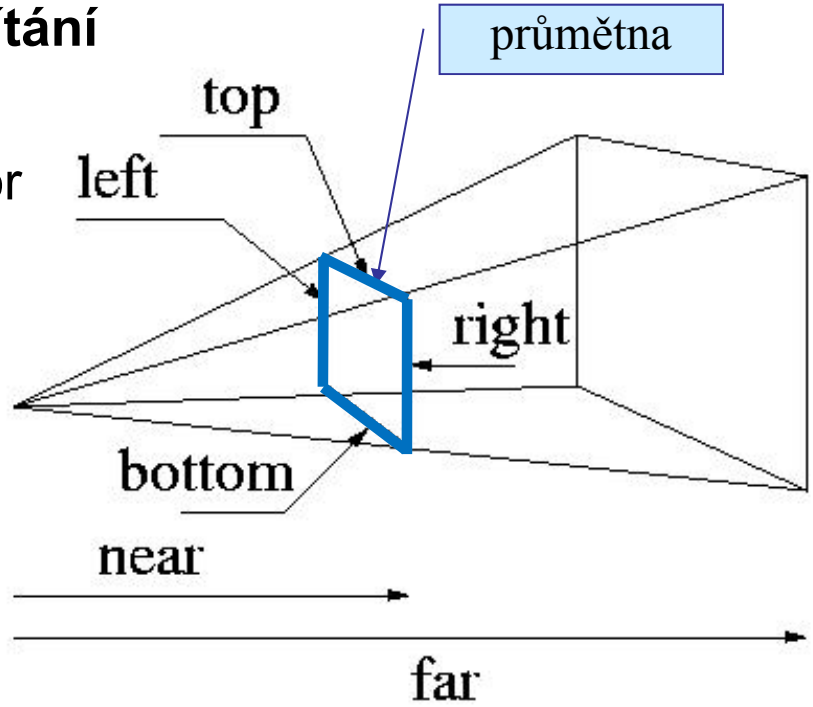


Perspektivní projekce

```
mat4 glm::frustum( float left, float right,
                   float bottom, float top,
                   float near, float far);
```

- vytvoří matici pro **perspektivní promítání**
- pohledový objem je komolý jehlan
- podstavy rovnoběžné, kolmé na vektor pohledu
- menší podstava = průmětna
- objekty blíže zvětšené (zaberou relativně větší část pohledového objemu)

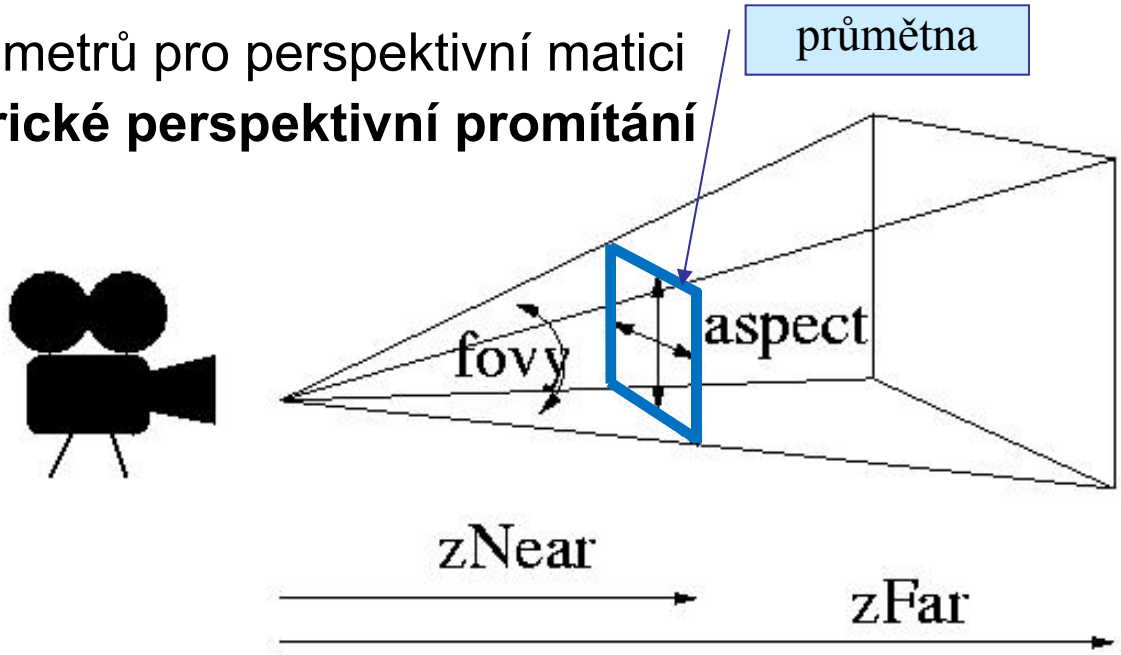




Perspektivní projekce

```
mat4 glm::perspective ( float fovy, float aspect,
                        float near, float far );
```

- jiný způsob definice parametrů pro perspektivní matici
- vytvoří matici pro **symetrické perspektivní promítání**
- **fovy** = úhel záběru ve směru y,
  - rozsah $\langle 0.0, \pi \rangle$
  - spolu s near určí h
- **aspect** je poměr šířky ku výšce pohledového objemu (w / h)
  w = aspect * h;
- hodnoty **near** a **far** musí být kladné (near > 0 !!!)



Homogenní souřadnice

- bod je reprezentován svými souřadnicemi $\mathbf{P} = [x, y, z]^t$
  v poč. grafice používáme **homogenní souřadnice** $\mathbf{P} = [x, y, z, w]^t$

- Kartézské souřadnice $\Rightarrow$ homogenní souřadnice

  $\mathbf{P} = [x, y, z]^t \;\Rightarrow\;$ zvolit $w \neq 0 \;\Rightarrow\; \mathbf{P} = [w.x, w.y, w.z, w]^t$

  příklad: bod v kartézských souřadnicích $[2, 3, 5]^t$.
  Jaké jsou jeho homogenní souřadnice?

  Pro body volíme w = 1

  $[w.2, w.3, w.5, w]^t$ a $w \neq 0 \Rightarrow$ např. $[2, 3, 5, 1]^t$, $[4, 6, 10, 2]^t$, atd.

- homogenní souřadnice $\Rightarrow$ Kartézské souřadnice

  $\mathbf{P} = [x, y, z, w]^t \;\Rightarrow\; \mathbf{P} = [x/w, y/w, z/w]^t \qquad w \neq 0, \quad w \in R$

  příklad : bod v homogenních souřadnicích $[9, 3, 12, 3]^t$.
  Jaké jsou jeho kartézské souřadnice?

  $\mathbf{P} = [9/3, 3/3, 12/3]^t = [3, 1, 4]^t$

Homogenní souřadnice ve 2D

Geometrická interpretace ve 2D:

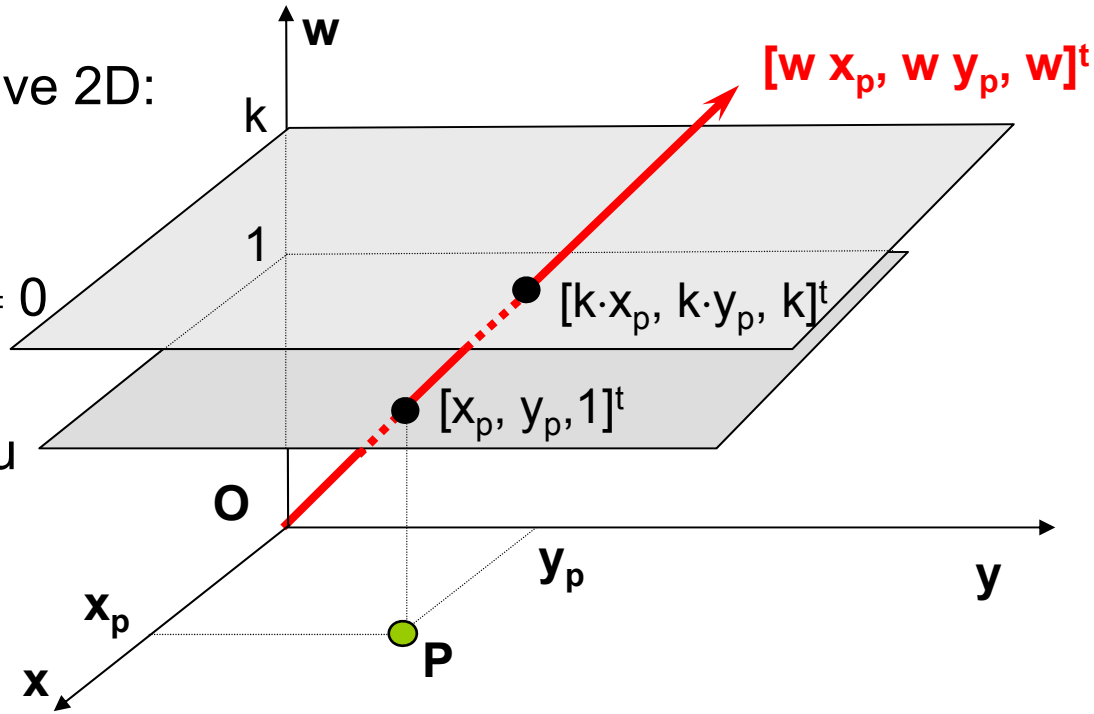
bodům s homogenními souřadnicemi

P = $[w.x, w.y, w.z, w]^t$, $w \neq 0$

Tvoří přímku bez počátku,

která je celá obrazem bodu

P = $[x, y, z]^t$



Výhody homogenních souřadnic

- Afinní transformace i projekce lze zapsat jednou maticí (ve 3D maticí 4x4)
- Skládání transformací jako násobení matic
- Kompaktní reprezentace bodů ($w \neq 0$) a vektorů ($w = 0$)

Pro body volíme w = 1
Pro vektory je w = 0

Matrices and transformations

$$\begin{pmatrix} a & b & c & 0 \\ e & f & g & 0 \\ i & j & k & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Linear transformation matrix

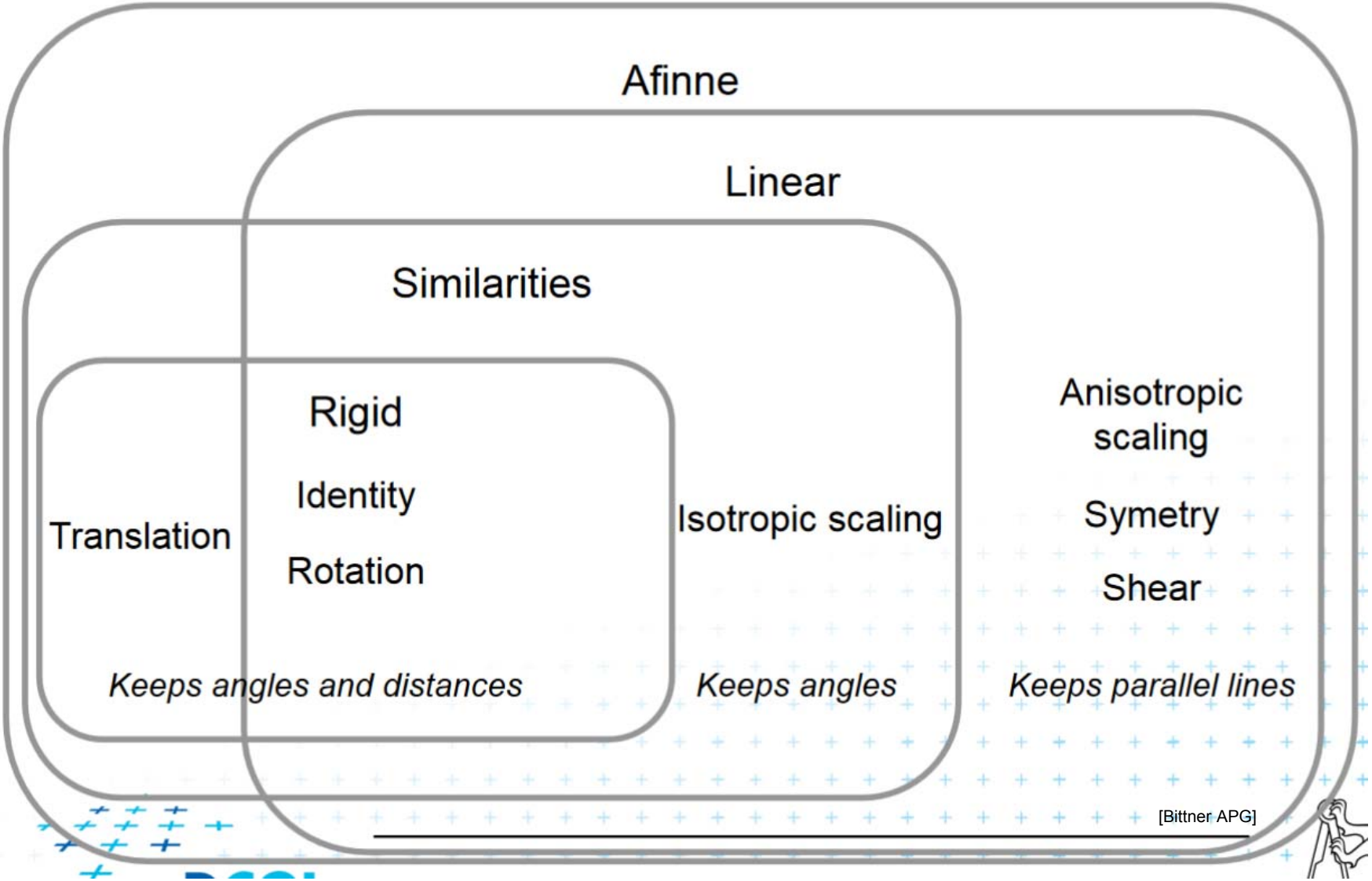
$$\begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Affine transformation matrix

$$\begin{pmatrix} a & 0 & c & 0 \\ 0 & f & g & 0 \\ 0 & 0 & -k & -l \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Perspective transformation matrix

Transformations

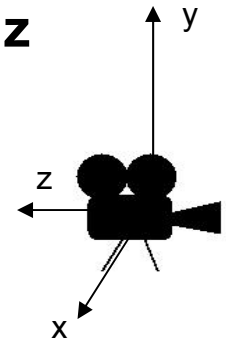


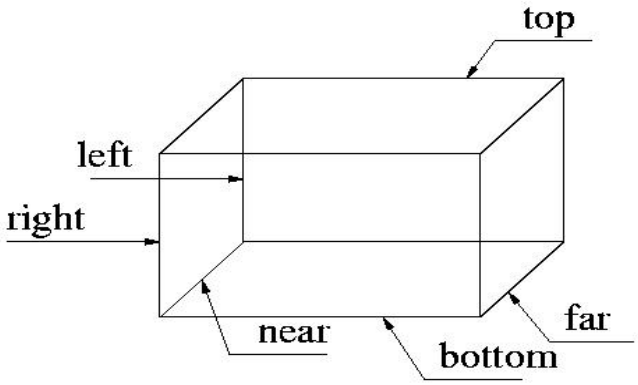
[Bittner APG]

Odvození paralelní projekční matice (Ortho)

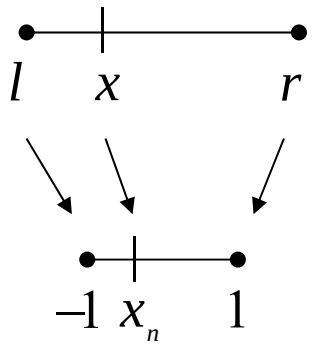
Princip: Zachová x a y. Ignoruje z

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \boxed{0} & d \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \begin{aligned} x' &= x \\ y' &= y \\ z' &= d \end{aligned}$$





Plus normalizace souřadnic: intervaly $\langle l, r\rangle, \langle t, b\rangle, \langle -n, -f\rangle \Rightarrow \langle -1, 1\rangle^3$



$$\frac{x-l}{r-l} = \frac{x_n-(-1)}{1-(-1)} = \frac{x_n+1}{2}$$

$$x_n = 2\frac{x-l}{r-l} - 1$$

$$x_n = \frac{2x}{r-l} - \frac{r-l+2l}{r-l}$$

$$x_n = \frac{2}{r-l}x - \frac{r+l}{r-l}$$

$$y_n = \frac{2}{t-b}y - \frac{t+b}{t-b}$$

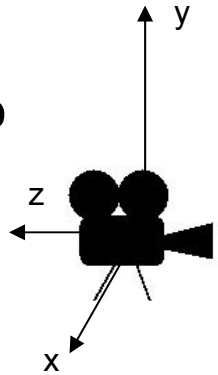
$$z_n = -1$$

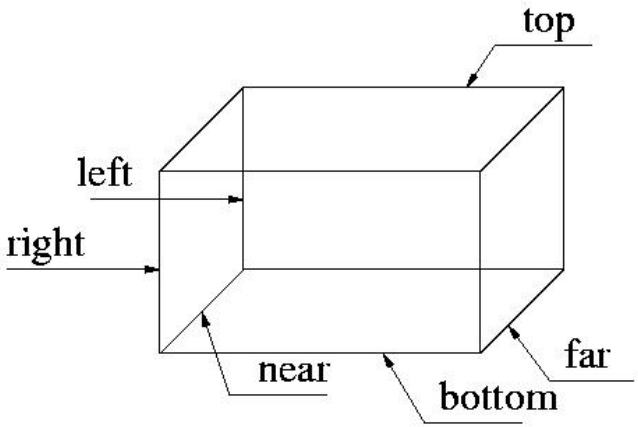
Hloubku ale potřebujeme na určení viditelnosti

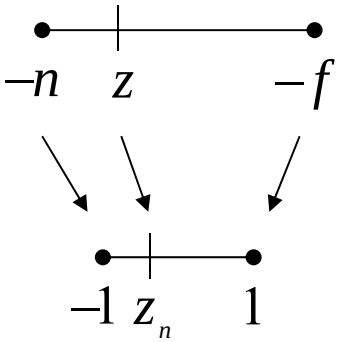
Odvození paralelní projekční matice (Ortho)

Odvození hloubky z_n

- near a far se zapisují v ortho kladně (vzdálenost od oka),
- jsou ale na záporné ose *z*,
- proto ve vzorcích záporné







$$\frac{z-(-n)}{-f-(-n)} = \frac{z_n-(-1)}{1-(-1)}$$

$$-\frac{z+n}{f-n} = \frac{z_n+1}{2}$$

$$z_n = -2\frac{z+n}{f-n} - \frac{f-n}{f-n}$$

$$z_n = \frac{-2z}{f-n} - \frac{f-n+2n}{f-n}$$

$$x_n = \frac{2}{r-l}x - \frac{r+l}{r-l}$$

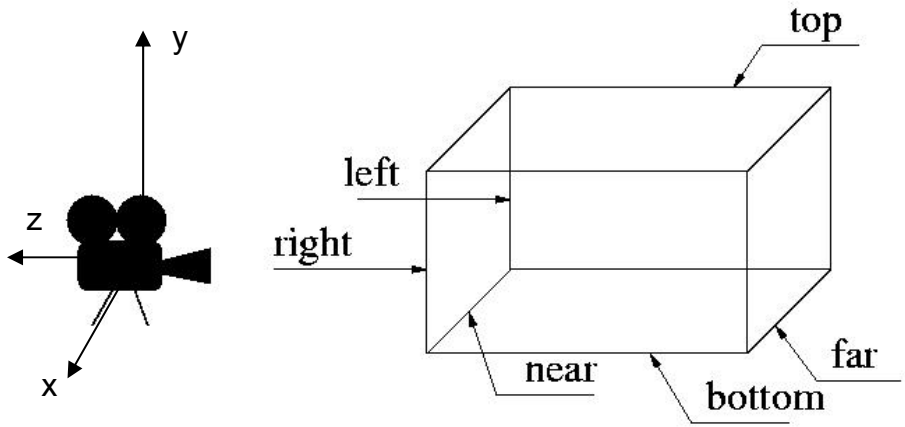
$$y_n = \frac{2}{t-b}y - \frac{t+b}{t-b}$$

$$z_n = \frac{-2}{f-n}z - \frac{f+n}{f-n}$$

Platí, pokud: $l \neq r,\ t \neq b$ a $n \neq f$

Odvození paralelní projekční matice (Ortho)

Výsledná matice



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

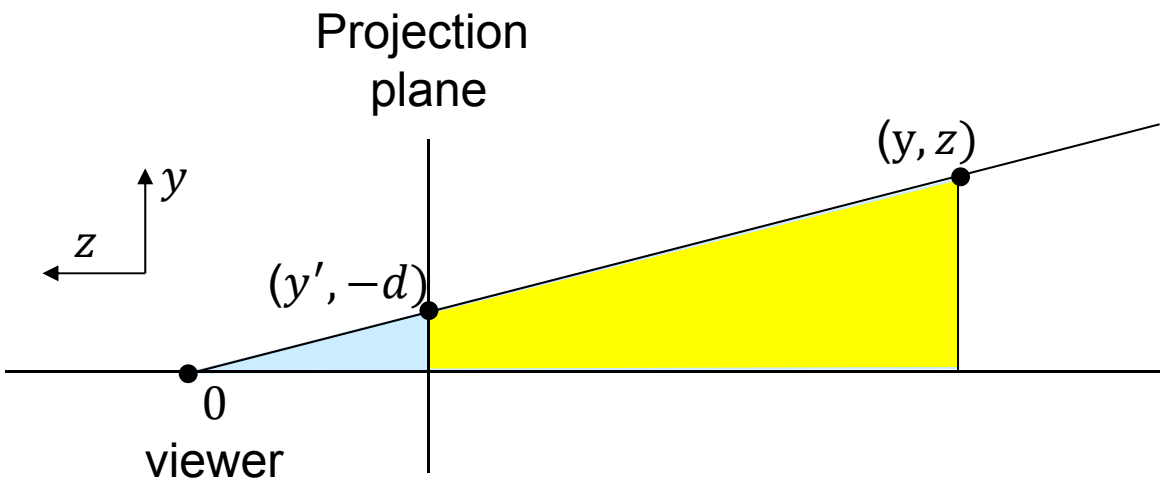
$$x_n = \frac{2}{r-l}x - \frac{r+l}{r-l}$$

$$y_n = \frac{2}{t-b}y - \frac{t+b}{t-b}$$

$$z_n = \frac{-2}{f-n}z - \frac{f+n}{f-n}$$

Platí, pokud : $l \neq r,\ t \neq b$ a $n \neq f$

Základ perspektivního zobrazení



Similar triangles:

$$\frac{y'}{d} = \frac{y}{-z}$$

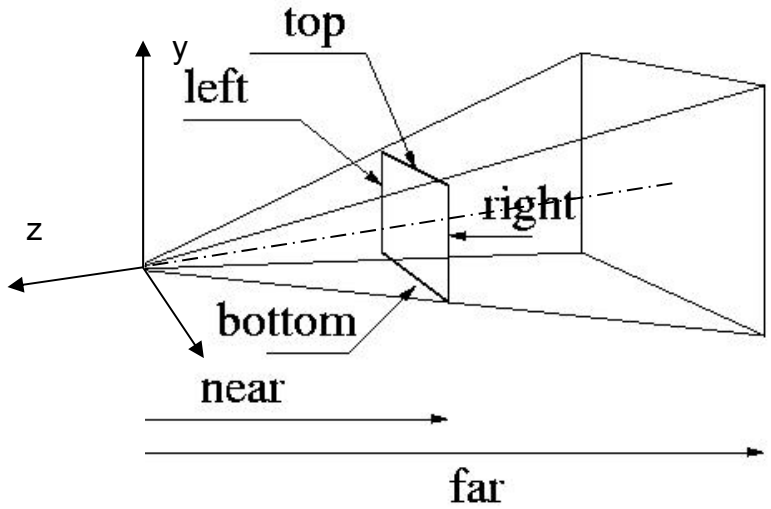
$$y' = -d\frac{y}{z} \qquad x' = -d\frac{x}{z}$$

Odvození perspektivní projekční matice (Frustum)

Odvození X a Y

- Dosadíme $d = -n$

$$x' = -n\frac{x}{z} \qquad y' = -n\frac{y}{z}$$



Plus normalizace souřadnic:

$$x_n = \frac{2}{r-l}x - \frac{r+l}{r-l} \qquad x_n = \frac{2n}{r-l}\frac{-1}{z}x - \frac{r+l}{r-l} \Big/ .(-z) \qquad -zx_n = \frac{2n}{r-l}x + \frac{r+l}{r-l}z$$

$$y_n = \frac{2}{t-b}y - \frac{t+b}{t-b} \qquad y_n = \frac{2n}{t-b}\frac{-1}{z}y - \frac{t+b}{t-b} \Big/ .(-z) \qquad -zy_n = \frac{2n}{t-b}y + \frac{t+b}{t-b}z$$

Odvození dále: $$z_n = \frac{-2nf}{f-n}\frac{1}{(-z)} + \frac{f+n}{f-n} \Big/ (-z) \qquad -zz_n = -\frac{f+n}{f-n}z - \frac{2nf}{f-n}$$

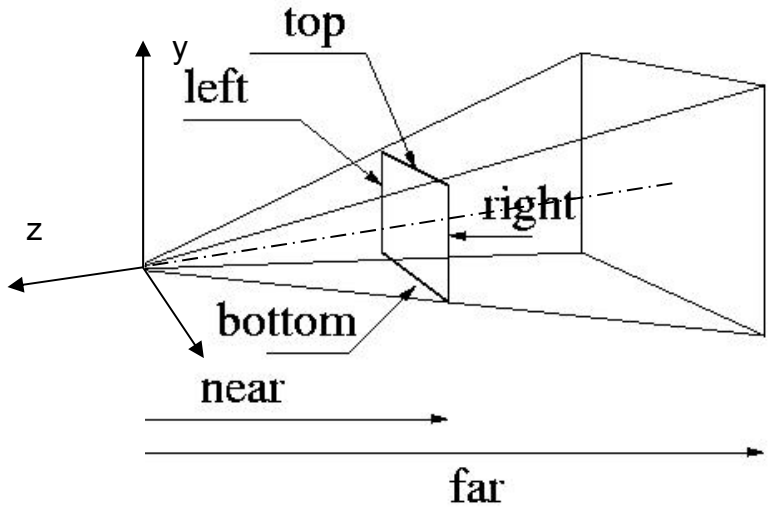
$$w = -z$$

Odvození perspektivní projekční matice (Frustum)

Mapování hloubky

- Interpoluje se 1/z
- Proto ve tvaru

$$z_n = \frac{A}{z} + B$$



Plus normalizace souřadnice *z*

- Dosadíme -n → -1, -f → 1

$$-1 = \frac{A}{-n} + B \quad \text{a} \quad 1 = \frac{A}{-f} + B$$

$$A = \frac{2nf}{f-n} \qquad B = \frac{f+n}{f-n}$$

Odvození A a B na dalším slajdu

$$z_n = \frac{-2nf}{f-n}\frac{1}{(-z)} + \frac{f+n}{f-n}$$

$$/ .(-z)$$

$$-zz_n = -\frac{f+n}{f-n}z - \frac{2nf}{f-n}$$

Odvození perspektivní projekční matice (Frustum)

Odvození A a B: Dosadíme -n → -1, -f → 1

$$z_n = \frac{A}{z} + B$$

$$-1 = \frac{A}{-n} + B \quad /-1 \quad \text{a} \quad 1 = \frac{A}{-f} + B$$

$$1 = \frac{A}{n} - B$$

$$1 = \frac{A}{-f} + B$$

$$2 = \frac{A}{n} + \frac{-A}{f} = A\frac{f-n}{nf}$$

$$A = \frac{2nf}{f-n}$$

$$B = \frac{A}{n} - 1$$

$$A = \frac{2nf}{f-n} \Rightarrow B = \frac{\frac{2nf}{f-n}}{n} - 1 = \frac{2f}{f-n} - 1$$

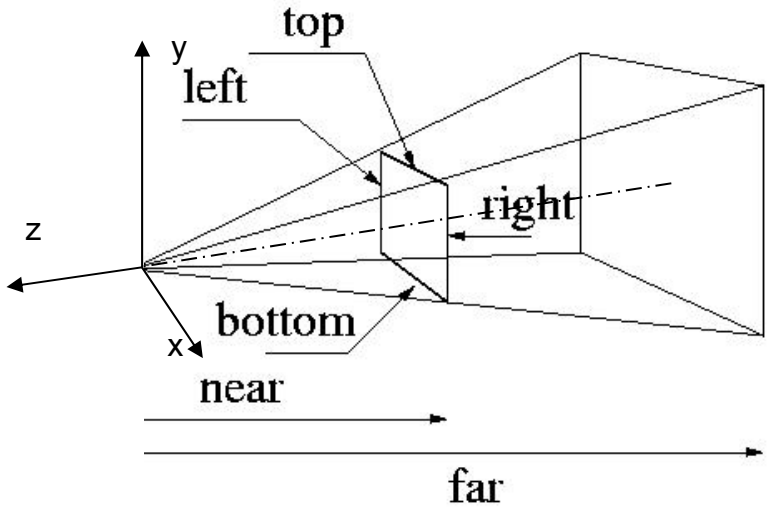
$$B = \frac{2f}{f-n} - \frac{f-n}{f-n}$$

$$B = \frac{f+n}{f-n}$$

Odvození perspektivní projekční matice (Frustum)

Výsledná matice

Platí pokud : $l \neq r,\ t \neq b$ a $n \neq f$



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

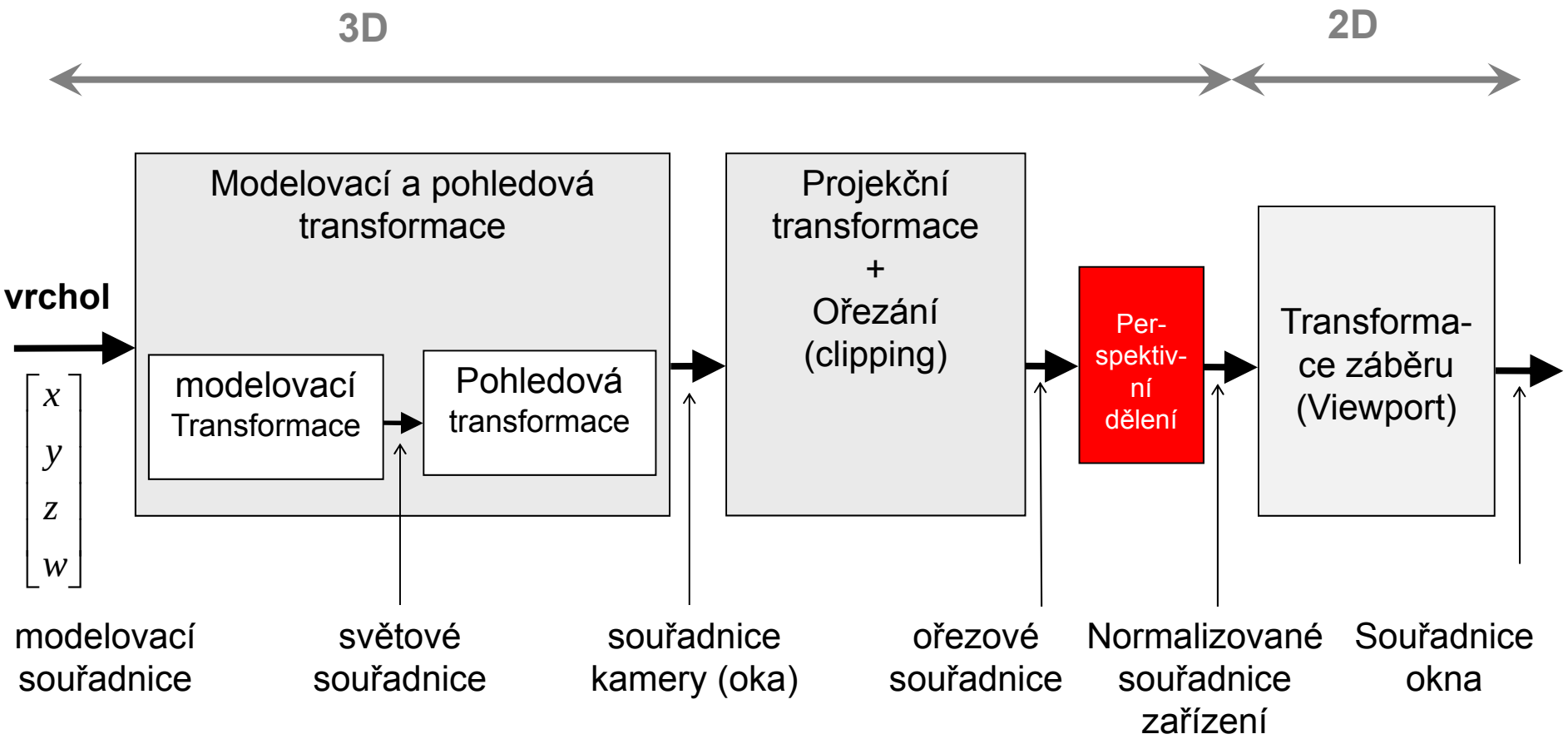
$$-zx_n = \frac{2n}{r-l}x + \frac{r+l}{r-l}z$$

$$-zy_n = \frac{2n}{t-b}y + \frac{t+b}{t-b}z$$

$$-zz_n = -\frac{f+n}{f-n}z - \frac{2nf}{f-n}$$

$$w = -z$$

Perspektivní dělení – 1/*w*



Division by w

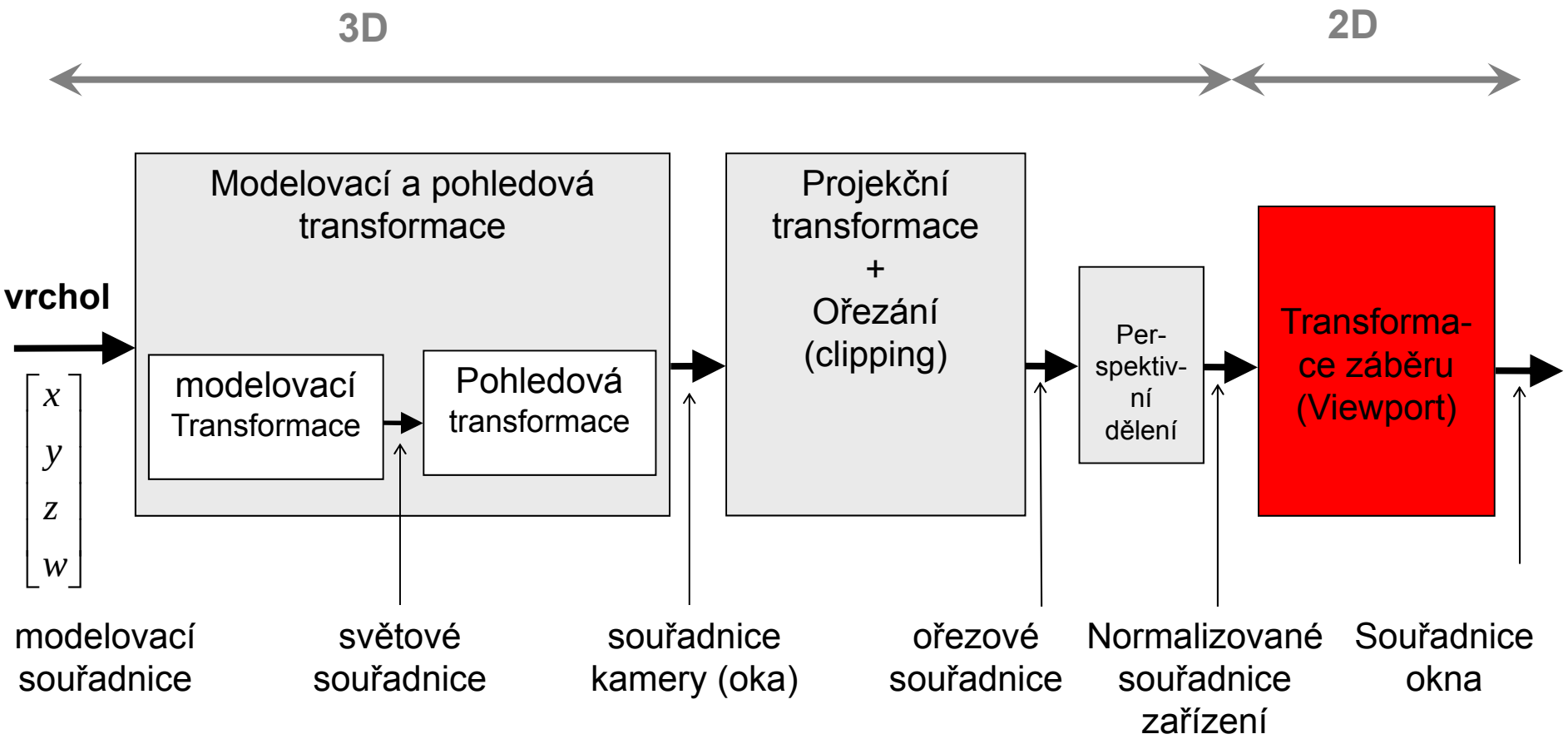
Matrix multiplication (from eye to clip coordinates)

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{bmatrix} x_e \\ y_e \\ z_e \\ w_e \end{bmatrix}$$

Division by w_c (from clip to normalized coordinates) = division by z_e

$$\begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} = \begin{bmatrix} x_c/w_c \\ y_c/w_c \\ z_c/w_c \end{bmatrix}$$

Transformace záběru



Transformace pracoviště (Viewport)

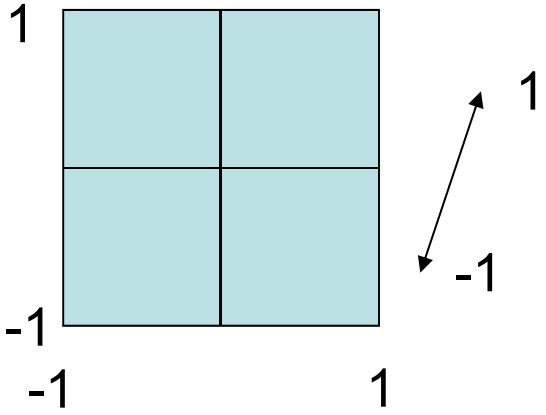
- Formát (viewport) = obdélníková oblast okna, do které se mapuje průmětna (= promítací rovina)

```
void glViewport(
        GLint x, GLint y, GLsizei width, GLsizei height);
```

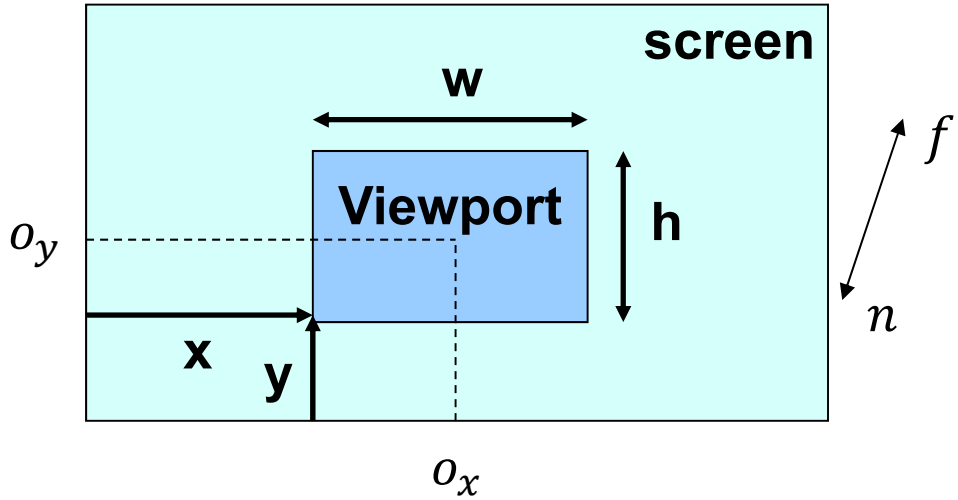
- [**x, y**] je levý dolní roh formátu (viewport),
- **width** a **height** je šířka a výška formátu …v souřadnicích okna – v pixelech
- pro knihovnu GLUT se nastavuje v callback funkci **reshape**
- poměr stran formátu by měl být stejný jako poměr stran průmětny
- jinak zkreslení obrazu
- GLUT: implicitně na celé okno (0,0,w, h)

Transformace pracoviště (Viewport)

Normalizované souřadnice zařízení $[x_d \quad y_d \quad z_d]^{\mathrm{t}}$



Souřadnice na obrazovce $[x_w \quad y_w \quad z_w]^{\mathrm{t}}$



$$x_w = \frac{w}{2} x_d + o_x$$

$$y_w = \frac{h}{2} y_d + o_y$$

$$z_w = \frac{f-n}{2} z_d + \frac{f+n}{2}$$

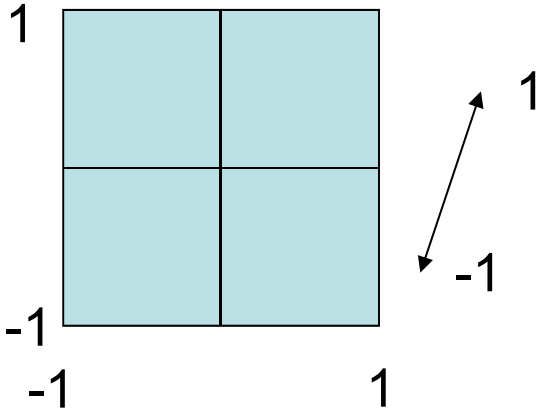
$$o_x = x + \frac{w}{2}$$

$$o_y = y + \frac{h}{2}$$

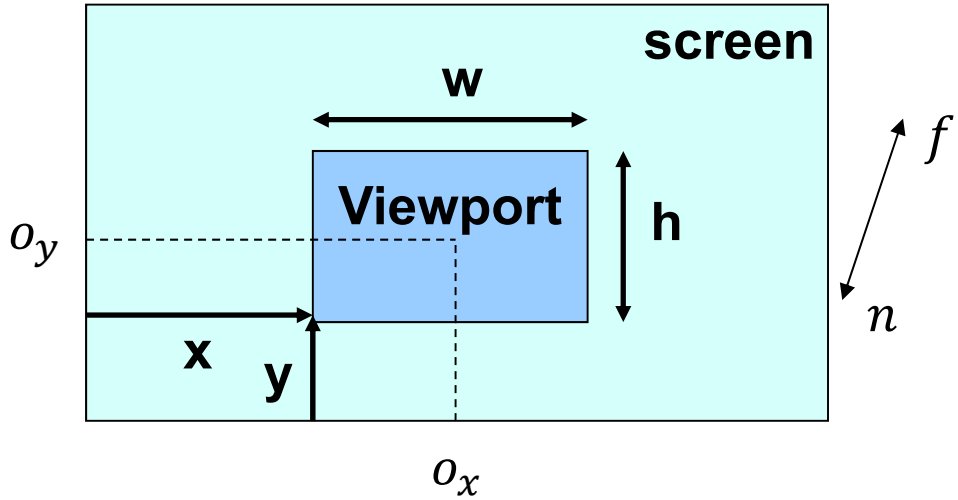
z_w na viditelnost (Z-buffer)

Transformace pracoviště (maticově)

Normalizované souřadnice zařízení $[x_d \quad y_d \quad z_d]^{\mathrm{t}}$



Souřadnice na obrazovce $[x_w \quad y_w \quad z_w]^{\mathrm{t}}$



$$\begin{pmatrix} \frac{w}{2} & 0 & 0 & x+\frac{w}{2} \\ 0 & \frac{h}{2} & 0 & y+\frac{h}{2} \\ 0 & 0 & \frac{f-n}{2} & \frac{f+n}{2} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$o_x = x + \frac{w}{2}$$

$$o_y = y + \frac{h}{2}$$

z_w na viditelnost (Z-buffer)

Transformace pracoviště (Viewport)

- Dotaz na aktuální formát (transformaci pracoviště)

```
int format[4];
glGetIntegerv( GL_VIEWPORT, format);
```

Význam: x, y, w, h

```
x = format[0],
y = format[1],
w = format[2],
h = format[3],
```

Zachování poměru stran obrazu

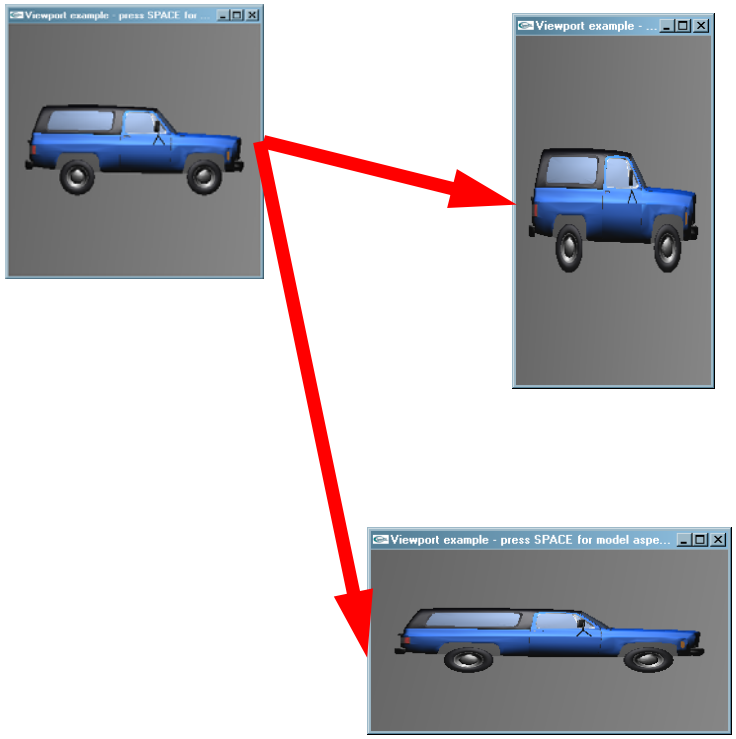
- … aby zůstala kružnice kulatá

- v proceduře reshape(int w, int h) registrována jako Callback …glutReshapeFunc(reshape);

- Dva způsoby

  a) zmenšením formátu

  b) protažením projekce

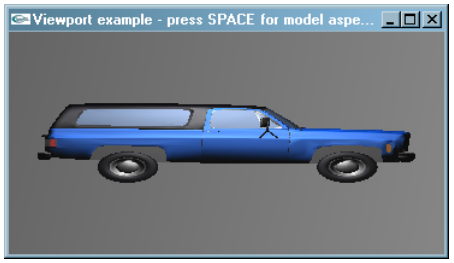


Zachování poměru stran obrazu

a) zmenšením formátu - zbude nevyužitá plocha okna

```
if( w > h )  // okno široké
    glViewport((w-h)/2, 0, h, h);
else         // okno vysoké
    glViewport(0, (h-w)/2, w, w);
```

zkreslené

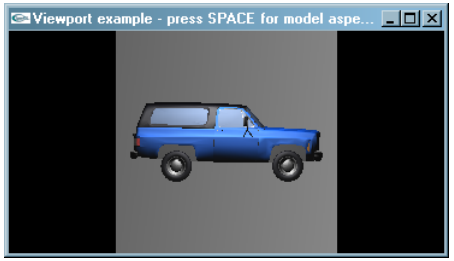


glViewport(0, 0, w, h);

x, y, w, h

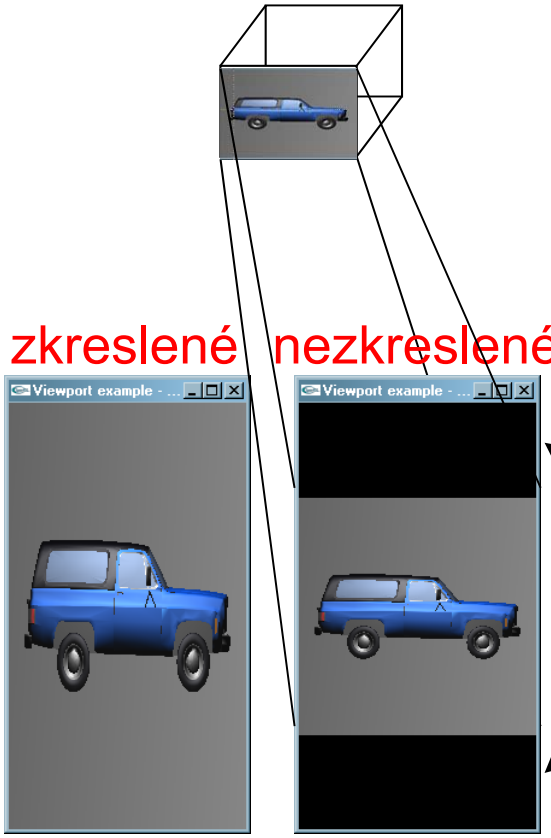
w > h

nezkreslené



glViewport((w-h)/2, 0, h, h);

Nevyužitá plocha



w <= h

Zachování poměru stran obrazu

b) protažením projekce (zvětší se pohledový objem)

```
GLfloat aspect = (GLfloat) w / (GLfloat) h;

if( aspect > 1.0 ) // w > h => okno široké
  m = glm::ortho(-1.15*aspect, 1.15*aspect, -1.15, 1.15, -10, 10);
else                    // left, right, bottom, top, near, far
  m = glm::ortho(-1.15, 1.15, -1.15/aspect, 1.15/aspect, -10, 10);
    glm::frustum(-1.15, 1.15, -1.15/aspect, 1.15/aspect, 0.1, 10);
```

Obdobně: `glm::perspective(60.0, aspect, 0.1, 10);`

zkreslené

nezkreslené

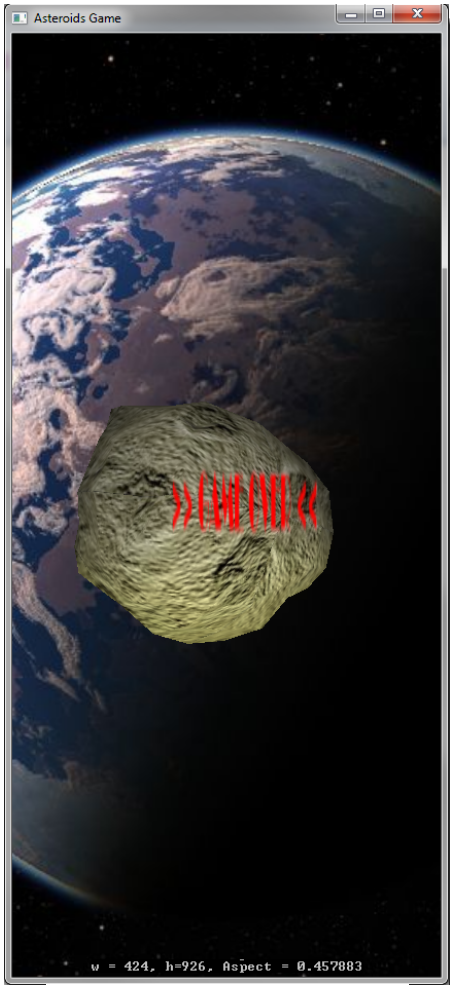
w > h => okno široké **left*aspect** **right*aspect**

Aspect

aspect = w / h

glm::perspective(60.0, aspect, 0.1, 10);

aspect = 2.7



aspect = 0.5

Rotace podle Eulerových úhlů a Gimbal lock

Rotace dle Eulerových úhlů

- Eulerovy úhly = pitch-yaw-roll (výška, kurs, rotace)
- komplexní otočení rozdělíme na otočení dle os X, Y, Z

X

$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Rotate(θ, 1, 0, 0)

Y

$$\begin{vmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

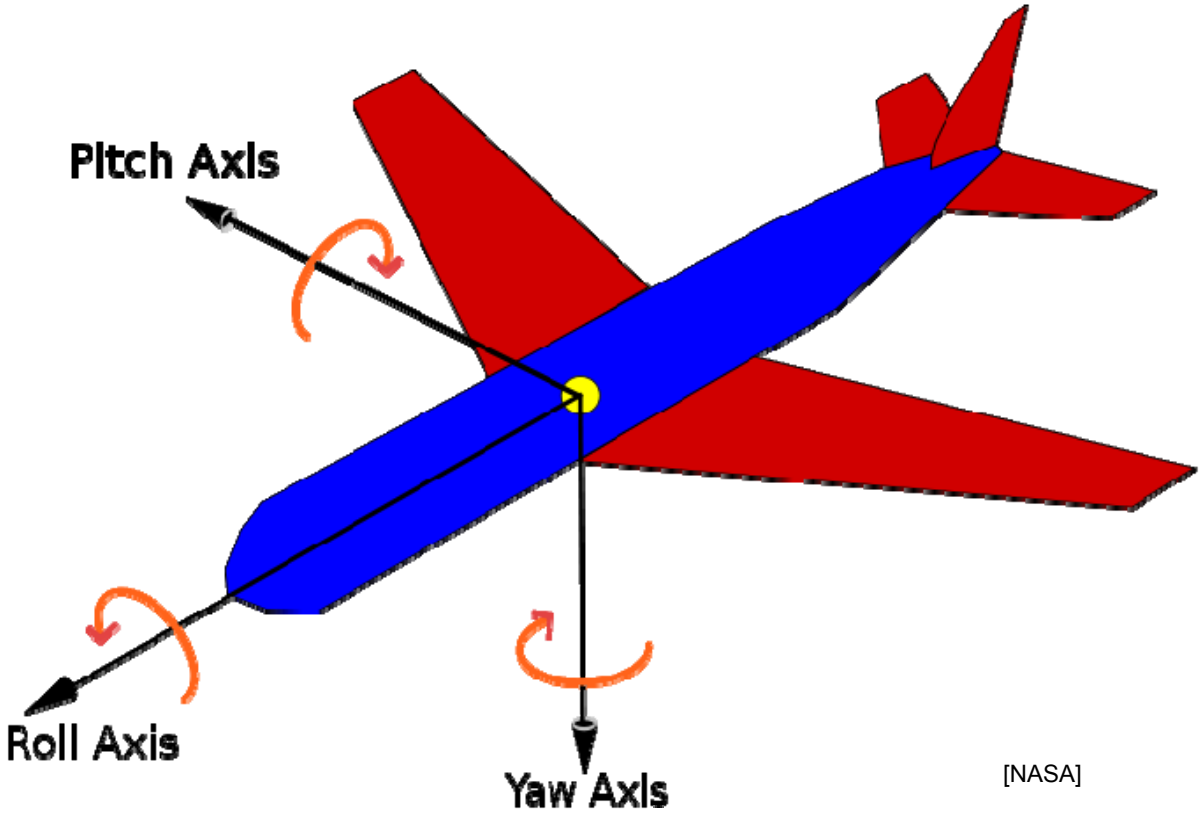
Rotate(θ, 0, 1, 0)

Z

$$\begin{vmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Rotate(θ, 0, 0, 1)

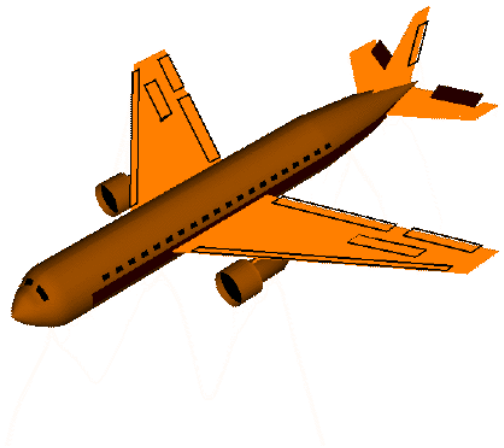
Základní orientace „Pitch-Yaw-Roll“



[NASA]

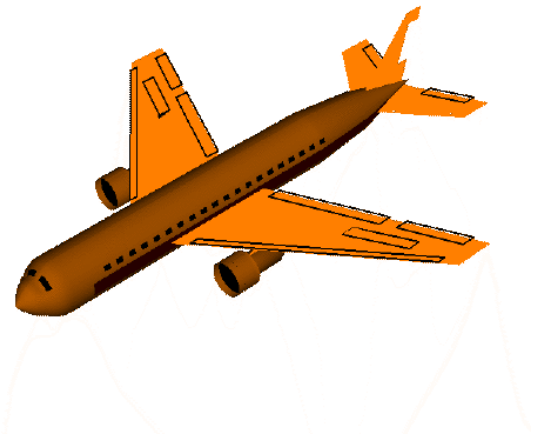
Základní orientace „Pitch-Yaw-Roll“

Pitch



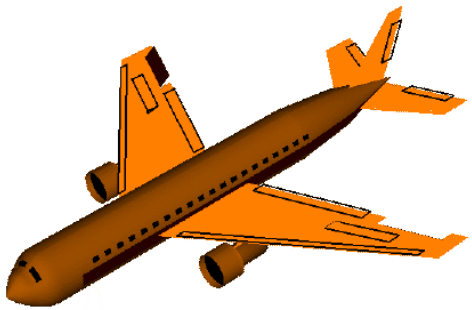
Výškovky

Yaw



Kormidlo

Roll

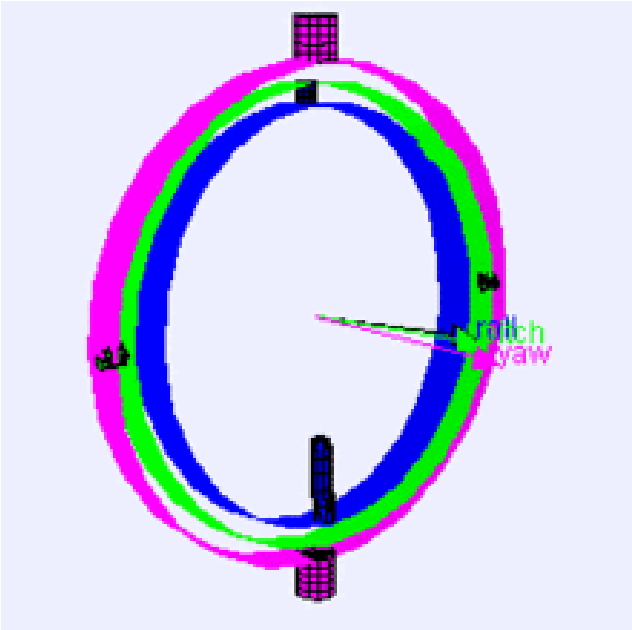


Křidélka

[NASA]

Gimbal [‘džimbl] – otočná podpěra

- Tři gimbals spojené dohromady + závaží či setrvačník

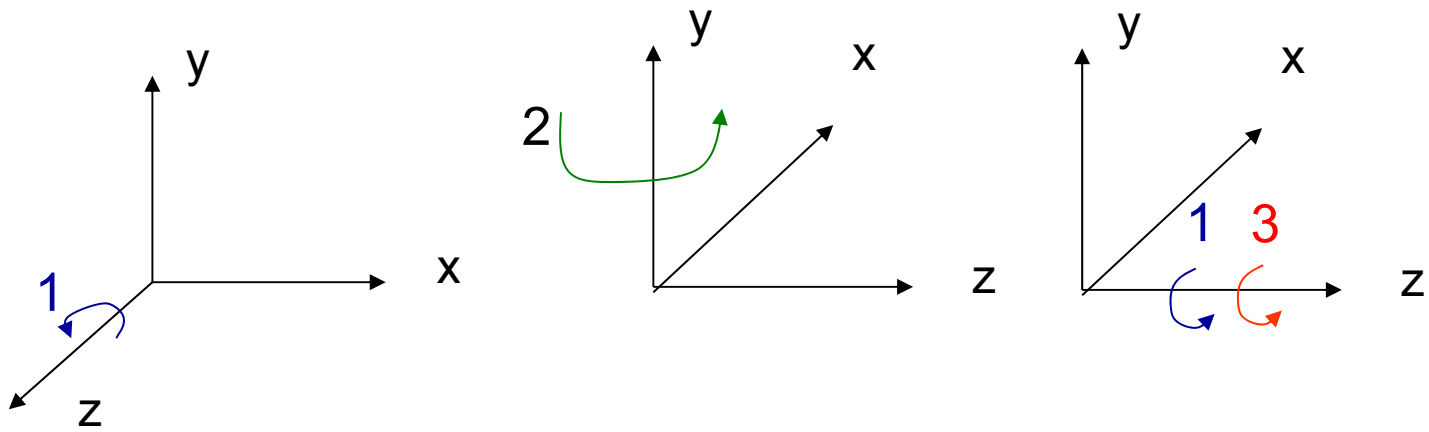


- Původní použití u gyroskopů a upevnění kompasů, kamen a sklenic na lodích od antiky.

[Wikipedia]

Rotace a „Gimbal lock“ [‘džimbl lok]

- Eulerovy úhly = pitch-yaw-roll (výška, kurs, rotace)
- komplexní otočení rozdělíme na otočení podle os X, Y, Z
- dělá se postupně => otočením druhé osy v hierarchii o 90° splynou osy a dojde ke ztrátě jednoho stupně volnosti (gimbal lock)



Pořadí rotací v programu:

Rotate(a, 1, 0, 0) // X 3

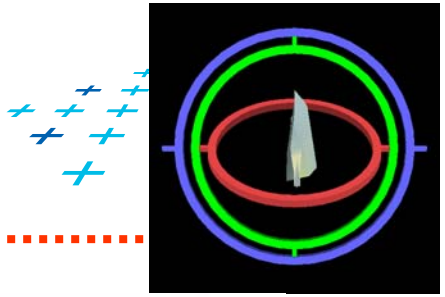
Rotate(90, 0, 1, 0) // Y 2

Rotate(c, 0, 0, 1) // Z 1

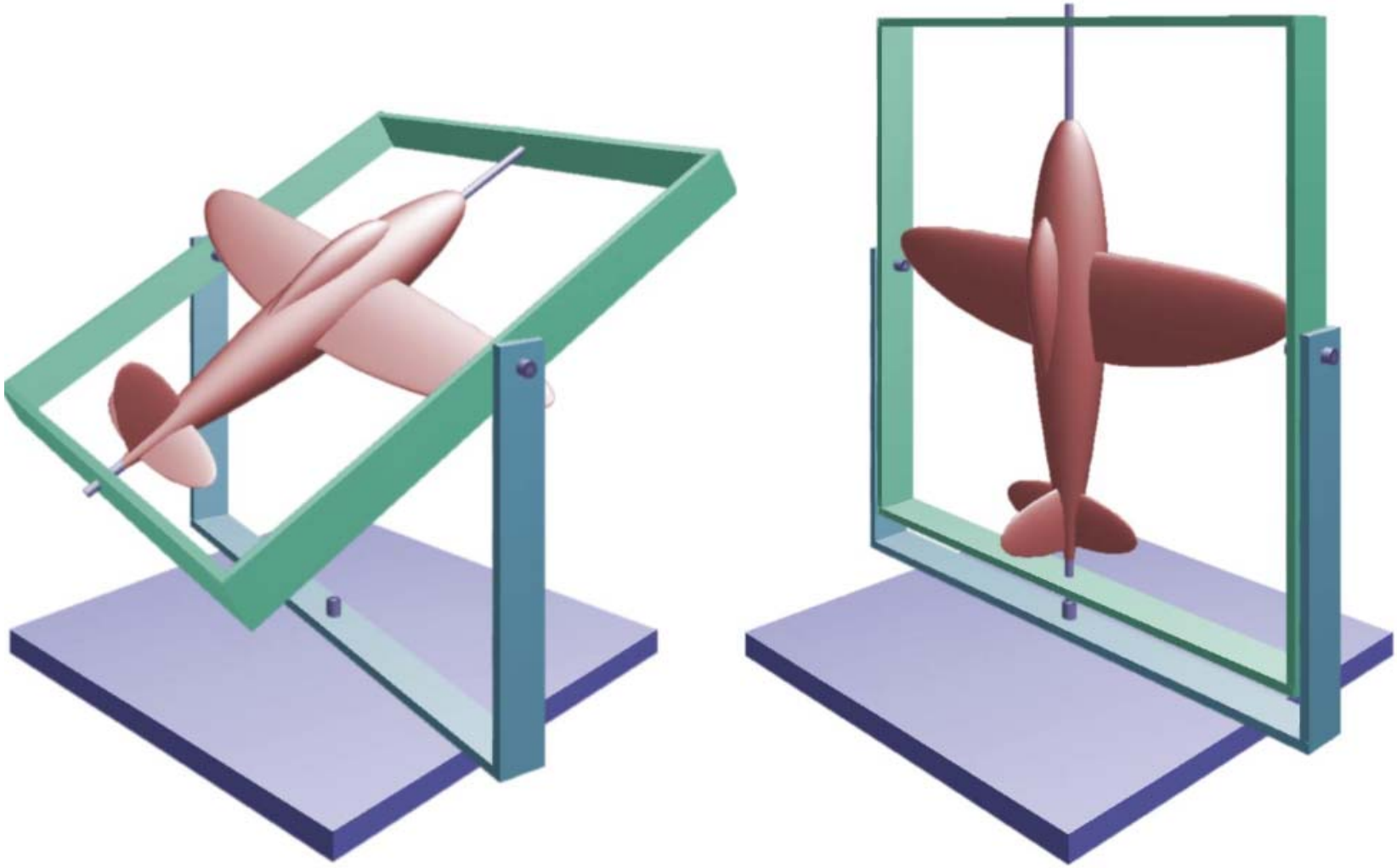
Z splyne s X

Rotace kolem X s modelem otočeným o 90°dle Y dělá totéž, co rotace kolem Z

Ztráta jednoho stupně „Gimbal Lock“



[Wikipedia]



[http://www.fho-emden.de/~hoffmann/gimbal09082002.pdf]

Vysvětlení

```
rax = RotationTransform[α, {1, 0, 0}]
```

$$\text{TransformationFunction}\left[\left(\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & \text{Cos}[\alpha] & -\text{Sin}[\alpha] & 0 \\ 0 & \text{Sin}[\alpha] & \text{Cos}[\alpha] & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)\right]$$

```
rbz = RotationTransform[β, {0, 0, 1}];
r90y = RotationTransform[Pi / 2, {0, 1, 0}];
```

Vysvětlení

Composition[rax, r90y, rbz] = rax * r90y * rbz

$$= \mathrm{Sin}[\alpha+\beta]$$

$$\left(\begin{array}{ccc|c} 0 & 0 & 1 & 0 \\ \mathrm{Cos}[\beta]\,\mathrm{Sin}[\alpha]+\mathrm{Cos}[\alpha]\,\mathrm{Sin}[\beta] & \mathrm{Cos}[\alpha]\,\mathrm{Cos}[\beta]-\mathrm{Sin}[\alpha]\,\mathrm{Sin}[\beta] & 0 & 0 \\ -\mathrm{Cos}[\alpha]\,\mathrm{Cos}[\beta]+\mathrm{Sin}[\alpha]\,\mathrm{Sin}[\beta] & \mathrm{Cos}[\beta]\,\mathrm{Sin}[\alpha]+\mathrm{Cos}[\alpha]\,\mathrm{Sin}[\beta] & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)$$

$$= \left(\begin{array}{ccc|c} 0 & 0 & 1 & 0 \\ \mathrm{Sin}[\alpha+\beta] & \mathrm{Cos}[\alpha+\beta] & 0 & 0 \\ -\mathrm{Cos}[\alpha+\beta] & \mathrm{Sin}[\alpha+\beta] & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)$$

- Otočením o 90º okolo osy Y (prostřední při skládání transformací) se ztratí možnost otáčení okolo osy X (osa X splyne s osou Z).
- Místo toho se otáčí okolo osy Z původního modelu. Výsledný úhel otočení vznikne jako součet požadovaných úhlů otočení kolem X a Z

Jak nezpůsobit gimbal lock

- Rotaci podle obecné osy (místo skládání rotací podle Eulerových úhlů)
- Používat kvaterniony (*quaternions*)